Horizon3.ai User Documentation

None

Horizon3.ai

None

Table of contents

1. Documentation Home 6
1.1 Downloads 8
2. Quickstart 10
2.1 Quickstart Guide for NodeZero 10
2.2 Register an Account 11
2.3 Network Requirements 13
2.4 Set Up NodeZero Host 23
2.5 Validate NodeZero Host 54
2.6 Run an Internal Pentest 56
3. Portal 65
3.1 NodeZero Portal 65
3.2 Tests & Assessments 67
3.3 Features 151
3.4 Settings 251
3.5 NodeZero Runner 316
3.6 Deployment Strategy 344
4. Tripwires 353
4.1 NodeZero Tripwires 353
4.2 Getting Started with NodeZero Tripwires 355
4.3 Tripwire Management 357
4.4 Tripwire Alerts 360
4.5 Testing 361
4.6 Settings 369
4.7 AD Tripwires 378
5. Insights 427
5.1 NodeZero Insights 427
6. Rapid Response 432
6.1 Rapid Response 432
6.2 Notifications 434
6.3 Portal View 436
6.4 Rapid Response Details 439
6.5 Asset Details 444
6.6 Targeted Tests 446
6.7 Managing Rapid Response 450

7. CLI 452
7.1 NodeZero CLI 452
7.2 Getting Started 453
7.3 CLI Guides 462
8. API 479
8.1 NodeZero API 479
8.2 Getting Started 480
8.3 GraphQL API Reference 510
8.4 Using the API 517
8.5 Using the API: Pagination 517
8.6 Using the API: Metadata 526
8.7 Setup 530
8.8 Setup: Op Config 530
8.9 Setup: Op Templates 548
8.10 Setup: Runners 566
8.11 Setup: Schedules 597
8.12 Setup: Asset Groups 617
8.13 Setup: Asset Groups: Domains 626
8.14 Setup: Asset Groups: IPs 637
8.15 Setup: Asset Groups: Authorization 647
8.16 Pentesting 660
8.17 Pentesting: Ops 660
8.18 Pentesting: Pentests 698
8.19 Pentesting: Impacts 706
8.20 Pentesting: Weaknesses 724
8.21 Pentesting: Proofs 746
8.22 Pentesting: Credentials 749
8.23 Pentesting: Data Stores 759
8.24 Pentesting: Hosts 768
8.25 Pentesting: EDR 776
8.26 Pentesting: External Domains 777
8.27 Pentesting: LDAP Domains 777
8.28 Pentesting: Cloud Resources 778
8.29 Pentesting: Applications 779
8.30 Pentesting: Services 780
8.31 Pentesting: Action Logs 781
8.32 Pentesting: Action Log V2 793
8.33 Pentesting: Op Compare 797

8.34 Vuln Management Hub 804
8.35 Vuln Management Hub: Weakness Series 804
8.36 Vuln Management Hub: Assets 833
8.37 Vuln Management Hub: Perspectives 838
8.38 Vuln Management Hub: Campaigns 846
8.39 Platform Features 859
8.40 Platform Features: Rapid Response 859
8.41 Platform Features: Tripwires 868
8.42 Platform Features: Vuln Risk Intelligence 882
8.43 Platform Features: CSV Format 918
8.44 Platform Features: Inventory 942
8.45 Admin 995
8.46 Admin: Account Management 995
8.47 Admin: MSP 1008
9. Release Notes 1034
9.1 Release Notes 1034
9.2 2026.06 1050
9.3 2026.05 1054
9.4 2026.04 1056
9.5 2026.03 1059
9.6 2026.02 1062
9.7 2026.01 1065
10. Downloads 1067
10.1 Downloads 1067
10.2 NodeZero Host Virtual Machine (OVA/VHD) 1069
10.3 Ubuntu Setup Script 1078
10.4 Host Utility 1081
10.5 Validate Host 1084
10.6 NodeZero App for Splunk 1087
10.7 Verify Download Authenticity 1091
11. Knowledge Base 1094
11.1 Knowledge Base 1094
11.2 NodeZero Modules 1095
11.3 Glossary 1099
11.4 Sensitive Data 1102
11.5 Weaknesses 1104
11.6 Exposure Score Levels 1124
11.7 Attack Configurations 1125

12. Credits 1136

12.1 Credits 1136

1. Documentation Home

All the info you need to deploy, configure, and maximize the effectiveness of NodeZero, our autonomous penetration testing platform. Whether you're a security analyst, system administrator, or DevOps engineer, this is where you'll find step-by-step guides, integration instructions, and troubleshooting resources to help you identify, prioritize, and remediate weaknesses across your environment.

Here's a suggested sequence to put NodeZero to work securing your environment – scroll down for more resources.



Get started

Choose your starting point

- Contact Horizon3 Sales
- Start on AWS Marketplace



Register your account

Enable access to NodeZero



Log in to the NodeZero Portal

Jump into our cloud UI



Verify network requirements

Confirm communication between the NodeZero platform and your local host



Quickstart with OVA

Use our preconfigured virtual machine to simplify local deployment



Validate your host

Run our preconfigured script to confirm compatibility



Run your first pentest

An internal test simulates an attack on your network from within

NodeZero | Essentials

Quickstart

Set up your account, create a host, and run your first pentest in just a few steps.

NodeZero Portal

Explore NodeZero Portal capabilities, including penetration testing, compliance, and advanced features.

Premium Add-Ons

Unlock NodeZero's premium add-on features to elevate your security testing with deeper insights and more powerful tools.

NodeZero | Additional References

Release Notes

View the latest updates, fixes, and new features in the Release Notes.

API Reference

Access detailed documentation for interacting with NodeZero programmatically via APIs.

CLI Reference

Access detailed documentation for interacting with NodeZero through the command-line interface.

Knowledge Base

Find solutions to common issues and learn how to optimize and debug NodeZero.

Resources

Browse webinars and other materials to enhance your understanding of NodeZero.

Contact Us

Reach out to our team for support, feedback, or inquiries about Horizon3.ai and NodeZero.

1.1 Downloads

Offline Documentation

PDF Version of this site for offline viewing.

2. Quickstart

2.1 Quickstart Guide for NodeZero

Ready to strengthen your security? Set up your account, create a host, and run your first pentest in just a few steps!

2.1.1 Getting Started

Get up and running with NodeZero quickly by following this step-by-step guide. You'll learn how to set up your account, create a host, and run your first penetration test, giving you immediate insights into your security posture. Whether you're new to penetration testing or an experienced security professional, this guide will help you deploy NodeZero efficiently and start identifying and prioritizing weaknesses right away.

What's Needed to Run Your First Penetration Test?

To perform your first pentest with NodeZero, you'll need to:

1. **Log In or Register an Account**
   Use an existing account, or register a new NodeZero account and log in to the platform.
2. **Network Requirements**
   Check network prerequisites for NodeZero to work properly.
3. **Set Up NodeZero Host**
   Deploy a NodeZero host inside your network to perform internal security assessments.
4. **Run Internal Pentest**
   Run a test, let NodeZero uncover weaknesses, and review the findings.

2.2 Register an Account

2.2.1 Log In or Register an Account

To start using NodeZero, you must log in or register an account.

Existing Users

If you have an account, log in to the NodeZero Portal.

New Users

1. If you've been invited to create a new trial account, follow the link you've been provided to complete the registration form.

a. Fill in any required fields, such as your business email and company information.

b. A contact list, also known as a distribution group (i.e. admin@company-domain.com) is not permitted.

c. Click **Create Your Account** to proceed.

2. A temporary password will be sent to your email.

a. Enter the temporary password and create a new one.

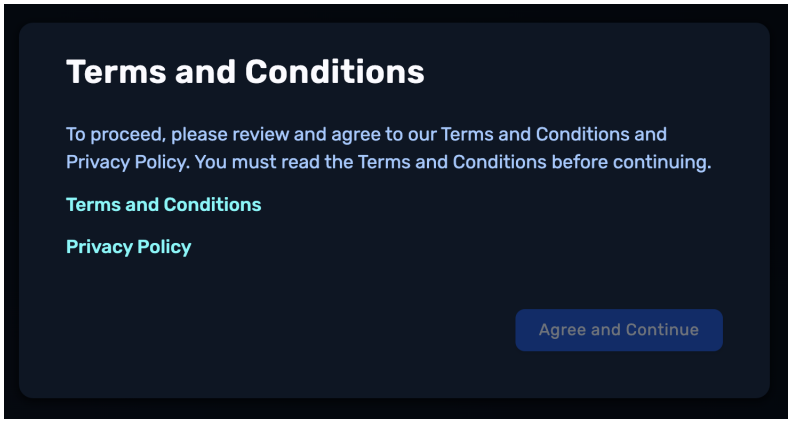
b. Click **Set Your Password** to finalize the process.

3. Enable Two-Factor Authentication (2FA)

a. Scan the QR code with an authenticator app (e.g., Google Authenticator, Okta Verify).

4. Terms and conditions.

a. Upon initial login to NodeZero review and accept the terms and conditions.



5. Verify your email.

a. Check your inbox for a verification email from NodeZero.

b. Click the verification link to confirm your email address.

6. Log in to NodeZero Portal.

a. Return to the NodeZero Portal.

b. Sign in using your Google or Microsoft business email, or enter your credentials manually.

c. Once logged in, you'll be redirected to the NodeZero Portal.

Next Step

2 Network Requirements

Check network prerequisites for NodeZero to work properly.

2.3 Network Requirements

2.3.1 Network Requirements

To enable communication between a NodeZero host and the Horizon3.ai SaaS platform, you must configure specific network settings. This section details the required outbound and inbound connections, ensuring that the host can download, execute, and report on NodeZero pentests effectively.

Requirements vary based on the NodeZero **Portal instance** (US or EU) that is generating the test, and not on the host's location. Your network must maintain uninterrupted network access to all listed endpoints during the entire pentest operation.

> **Restricted network environments**
>
> For users with restricted environments, who need to make exceptions in their firewall for NodeZero to reach its SaaS infrastructure: You have the option to use the NodeZero Gateway, which reduces the number of firewall exceptions needed. To enable the NodeZero Gateway on an account, contact your Sales or Customer Success representative.

> **Are you using a proxy?**
>
> If your environment uses a proxy for internet access, configure the NodeZero host accordingly to ensure proper communication. See the Proxy setup guide.

Outbound Traffic

Outbound network access depends on the NodeZero Portal instance generating the test, rather than the location of the NodeZero host.

REQUIREMENTS BY PORTAL REGION

Requirements are grouped by **Portal region**. Choose a region to view requirements:

US-based Portal EU-based Portal

🇺🇸 US-based Portal

portal.horizon3ai.com

Ensure uninterrupted outbound access to these endpoints during NodeZero operations.

Port/Protocol	Endpoints	Purpose

HTTPS - 443/ TCP	`gateway.horizon3ai.com`
	`interact.gateway.horizon3ai.com`
	`api.gateway.horizon3ai.com`
	`registry.gateway.horizon3ai.com`
	`api.horizon3ai.com`
	`cognito-identity.us-east-2.amazonaws.com`
	`cognito-idp.us-east-2.amazonaws.com`
	`downloads.horizon3ai.com`
	`sqs.us-east-2.amazonaws.com`
	`*.ecr.us-east-2.amazonaws.com`
	`*.queue.amazonaws.com`
	`*.s3.amazonaws.com`
	`*.s3.us-east-1.amazonaws.com`
	`*.s3.us-east-2.amazonaws.com`
	`*.s3-w.us-east-2.amazonaws.com`
	`*.ubuntu.com`
	`*.canonical.com`
	`*.interacth3.io`
	
	`(Deprecated) *.docker.com`
	`(Deprecated) *.docker.io`
	`(Deprecated) docker-images-prod.6aa30f8b08e16409b46e0173d6de2f56.r2.cloudflarestorage.com`
		API access, authentication, storage, updates, and container registry

HTTP - 80/ TCP	`*.interacth3.io`	Interactive test communication

🇪🇺 EU-based Portal

portal.horizon3ai.eu

Ensure uninterrupted outbound access to these endpoints during NodeZero operations.

Port/Protocol	Endpoints	Purpose

HTTPS - 443/ TCP	`gateway.horizon3ai.eu`
	`interact.gateway.horizon3ai.eu`
	`api.gateway.horizon3ai.eu`
	`registry.gateway.horizon3ai.eu`
	`api.horizon3ai.eu`
	`cognito-identity.eu-central-1.amazonaws.com`
	`cognito-idp.eu-central-1.amazonaws.com`
	`downloads.horizon3ai.com`
	`sqs.eu-central-1.amazonaws.com`
	`*.ecr.eu-central-1.amazonaws.com`
	`*.queue.amazonaws.com`
	`*.s3.amazonaws.com`
	`*.s3.eu-central-1.amazonaws.com`
	`*.s3-w.eu-central-1.amazonaws.com`
	`*.s3-r-w.eu-central-1.amazonaws.com`
	`*.ubuntu.com`
	`*.canonical.com`
	`*.interacth3.eu`
	
	`(Deprecated) *.docker.com`
	`(Deprecated) *.docker.io`
	`(Deprecated) docker-images-prod.`
		API access, authentication, storage, updates, and container registry

HTTPS/SSL/TLS Inspections

SSL/TLS packet inspection may cause cURL commands to fail due to certificate mismatches - To avoid this, consider making an exception or disabling packet inspection for the Nodezero host.

OVA OUTBOUND NETWORKING REQUIREMENTS

For hosts set up using the NodeZero OVA, some additional endpoints are required:

US-based Portal EU-based Portal

US-based Portal

portal.horizon3ai.com

Port/Protocol	Endpoints	Purpose
HTTPS - 443/ TCP	`downloads.horizon3ai.com`	CLI/Runner updates and host utility updates
HTTP - 80/TCP	`*.ubuntu.com` `*.canonical.com`	Host library updates
ICMP	`downloads.horizon3ai.com` `gateway.horizon3ai.com` `interact.gateway.horizon3ai.com` `api.gateway.horizon3ai.com` `registry.gateway.horizon3ai.com`	Used to Ping for connectivity

EU-based Portal

portal.horizon3ai.eu

Port/Protocol	Endpoints	Purpose
HTTPS - 443/ TCP	`downloads.horizon3ai.com`	CLI/Runner updates and host utility updates
HTTP - 80/TCP	`*.ubuntu.com` `*.canonical.com`	Host library updates
ICMP	`downloads.horizon3ai.com` `gateway.horizon3ai.eu` `interact.gateway.horizon3ai.eu` `api.gateway.horizon3ai.eu` `registry.gateway.horizon3ai.eu`	Used to Ping for connectivity

These are needed for receiving OS and CLI updates

Inbound Traffic

TCP AND UDP PORTS

OVA is pre-configured

OVA users can skip setting up the inbound settings. The OVA is pre-configured with the necessary inbound ports already open

To simulate internal attacks, the NodeZero Host must allow inbound traffic on specific ports. These settings apply to the host itself, and not on perimeter firewalls.

Protocol	Ports
TCP	21, 23, 25, 53, 80, 88, 110, 135, 139, 143, 389, 443, 445, 587, 1433, 3306, 3389, 5900, 5985, 8002, 8080, 8443, 8888, 28069, 45000-49999
UDP	69

Do not punch extra holes

Do not alter your network beyond normal operations during a pentest. NodeZero simulates an attacker and does not require extra paths. For example, if your firewall blocks the marketing VLAN from the finance VLAN, keep it that way—NodeZero will validate this restriction.

NodeZero Gateway

For networks with restricted access, the NodeZero Gateway streamlines **outbound traffic** by routing through static IPs.

Enable NodeZero Gateway

NodeZero Gateway is available exclusively to paid customers. To enable the NodeZero Gateway for your account, please contact your Sales or Customer Success representative.

WHEN TO USE GATEWAYS

NodeZero Gateways are ideal for networks with strict security policies or outbound traffic restrictions. By routing traffic through static IPs, this feature simplifies network configuration, ensuring consistent and secure connectivity. It is especially beneficial for organizations that need to comply with strict firewall or proxy settings.

NodeZero Gateways are region-specific. Choose a region to view endpoints:

US-based NodeZero Gateway | EU-based NodeZero Gateway | FedRAMP High NodeZero Gateway

US-based NodeZero Gateway

Domains	Static IPs	Port/Protocol

`gateway.horizon3ai.com`
`interact.gateway.horizon3ai.com`
`api.gateway.horizon3ai.com`
`registry.gateway.horizon3ai.com`
	15.197.206.82
	3.33.191.122
		HTTPS - 443/TCP

Additional endpoint for using the Runner/CLI

Domains	Ports/Protocols	Purpose
`downloads.horizon3ai.com`	**HTTPS - 443/TCP**	Initial install of CLI and updates

Additional endpoints for using the OVA/VHD with Runner

Domains	Ports/Protocols	Purpose
`downloads.horizon3ai.com`	**HTTPS - 443/TCP**	Initial install of CLI and updates
`*.ubuntu.com` `*.canonical.com`	**HTTP - 80/TCP**	Host library updates

EU-based NodeZero Gateway

Domains	Static IPs	Ports/Protocols

`gateway.horizon3ai.eu`
`interact.gateway.horizon3ai.eu`
`api.gateway.horizon3ai.eu`
`registry.gateway.horizon3ai.eu`
	52.223.20.205
	99.83.187.197
		HTTPS - 443/TCP

Additional endpoint for using the Runner/CLI

Domains	Ports/Protocols	Purpose
`downloads.horizon3ai.com`	**HTTPS - 443/TCP**	Initial install of CLI and updates

Additional endpoints for using the OVA/VHD with Runner

Domains	Ports/Protocols	Purpose
`downloads.horizon3ai.com`	**HTTPS - 443/TCP**	Initial install of CLI and updates
`*.ubuntu.com` `*.canonical.com`	**HTTP - 80/TCP**	Host library updates

FedRAMP High NodeZero Gateway

Domains	Static IPs	Ports/Protocols

`gateway.gov-horizon3ai.com`
`interact.gateway.gov-horizon3ai.com`
`api.gateway.gov-horizon3ai.com`
	15.205.105.134
	56.136.174.32
	15.200.230.187
		HTTPS - 443/TCP

Additional endpoint for using the Runner/CLI

Domains	Ports/Protocols	Purpose
`downloads.horizon3ai.com`	**HTTPS - 443/TCP**	Initial install of CLI and updates

Additional endpoints for using the OVA/VHD with Runner

Next Step

3 **Set Up NodeZero host**

Deploy a NodeZero host inside your network to perform internal security assessments.

2.4 Set Up NodeZero Host

2.4.1 Set Up NodeZero Host

The NodeZero Host is a critical component of the NodeZero solution, enabling internal penetration testing to identify vulnerabilities within your network. This guide walks you through setting up and configuring a NodeZero Host.

What Is a NodeZero Host?

A NodeZero Host is a dedicated system that facilitates internal penetration testing within your network. It serves as the entry point for NodeZero's automated security assessments, allowing organizations to identify weaknesses inside their infrastructure. Unlike external tests that evaluate internet-facing assets, a NodeZero Host is essential for internal pentests, where threats originate from within a corporate network.

Deployment Strategy

The effectiveness of your NodeZero Host depends on its **Placement** and **Intent**, which determine the scope and focus of your pentest:

- **Placement**: Specifies the location of the NodeZero Host within your network.
- **Intent**: Sets the scope and objectives of the pentest.

Explore the Deployment Strategy page for a comprehensive overview of strategies to optimize your pentesting approach.

Deployment Strategy

Plan your NodeZero Host's placement and intent to identify vulnerabilities.

This quickstart guide is focused on **internal pentests**, we recommend placing the host **inside your target network scope**.

Methods to Deploy the Host

A NodeZero Host can be deployed using two methods: the **NodeZero OVA** (recommended) for quick setup in virtual environments, or **manual installation** on a Linux system. Select the method that best suits your infrastructure, with OVA being the preferred option for a streamlined installation and automatic updates.

CHOOSE ONE:

Preconfigured OVA — Recommended

Use a Preconfigured Open Virtual Appliance (OVA) VM image for quick deployment in virtual environments (e.g. VMware, VirtualBox).

- Includes a utility to keep the NodeZero Host updated.
- Includes NodeZero Command Line Interface (CLI).

Convert Ubuntu to Host

Upgrades any Ubuntu 20.04/24.04 to be a NodeZero host ready to run NodeZero.

- Installs a utility to keep the NodeZero Host updated.
- Installs the NodeZero Command Line Interface (CLI).

Manual Installation

Install the NodeZero Host manually on Linux for more control. This method requires dependency setup and configuration.

- Direct setup on a Linux-based system.
- Install using a script or manual package setup.

2.4.2 Preconfigured OVA

Preconfigured OVA NodeZero Host

The **NodeZero OVA** host is a preconfigured virtual machine designed for quick and reliable deployment in **VMware, VirtualBox, and Hyper-V**. It simplifies installation by bundling all required dependencies, reducing setup complexity while ensuring optimal performance.

When to Use Preconfigured OVA

This solution is ideal for quick setup, virtualization support, and robust security.

- **Preconfigured for immediate use** – No manual dependency installation required.
- **Optimized for virtualized environments** – Supports **VMware, VirtualBox, and Hyper-V**.
- **Hardened security** – Configured with best practices for security and stability.

For a more customized deployment, consider using the Manual Installation Process which offers greater flexibility and control. This method requires knowledge of virtualization platforms and manual configuration steps. If you're comfortable with these steps, the manual installation will allow you to customize the environment to meet your specific needs.

Deployment steps

Follow these steps to set up your **NodeZero OVA** host:

1 Download OVA Host

Get the latest NodeZero OVA file from the Portal.

2 Set Up & Configure

Import and configure the OVA in your virtualized environment.

3 Validate Host

Test connectivity and Docker to confirm readiness.

Having Trouble?

Check out Common Setup Issues for troubleshooting tips.

Download OVA Host

The NodeZero OVA host is a lightweight, pre-configured Ubuntu 24.04 virtual machine designed to run NodeZero penetration tests. It includes essential tools for pentest execution, along with built-in utilities for debugging, maintenance, and system monitoring.

OVA IMAGES

Two types of OVA images are available to accommodate different virtualization environments:

- VMWare/VirtualBox OVA - Preconfigured for VMware and VirtualBox.
- Windows Hyper-V VHD - Specifically formatted for Windows Hyper-V environments.

CHOOSE AN IMAGE TO DOWNLOAD

VMWare/ Virtualbox

Preconfigured OVA image for quick deployment in VMware and VirtualBox.

Download SHA256

Windows Hyper-V

Windows Hyper-V using the VHD format for efficient virtualization setups.

Download SHA256

VERIFY DOWNLOAD AUTHENTICITY

To ensure the integrity and authenticity of your files, always verify that downloads originate from **Horizon3.ai**.

Every download distributed here includes:

- A **download file** (the actual tool or resource).
- A **checksum file** containing a **SHA256 hash** to confirm the file's authenticity and integrity.

Given the nature of cybersecurity, we strongly recommend verifying any file claiming to come from **Horizon3.ai** against the official checksums posted on this site.

CHOOSE YOUR OPERATING SYSTEM

Select a tab below to view step-by-step instructions.

Linux macOS Windows Windows (manual)

Linux

Ensure that the **downloaded image file** and the `sha256sum` **checksum file** are in the same directory.

Run the following command in the directory:

```
sha256sum -c *.sha256.checksum
```

Sample output:

```
nodezero@nodezero:~$ sha256sum -c *.sha256.checksum
NodeZero-1717717928.ova: OK
checkenv.sh: OK
nodezero@nodezero:~$
```

Note

Most Linux distributions come with a `sha256sum` command line utility that reads a checksum file, computes the checksum against the file it represents, and returns `OK` if they match.

MacOS

Ensure that the **downloaded image file** and the `sha256sum` **checksum file** are in the same directory.

Run the following command:

```
shasum -c *.sha256.checksum
```

Sample output:

```
horizon3@mac:~$ shasum -c *.sha256.checksum
NodeZero-1717717928.ova: OK
horizon3@mac:~$
```

Note

Most macOS systems include a `shasum` command-line utility that reads a checksum file, computes the checksum against the file that it represents, and returns `OK` if they match.

Windows

Ensure that the **downloaded image file** and the `sha256sum` **checksum file** are in the same directory.

Run the following command:

```
Get-Content -Path .\*.sha256.checksum | % {$Hash, $_, $File = $_.Split(" ", 3); if ($Hash) { [PSCustomObject]@{Path=$File;
Result=if ((Get-FileHash -Path $File -Algorithm SHA256).Hash -eq $Hash) { "OK" } else { "FAILED" }}}} | Format-List
```

Sample output:

```
PS C:\Users\horizon3\Downloads> Get-Content -Path .\*.sha256.checksum | % {$Hash, $_, $File = $_.Split(" ", 3); if ($Hash) {
[PSCustomObject]@{Path=$File; Result=if ((Get-FileHash -Path $File -Algorithm SHA256).Hash -eq $Hash) { "OK" } else {
"FAILED" }}}} | Format-List


Path   : checkenv.sh
Result : OK

Path   : NodeZero-1717717928.ova
Result : OK

PS C:\Users\horizon3\Downloads>
```

Note

Windows PowerShell has its own checksum utility.

NEXT STEP

2 Setup & Configure OVA

Import and configure the OVA in your virtualized environment.

Set Up and Configure OVA

After downloading the NodeZero OVA, follow these steps to set it up and configure your environment.

SPECIFICATIONS

The NodeZero host virtual machine comes preconfigured to use these resources:

- 2 x CPUs
- 8GB of RAM
- 128GB of disk (minimum required: 40GB)
- Bridged network adapter

INSTALLATION

Installing the virtual machine is as simple as **importing the OVA file** into your virtualization environment. Below, we provide step-by-step instructions for deploying **NodeZero** using **vSphere**, **VirtualBox** or **Hyper-V**.

Choose your virtualization environment and follow the view step-by-step instructions:

VMWare vSphere VirtualBox Windows Hyper-V

Deploying NodeZero on VMWare vSphere

The vSphere client is one of VMWare's virtual environment management solutions. You can find more information on the client itself in VMWare's documentation.

Note

The following steps are for vSphere client version `7.0.3.00500`.

After downloading and verifying the most recent `NodeZero-####.ova` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Log into your `VMware vSphere` client.
2. Open the `Actions` menu and select `Deploy OVF Template`.
3. Choose `Local File`, then click `Upload Files` and select the downloaded OVA file.
4. (Optional) Rename the VM and select a deployment location. Click `Next`.
5. Select `compute resources` for the VM. Click `Next`.
6. Verify the import settings and ensure the signature is from `Horizon3.ai`. Click `Next`.
7. Choose the `storage destination` for the VM. Click `Next`.
8. Select the `network` for the VM. Click `Next`.
9. Review all selections, then click `Finish` to start deployment.
10. Once deployment completes, select the VM from the list and click `Power On` to launch it.

Deploying NodeZero on VirtualBox

Oracle's VirtualBox is a powerful open-source virtualization solution for running multiple operating systems on a single physical machine. You can find more information in VirtualBox documentation.

After downloading and verifying the most recent `NodeZero-####.ova` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Open VirtualBox.
2. Click on `Tools`, then `Import`.
3. Enter the location of the OVA file. Click `Continue`.
4. Click `Import`, and wait for it to complete.
5. Make sure the network is set to use a `bridged network adapter`.
6. Select the newly imported NodeZero virtual machine from the list on the left.
7. Click `Settings`, then `Network`.
8. Confirm that `Attached to` is set to `Bridged Adapter`.
9. Confirm that `Name` is set to the name of the adapter connected to your internal network.
10. Click `OK`.
11. Select the NodeZero virtual machine from the list on the left.
12. Launch the VM by clicking `Start`.

Deploying NodeZero on Windows Hyper-V

Hyper-V is Window's native virtualization platform for creating and managing virtual machines on Windows systems. You can find more information in Hyper-V documentation.

After downloading and verifying the most recent `NodeZero-####.vhd` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Open and enable Hyper-V.
2. Ensure both `Management Tools` and `Platform` are enabled.
3. Create a New Virtual Machine in Hyper-V Manager by going to `Action > New > Virtual Machine...`.

Accessing Your OVA Host

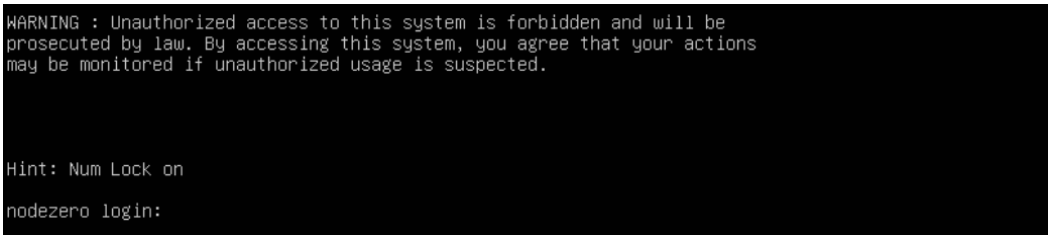
VMWare vSphere VirtualBox / Hyper-V

Once the NodeZero host is powered on, you can connect using either the **Web Console** or the **Remote Console** from the vSphere client interface.

After starting the VM, a display window will appear, showing the operating system loading screen.

First-Time Login & Password Update

Regardless of the platform, once the OS has fully loaded, you will see a **login screen** similar to this:



When launching the **NodeZero Host** for the first time, **SSH password access is disabled** until the initial login is completed and the default password is updated.

Initial Login Credentials

Use the following credentials to log in:

- **Username:** `nodezero`
- **Password:** `nodezero`

Password Update Process

Upon successful login, you will see the following prompt:

```
You are required to change your password immediately (administrator enforced)
Changing password for nodezero.

Current password:
```

1. Enter the **current** password (nodezero) and press `Enter`.
2. When prompted for the `New password:`, enter a secure password and press `Enter`.
3. Confirm by entering the same password when prompted to `Retype new password:` and press `Enter`.

```
WARNING : Unauthorized access to this system is forbidden and will be
prosecuted by law. By accessing this system, you agree that your actions
may be monitored if unauthorized usage is suspected.



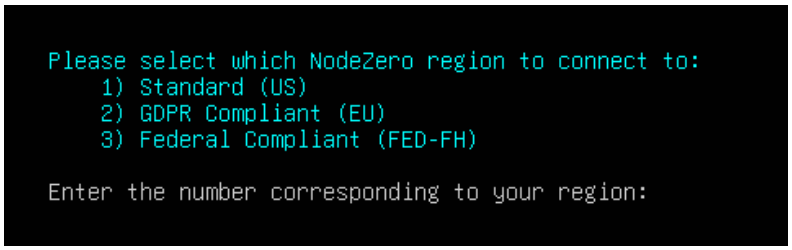
nodezero login: nodezero
Password:
You are required to change your password immediately (administrator enforced).
Changing password for nodezero.
Current password:

Passwords require min 14 characters and must include at least
1 Digit, 1 Special, 1 Upper and 1 Lower Case

New password:
Retype new password:
```

Select Region

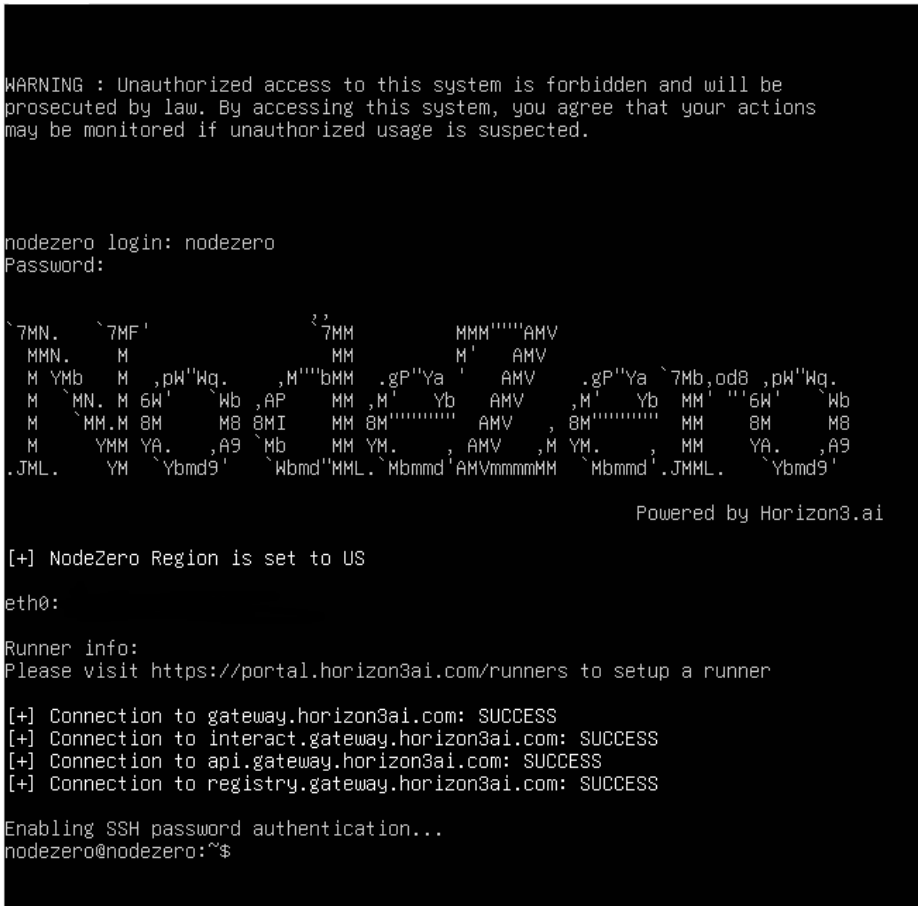
Once the password has been successfully changed, the region the host needs to connect to will need to be selected.



After specifying which region the NodeZero host will connect to, log out and log back into the system for the change to take effect. Upon logging in, a splash screen will display:

- Value of `eth0`.
- Prompt to set up a Runner.
- Status of connections to Horizon3.ai endpoints.

At this point, **SSH has been enabled** on the host, and you can access it using an SSH client. Below is a screenshot of the login screen when the host is fully set up.



Successful OVA login confirmation

SSH Access

After logging to the NodeZero Host successfully it is recommended to use SSH for access as this allows for the user to copy-paste the commands to set up a NodeZero Runner or to run a NodeZero pentest.

To connect over SSH with Linux or MacOS, run the command below, replacing `<IP_ADDRESS>` with the one shown in the login screen after `eth0`.

```
ssh nodezero@<IP_ADDRESS>
```

If using Windows, a client like PuTTY will be needed to connect. Fill out the `Host Name (or IP Address)` field with the address shown in the login screen.

> **Additional Setup Options**
>
> The n0 utility helps with configuring the host with a static ip, using a proxy, downloading the latest CLI, and setting timezones as well as running the checkenv script

CONFIGURE A NODEZERO RUNNER

Horizon3 AI strongly recommends that you set up a NodeZero Runner on your NodeZero Host. The NodeZero Runner enables you to deploy pentests from the Web UI without needing to log into each host to run the pentests. Runners also enable scheduling pentests for continuous pentests against the target network.

EDRs on NodeZero Host

Do not install or configure any EDR (Endpoint Detection and Response) services on the NodeZero host. EDRs are known to disable NodeZero once they detect the deliberately malicious actions that our tests conduct. To get a complete test of your overall environment, NodeZero attacks must launch from the host machine unimpeaded.

NEXT STEP

3 Validate Host

Test connectivity and Docker to confirm readiness.

2.4.3 Ubuntu Upgrade

Ubuntu Setup Script

> **OVA is preferred**
>
> We highly recommend using our OVA virtual machine for NodeZero installation, as it is the preferred method to ensure a seamless setup.

SET UP UBUNTU FOR RUNNING NODEZERO

If deploying the OVA is not a viable option for your infrastructure, Horizon3.ai provides a script to help set up a base Ubuntu 20.04 or 24.04 LTS OS for use with NodeZero.

MINIMUM SPECIFICATIONS

The NodeZero host requires a minimum configuration of:

- 2 x CPUs (4 x recommended)
- 8GB of RAM
- 40GB of disk (128GB recommended)
- Bridged network adapter

BEFORE SETTING UP

If your environment restricts access to external sites through an outbound proxy or similar mechanism, please make sure that it allows connections to the following sites:

- `*.ubuntu.com`
- `*.canonical.com`
- `downloads.horizon3ai.com`
- `download.docker.com` (Docker installations only)

DOWNLOADS

> **Tip**
>
> Always verify that the files you're downloading come from Horizon3.

NodeZero Ubuntu Setup Script

Download **SHA256**

RUNNING THE SCRIPT

> **User Permissions**
>
> A user with `sudo` permission is required. Do not install as `root`.

Download the script onto the host that is going to be running NodeZero, execute and run the script. After running the script it can be removed.

```
curl -o linux-build.sh https://downloads.horizon3ai.com/utilities/linux-build.sh
bash linux-build.sh
sudo rm -f linux-build.sh
```

System Updates and Reboots

Executing this script "as is" will update the OS libraries that do not require user interaction. The script will automatically reboot the OS upon completion.

WHAT THE SCRIPT DOES

This script sets up ubuntu with the required libraries for NodeZero to run successfully on the host machine. It sets the MOTD letting users that login know they are logging into a NodeZero host and it installs 2 utilities (`h3-cli` and `n0`)

h3-cli allows for communication and management of NodeZero on the host.

Host Utility enables management of the NodeZero hosts settings.

This script does the following:

- Disables journald
- Disables the firewall (ufw)
- Updates the OS
- Installs the following libraries:
- cloud-init
- open-vm-tools
- docker.io
- nmap
- tcpdump
- dnsutils
- unzip
- jq
- telnet
- Sets up Message of the Day (MOTD)
- Downloads h3-cli
- Downloads Host Utility
- Adds current user to docker group

A successful run of the script looks like this:



EDRs on NodeZero Host

Do not install or configure any EDR (Endpoint Detection and Response) services on the NodeZero host. EDRs are known to disable NodeZero once they detect the deliberately malicious actions that our tests conduct. To get a complete test of your overall environment, NodeZero attacks must launch from the host machine unimpeaded.

Validate Host

Test connectivity and Docker to confirm readiness.

2.4.4 Manual Installation

Manual Installation of a NodeZero Host

Manually installing a NodeZero Host gives you complete control over its deployment and configuration, allowing you to tailor the setup to your specific environment. This guide walks you through the process of preparing, installing, and validating a NodeZero Host to ensure it's ready for internal penetration testing with NodeZero.

WHEN TO INSTALL A HOST MANUALLY

Manual installation is recommended in the following scenarios:

- **Use of Custom Images or OS**

Organizations, particularly large enterprises, may prefer to install NodeZero on their own pre-approved **operating system image** or **OS of choice** rather than using the preconfigured OVA.

- **Reusing an Existing Virtual Machine**

If you need to install NodeZero on an **existing virtual machine** rather than provisioning a new one, manual installation allows you to configure it while following Horizon3.ai's requirements.

For faster deployment, consider using NodeZero's Preconfigured Open Virtual Appliance (OVA), which automates much of this process. Manual installation requires familiarity with Linux system administration, networking, and container runtimes (Docker or Podman). If you're new to these, the OVA method may be more suitable.

DEPLOYMENT STEPS

Follow these steps to set up your NodeZero Host manually:

1. **Host Requirements**
   Verify minimum hardware and software needs (e.g., CPU, memory, Ubuntu 20.04+).
2. **Container Runtime**
   Install Docker or Podman as your container runtime.
3. **NodeZero CLI**
   Install NodeZero CLI utility
4. **Validate Host**
   Test connectivity and container runtime to confirm readiness.

HAVING TROUBLE?

Check out Common Setup Issues for troubleshooting tips.

Host Requirements

The NodeZero host acts as the foundation for Horizon3.ai's autonomous penetration testing solution, serving as the launch point for security assessments. We strongly recommend a Linux-based system (e.g., Ubuntu), though Windows with Docker has been used successfully.

Position the host within the network segment where you want pentesting to start. Ensure the host is powered on during setup, deployment, and testing to download and run NodeZero. You may shut it down or remove it post-operation.

FOR ALTERNATIVE OPERATING SYSTEMS

Contact the Horizon3.ai team to assess compatibility. While most distributions should work, we've only validated a subset, and support may be limited for untested systems.

REQUIREMENTS

To manually set up a NodeZero host, your system **must** meet at least the minimum specifications below:

Requirement	Minimum	Recommended
Operating System	Ubuntu 20.04 LTS or later, RHEL 9+	Same as minimum If running NodeZero on an operating system not listed, Horizon3.ai will not be able to provide support in the event an issue arises.
CPU	2 cores	4 cores for optimal performance
RAM	8 GB	16 GB or more for heavy workloads
Storage	40 GB free disk space	80 GB SSD preferred for faster I/O
Network	Stable internet connection and internal network access from the host	Same as minimum
Software	Git, bash	Same as minimum
Container Runtime	Docker >= 20.10 **or** Podman >= 4.0	Same as minimum
NFS Client	Required only for test portion that attacks an NFS mount. Omission will not block remainder of the test.	Install NFS Client library For Ubuntu, this library is `nfs-common`.

EDRs on NodeZero Host

Do not install or configure any EDR (Endpoint Detection and Response) services on the NodeZero host. EDRs are known to disable NodeZero once they detect the deliberately malicious actions that our tests conduct. To get a complete test of your overall environment, NodeZero attacks must launch from the host machine unimpeded.

SCALING LARGER DEPLOYMENTS

The minimum and recommended requirements above perform well for testing up to about 5,000 assets. The following table lists the same host requirements in the first ("Small") row. The next two rows offer host scaling recommendations to support wider test coverage.

Scale	Deployment Scenario	Typical IP/ Asset Count	Host Configuration
Small	Single site or simple flat/ segmented network	< 5,000 assets	1 NodeZero host with 2–4 cores, 8–16 GB RAM, and 40–128 GB storage
Medium	Multi-site or multi-VLAN environments; hosts could be split by site or business unit	5,000–20,000 assets	1–2 NodeZero hosts; per host: 2–4 cores, 16 GB RAM, and 128 GB storage
Large	Multi-region, multi-site; plan this architecture with Horizon3.ai Customer Success	20,000+ assets	Multiple NodeZero hosts (per site/region); per host: 4 cores, 16–32 GB RAM, and 128+ GB storage

Notes:

- These are basic configurations for each scenario. Depending on network configuration, more resources might be required.
- Adding cores increases how many instances of each attack NodeZero can leverage at once. It does not directly accelerate processing speed per instance.

NEXT STEP

2 Container Runtime Installation

Install Docker or Podman as your container runtime.

Container Runtime Installation

> **Using the OVA?**
>
> Skip this section if you are using the OVA. Docker is pre-installed on the OVA.

LINUX RUNTIME OPTIONS

To run the NodeZero container, you need either **Docker** or **Podman** installed. Choose the runtime that best fits your environment.

- **Docker** is recommended for Ubuntu- and Debian-based systems.
- **Podman** is recommended for RHEL 9+ and other systems where Podman is the default container runtime. Podman 4.0+ is required.

Select a tab below to display complete installation instructions for either Docker or Podman.

Docker **Podman**

Follow these steps to install and configure Docker. For additional help, see Docker's own installation documentation for Ubuntu or Debian.

INSTALL DOCKER

Begin by updating your system and installing Docker:

```
sudo apt update
sudo apt install -y docker.io
```

Start and Enable Docker

Ensure that Docker starts automatically:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Verify Docker Installation

Check the Docker version (must be 20.10+):

```
docker --version
```

Run a test container to verify functionality:

```
sudo docker run hello-world
```

Additional Linux Distributions

If you're using a different Linux distribution, you can find corresponding installation guides in Docker's documentation:

- Install Docker on CentOS
- Install Docker on Fedora

Follow these steps to install and configure Podman on RHEL-based systems. For additional help, see the Podman documentation.

INSTALL PODMAN

Install Podman using your package manager:

```
sudo dnf install -y podman
```

Verify Podman Installation

Check the Podman version (must be 4.0+):

```
podman --version
```

Run a test container to verify functionality:

```
sudo podman run hello-world
```

Podman version and `sudo` requirements

NodeZero requires **Podman 4.0+**. Podman 4.0+ is available on RHEL 9+. Earlier versions might lack required networking and SELinux features.

NodeZero requires root-level access for mount operations and image store access. When using Podman, NodeZero always runs containers with `sudo`.

NodeZero container getting shut down prematurely?

If the container running NodeZero is getting shut down prematurely, an EDR (Endpoint Detection and Response) service might be running on the NodeZero host and blocking the execution of the container. Check for any EDRs on the NodeZero host, and disable them. This allows NodeZero to run unimpeded from the host machine.

NEXT STEP

3 H3-CLI Installation

Test connectivity and Docker to confirm readiness.

NodeZero CLI Installation

For advanced host management, the NodeZero Command Line Interface (CLI) provides additional control and flexibility.

When to Use the NodeZero CLI

Use the NodeZero CLI when you need greater control over your NodeZero host or when automating tasks and configurations through scripts. It's especially useful for advanced users who require frequent status checks or advanced management of their hosts.

> **Note**
>
> NodeZero CLI is **optional**.

Installation

Clone and configure the CLI with the following commands:

zip (Ubuntu / Debian) **tar.gz (RHEL / systems without unzip)**

```
cd ~
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.zip -o h3-cli.zip
unzip h3-cli.zip -d h3-cli
cd h3-cli
bash install.sh
export PATH="$PWD/bin:$PATH"
```

```
cd ~
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.tar.gz -o h3-cli.tar.gz
mkdir -p h3-cli && tar -xzf h3-cli.tar.gz -C h3-cli
cd h3-cli
bash install.sh
export PATH="$PWD/bin:$PATH"
```

Usage

For detailed instructions and usage, refer to the CLI Guide

CONFIGURE A NODEZERO RUNNER

It is strongly recommended to set up a NodeZero Runner on your NodeZero Host. The NodeZero Runner enables deployment of pentests from the WebUI without needing to log into each host to run a pentest. Runners also enable scheduling pentests for continuous pentests against the target network.

NEXT STEP

4 Validate Host

Test connectivity and container runtime to confirm readiness.

2.4.5 Common Setup Issues

Common Setup Issues

This section offers solutions to common installation, configuration, and operational issues encountered with NodeZero. Whether you're dealing with connectivity problems, container runtime permission errors, or disk space limitations, you'll find guidance here to help you quickly troubleshoot and resolve these issues. For additional support, feel free to contact our Support Team.

HOST CANNOT CONNECT TO THE PORTAL

Your NodeZero Host fails to communicate with the Portal, showing errors like `Unable to connect` or `Connection timed out`, or warnings like `Attempted 1 Ping to <resource>: FAILED`.

Verify firewall settings

Ensure that your firewall is not blocking outbound connectivity to the required domains.

Whitelist for domains and ports

Whitelist requirements differ depending on whether you're using the **Preconfigured OVA** or performing a **Manual Installation**.

- **Preconfigured OVA**: Review the list of required domains for the Preconfigured OVA setup in the OVA setup documentation.
- **Manual Installation**: Review the list of required domains for the Manual Installation setup in the Manual Installation documentation.

DNS filtering

Confirm that **DNS filtering** is not redirecting or blocking the whitelisted domains. Even if the domains are whitelisted in the firewall, DNS rules can still block them, preventing successful outbound connectivity.

Test connectivity

US Region

```
ping portal.horizon3ai.com
```

```
curl -I https://portal.horizon3ai.com
```

EU Region

```
ping portal.horizon3ai.eu
```

```
curl -I https://portal.horizon3ai.eu
```

Check network reachability

Ensure that your network allows outbound access to the internet.

- If using VLANs, verify that VLAN tagging is not blocking access to external domains.
- Confirm that your NodeZero Host has a valid route to the required domains.

Incorrect IP configuration

The NodeZero Host may fail to communicate with other devices or the internet due to misconfigured IP settings, such as an incorrect IP address, default gateway, or DNS.

Verify IP settings

Check if the IP address, default gateway, and DNS settings were correctly configured during setup.

- Review the initial configuration values entered for the VM.
- Confirm the assigned IP address using:

```
ip addr
```

Configure a static IP address

By default, the NodeZero Host uses DHCP. To set a static IP, use the built-in Host Utility.

1. Open a terminal and run the `n0` command.
2. Select option `3` to configure a static IP.
3. Follow the prompts to enter:
- **IP Address:** The static IP for the host (e.g., `192.168.1.100`).
- **Subnet Mask:** (e.g. `255.255.255.0`).
- **Gateway:** Your network's gateway IP (e.g., `192.168.1.1`).
- **DNS Nameserver:** Primary and secondary DNS servers (e.g., `8.8.8.8`, `8.8.4.4`).
4. Verify the new IP configuration:

```
ip addr show
```

5. Test connectivity with the new static IP:

```
ping <static-ip>
```

6. If changes don't apply immediately, reboot the host:

```
sudo reboot
```

7. To revert to DHCP, rerun the `n0` utility, select option `3`, and choose the DHCP toggle.

Container runtime permission issues

Proper container runtime configuration and user permissions are essential for NodeZero to function correctly. Many users encounter issues related to Docker or Podman validation, permissions, and access restrictions. These problems typically arise due to incorrect user permissions, missing group memberships, or insufficient `sudo` privileges.

"Failed to validate your container runtime" error

The NodeZero Launch Script detects your container runtime and reports a corresponding error. The text after the runtime name tells you the cause. Select your runtime below for the message you'll see and the most common causes and fixes.

Docker **Podman**

```
[!] FAILED: Failed to validate Docker. ...
```

Permissions to run Docker (`Verify this account has permissions to run docker`)

The user running NodeZero is not in the `docker` group. Add them, then log out and back in (or reboot) for the change to take effect:

```
sudo usermod -aG docker $(whoami)
groups $(whoami)   # confirm "docker" is listed
```

Docker daemon not started (`Verify the docker daemon/service is started`)

```
sudo systemctl start docker
```

Docker not installed (`Verify that docker is installed`)

Install a container runtime — see Container Runtime Installation.

Volume-mount path error (`The path ... is not shared and is not known to docker`)

This typically occurs on Docker Desktop (macOS/Windows), where directories must be explicitly shared with the runtime. Run NodeZero from a directory Docker is allowed to mount. The error output includes a link to the relevant file-sharing documentation for your platform (Docker for Mac, Docker for Windows).

```
[!] FAILED: Failed to validate Podman. ...
```

When using Podman, NodeZero always runs containers with `sudo`.

Permissions / sudoers (`Verify this account has permissions to run podman` or `Verify this account is in the sudoers file`)

Ensure the user running NodeZero has `sudo` privileges, then confirm Podman works under `sudo`:

```
sudo podman run hello-world
```

Podman not installed (`Verify that podman is installed`)

Install a container runtime — see Container Runtime Installation.

Volume-mount path error (`The path ... is not shared and is not known to podman`)

Run NodeZero from a directory Podman is allowed to mount. See the Podman documentation for more information.

"Permission Denied" on job execution

Users may encounter a "Permission Denied" error when the NodeZero Runner attempts to execute jobs. This is the same underlying problem as the permissions case above — the user lacks access to the container runtime.

Apply the permissions fix for your runtime under Failed to validate your container runtime, then restart the Runner so it picks up the new permissions:

```
h3 stop-runner
h3 start-runner {runner-name}
```

CLI, API and Runner issues

This section provides resolutions for frequently reported errors related to the CLI, API authentication, and Runner operations. If you are facing command issues or your Runner is not appearing in the Portal, see below.

CLI & API

Below are common issues related to the command-line interface (CLI) and API authentication.

"h3: Command Not Found" error

When attempting to use `h3-cli`, the system returns an error stating that the command cannot be found.

Add `h3-cli` to `PATH`

The CLI may not be installed correctly, or it might not be included in the system's PATH variable. To fix this:

```
export PATH="$H3_CLI_HOME/bin:$PATH"
echo 'export PATH="$H3_CLI_HOME/bin:$PATH"' >> ~/.bashrc
```

After running these commands, restart your terminal and verify the installation by executing:

```
h3 --version
```

Portal issues

This section helps troubleshoot problems related to the NodeZero Portal, including login issues and data clearing steps.

Google Chrome cache

Users may experience login or loading issues with the Horizon3.ai Portal due to cached data.

Login or load issues with the Portal

The Portal fails to load correctly or users are unable to log in.

Clear Chrome application data

Follow these steps to clear app data in Chrome:

1. Open Chrome DevTools (`Cmd+Opt+I` on Mac / `Ctrl+Shift+I` on Windows/Linux).
2. Navigate to the **Application** tab.
3. Under **Storage**, click **Clear site data**.

NodeZero exiting prematurely

NodeZero can get stopped by Endpoint Detection and Response (EDR) services running on the same host machine. To prevent this from occurring, remove or disable any EDRs on the NodeZero host.

2.5 Validate NodeZero Host

2.5.1 Validate Host

Once the installation is complete, validate that your NodeZero host is ready to run operations by running the NodeZero environment validation script.

VALIDATION SCRIPT

The validation script ensures that your NodeZero host is properly configured and ready for operations by checking key system and network requirements. The script will complete the following:

- Check for minimum requirements (2 CPUs, 8GB RAM, 40GB+ disk).
- Ensure that necessary packages and libraries are installed.
- Confirm compatibility and proper configuration.
- Ensure that the host can resolve external domains.
- Check for blocking configurations or restricted access.

DOWNLOAD VALIDATION SCRIPT

Pick the tab that matches how you are connecting to NodeZero's cloud infrastructure.

US-based + Gateway | **EU-based + Gateway** | **US-based** | **EU-based** | **Federal**

Environment check script for connectivity and host settings for users connecting with the Gateway and US website.

Download checkenv_us.sh | **SHA256 checkenv.sh**

Environment check script for connectivity and host settings for users connecting with the Gateway and EU website.

Download checkenv_ce_eu.sh | **SHA256 checkenv.sh**

Environment check script for connectivity and host settings for users connecting to the US website.

Download checkenv_us.sh | **SHA256 checkenv.sh**

Environment check script for connectivity and host settings for users connecting to the EU website.

Download checkenv_eu.sh | **SHA256 checkenv.sh**

Environment check script for connectivity and host settings for users connecting to the Federal infrastructure

Download checkenv_federal.sh | **SHA256 checkenv.sh**

Warning

Always verify that the files you download come from Horizon3.

Plug the file name of your downloaded script version into the run command directly below.

RUN VALIDATION SCRIPT

To execute the validation script on the OVA host, use a command that matches the script file name you've downloaded. For example, in a basic US environment, with a bash shell, you'd run the following command:

```
curl https://downloads.horizon3ai.com/utilities/checkenv_us.sh | bash
```

In a EU environment with the NodeZero Gateway, you'd run this command:

```
curl https://downloads.horizon3ai.com/utilities/checkenv_ce_eu.sh | bash
```

With any script version, the output should look similar to the following:

```
# curl https://downloads.horizon3ai.com/utilities/checkenv_us.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 18646  100 18646    0     0  61134      0 --:--:-- --:--:-- --:--:-- 61134

[#] Conducting pre-checks to validate the environment is NodeZero ready:

[#] Checking Proxy settings and configurations:
[+] Found 0 of 6 proxy settings in /etc/environment
[+] Found 0 of 6 proxy settings in env
[+] PASSED: Proxy is NOT configured

[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[+] PASSED: Docker is installed and functioning properly.

[#] Checking Docker permissions to volume mount files from /home/nodezero directory:
[+] PASSED: Docker permissions are correct for the /home/nodezero directory location.

[#] Checking Operating System:
[+] PASSED: Linux is a supported Operating System.

[#] Gathering environmental variables to conduct further checks:
[+] PASSED: All environmental variables set and proceeding with next checks.

[#] Checking host time against current UTC time:
[+] PASSED: System time is within 5 minutes of UTC time.

[#] Checking HDD space requirements (30GB Recommended, 20GB Required):
[+] PASSED: There is enough space for the NodeZero container: 79GB

[#] Checking 8GB RAM requirement:
[+] PASSED: This system meets the recommended minimum RAM to support NodeZero.

[#] Checking compute resource requirements:
[+] PASSED: This system has 2 CPUs which meets the minimum logical CPU requirements to run NodeZero.

[#] Pre-check validation completed successfully.
```

AFTER SUCCESSFUL VALIDATION

After successfully running the validation script, your NodeZero host is ready for operations.

Next Step

Run Your First Pentest

Run a test, let NodeZero uncover vulnerabilities, and review the findings.

2.6 Run an Internal Pentest

2.6.1 Run an Internal Pentest

A NodeZero **Internal Penetration Test** is a simulated attack on your organization's internal network designed to identify weaknesses that could be exploited by attackers within your own infrastructure. This test mimics the behavior of a malicious actor attempting to exploit deficiencies in your internal systems, applications, and devices. By performing this test, you can gain insights into potential security gaps before they are discovered and exploited by real-world threats.

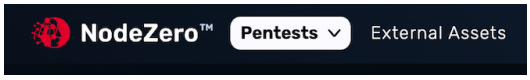
This guide will help you understand how to configure and run an internal penetration test using NodeZero. It will walk you through the steps to set up the test, define the scope, customize the parameters, and analyze the results to improve your network's security posture.

Instructions

Follow these steps to run an **internal pentest** within your network.

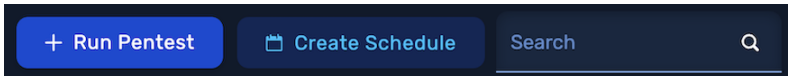
ACCESS NODEZERO PORTAL

After setting up your NodeZero Host, log in to the NodeZero Portal, and click the **Pentests** button. Select **Pentests** from the drop-down menu.

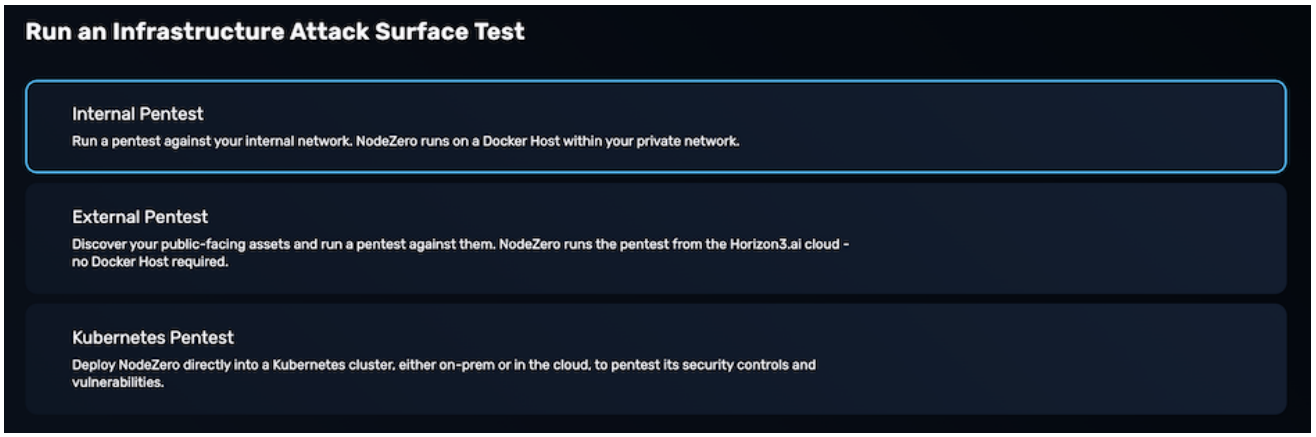


SELECT INTERNAL PENTEST

Choose the option to **+ Run Pentest**.



Select **Infrastructure Attack Test**. Then, select **Internal Pentest**.

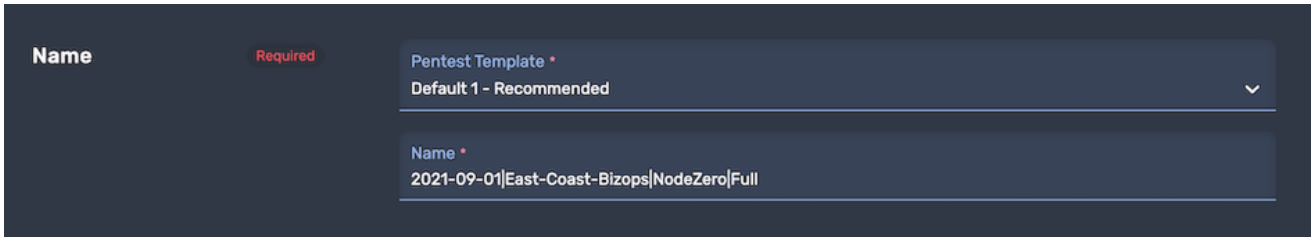


CONFIGURE INTERNAL PENTEST

Name the Internal Pentest

Assign a meaningful name to the Internal Pentest and select an appropriate pentest template.

> **Naming Conventions**
>
> Establish a consistent naming convention to easily identify pentests in your pentest list.
>
> **Example naming convention:** `[date]|[location/network]|[NodeZero source]|[scope]`
>
> **Example following this convention:** `2021-09-01|East-Coast-Bizops|NodeZero|Full` This name indicates that the NodeZero host was placed in the East Coast BizOps network, and that the scope covered the entire enterprise.



Select scope

The pentest scope defines the IPs and/or subnets where you want to run the pentest. A broader scope yields more comprehensive results. Unlike traditional vulnerability scanners, NodeZero assesses your environment holistically, using discovered data and context to identify and exploit weaknesses, vulnerabilities, misconfigurations, and poor cybersecurity practices.

- Configure the following scope options:

Scope
Required
The test's scope defines the list of subnets NodeZero will attempt to scan. Subnets are specified using IP or CIDR notation.
Intelligent Scope
Add Full Private IP Space
Auto-expand Scope
Include *
Enter a valid IP: 10.0.0.0/8, 172.16.0.0/12, etc.
Please provide an IP range or use the 'Intelligent Scope' option.
Exclude
Enter a valid IP: 10.0.0.0/8, 172.16.0.0/12, etc.

- **Intelligent scope** This feature utilizes the IP of the NodeZero host from which the test is deployed, enumerating and testing against the /16 subnet it belongs to. NodeZero then expands into adjacent /23 subnets, continuously identifying and testing nearby assets. The cycle repeats until no additional devices are detected. The scope of the test depends on the level of privilege and access NodeZero has, either through capture or granted permissions.

> **Tip**
>
> Great for scenario-based testing - assess what an attacker can view and do from the perspective of your NodeZero host.

- **Add full private IP space:** Adds the following IP address range to the scope (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16).
- Selecting this option allows NodeZero to enumerate your local network looking for any RFC 1918 address, which should not be publicly routable.
- RFC 1918 is a standard published by the Internet Engineering Task Force (IETF) that defines private IP address spaces for use in internal networks. These IP addresses are not routable over the public internet, meaning they are used for local communication within private networks.

> **Tip**
>
> Ideal for comprehensive coverage – understanding what's in your infrastructure and prioritizing from an attacker's viewpoint.

- **Auto-expand scope** If selected, NodeZero will automatically include any additional devices it can discover during the test. While NodeZero can be scoped to specific IP addresses, it's recommended to broaden the scope for better coverage. This approach helps identify potential vulnerabilities across a wider range of devices, improving the overall effectiveness of the test.
- **Included scope** The **include** section in an internal penetration test lets you specify specific areas or systems within your network that you want to focus the test on. This is useful when you want to target particular assets, such as high-value servers or applications, for in-depth testing. By using the Include section, you ensure that only the designated parts of your infrastructure are tested, while other areas are not affected.
- Inclusions require **Classless Inter-Domain Routing (CIDR)** notation.

  Understanding CIDR Notation CIDR Calculator Tool

> **Tip**
>
> **Example:** If your environment uses `192.168.0.1` with a subnet mask of `255.255.255.0`, add `192.168.0.0/24` to the **Include** section.
>
> **Example:** For segmented environments, use comma-separated CIDR notation to cover multiple subnets.
> `192.168.0.0/16,172.16.10.0/24,10.0.0.0/8`

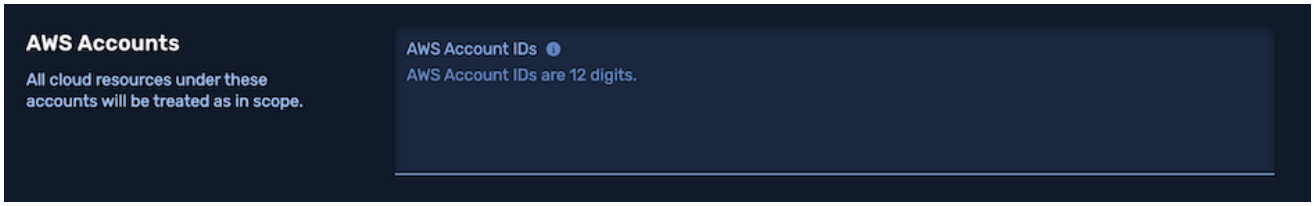
  In more complex environments, set the scope to include as many subnets as possible. Consult your Network Administrator for a list of subnets in CIDR notation.
- **Excluded scope** The **exclude** section in an internal penetration test allows you to specify areas of your network or systems that should be excluded from testing. This feature is useful when you want to avoid testing sensitive systems, applications, or environments that should not be tested for security reasons, such as production environments or systems with critical data.

  Use this section to specify IPs or subnets that NodeZero should avoid scanning or exploiting. Excluded IPs may still appear in the **Out of Scope** list in the pentest results if they are discovered during the test. Exclusions require CIDR notation.

  Once you're satisfied with the scope, proceed to the next step.

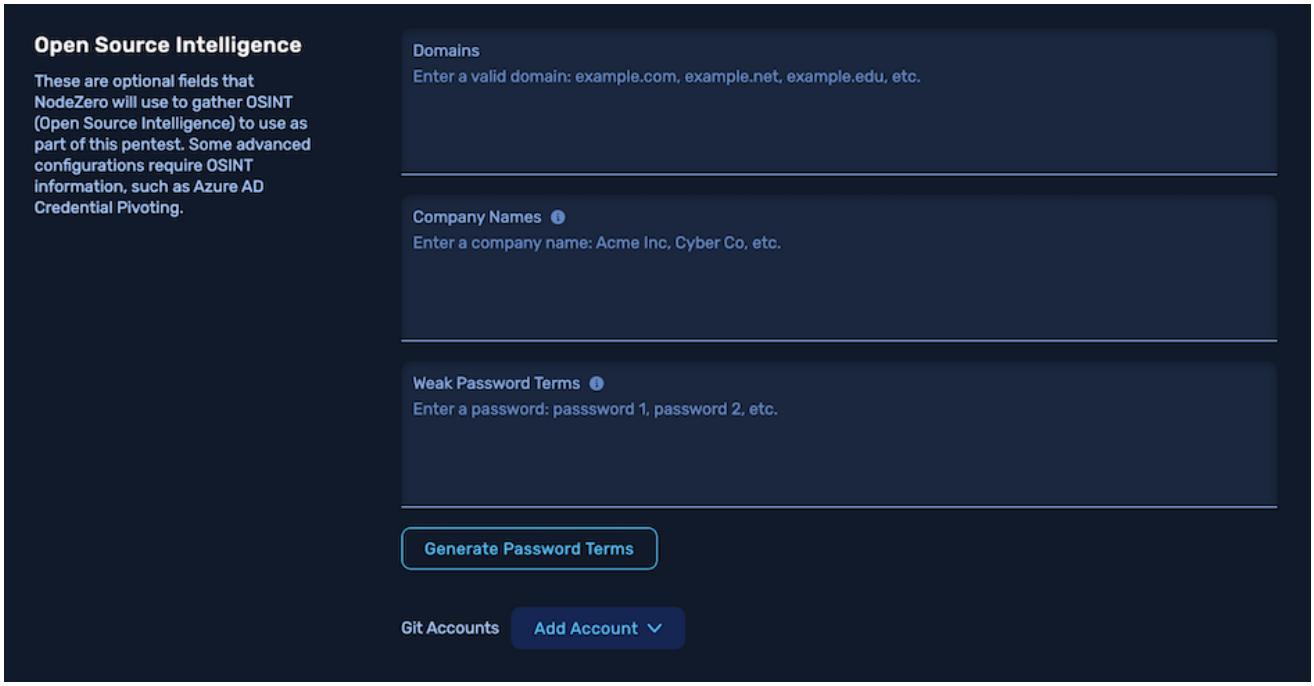
Amazon Web Service (AWS) Accounts

Optionally, add AWS accounts here. All cloud resources under these accounts will be treated as in scope.



Add Open-Source Intelligence

Optionally add details such as domains, company names, weak password terms, or Git account information.



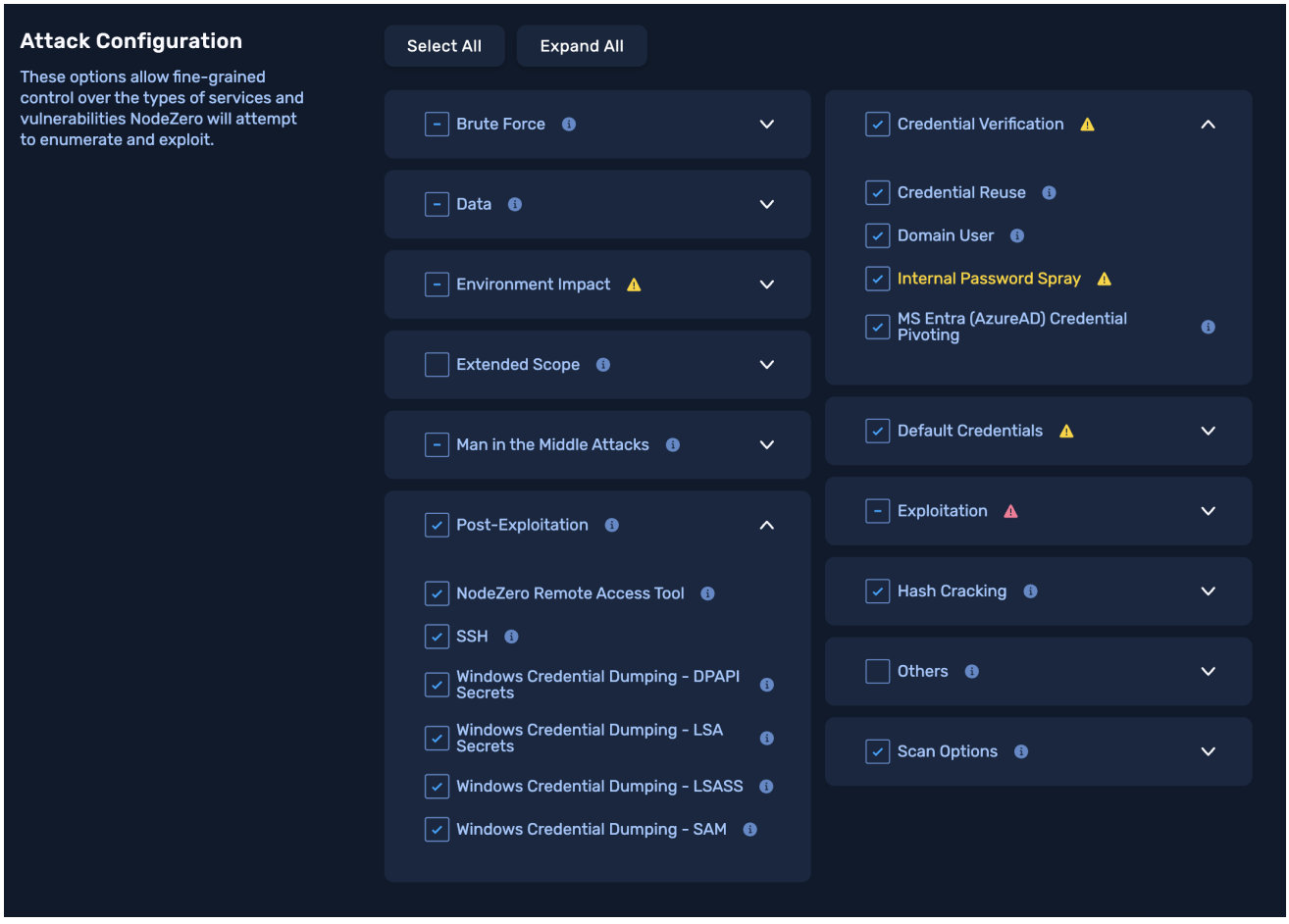
Tripwires (Optional)

> Not required for Quickstart.

If your organization has purchased Tripwires, enable this option to allow NodeZero to deploy tokens in exploitable assets during the pentest. A maximum of 100 tripwires will be deployed per test.

Attack configuration

Choose the specific services and vulnerabilities that NodeZero should enumerate and exploit during the pentest.



Duration

Specify a minimum or maximum duration to provide more time for certain attacks to execute effectively.

Runner (Optional)

> Not required for Quickstart.

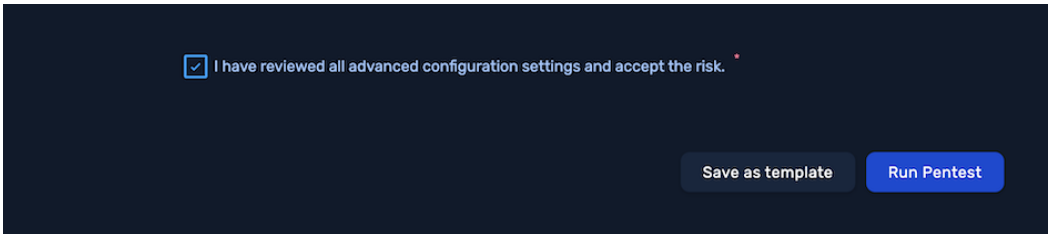
Choose a NodeZero Runner to automatically deploy NodeZero on your Docker host.



Review Config

Once you've finalized your pentest selections, confirm your review of all advanced configuration settings by checking the designated box.

Finally, click **Run Pentest** to initiate the internal pentest.



DEPLOY NODEZERO

As the pentest provisions, a one-time-use `curl script` is prepared for deployment on your NodeZero host.

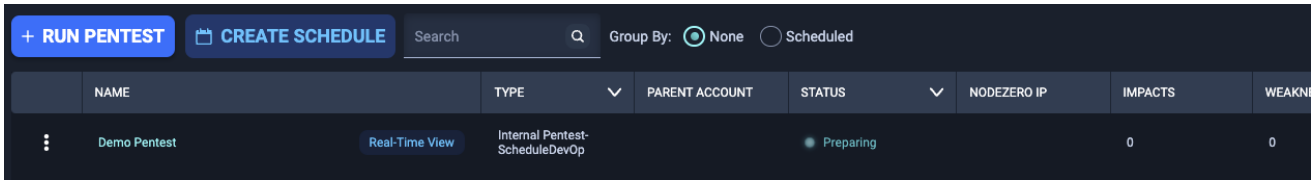
Deployment instructions:

1. SSH into the NodeZero host
2. Copy the launch script from the Portal
3. Paste the script into the shell of your NodeZero host

   The script will:

- Validate the Docker installation.
- Download the latest NodeZero Docker image.
- Initiate the pentest.

  In the Portal: - The pentest status will update from **Ready** to **Running**. - An email notification will be sent by NodeZero once the internal pentest is completed.



MONITOR IN REAL-TIME

Click **Real-Time View** to track the progress of the pentest, watch identified vulnerabilities, and gain instant feedback on any issues that arise.

REVIEW TEST RESULTS

After the pentest is complete, review the findings to identify vulnerabilities, misconfigurations, and potential attack vectors.

TAKE ACTION

Prioritize and address the vulnerabilities identified during the pentest. NodeZero provides recommendations and remediation steps to help you secure your environment.

Additional Options

Gain deeper insights by running the pentest from an authenticated perspective.

INJECT CREDENTIALS

For a more comprehensive test, you can Inject Credentials to simulate an attacker with authenticated access. This gives you deeper insights into the potential impact of compromised credentials.

Congratulations!

You've successfully run an internal pentest with NodeZero. You're now ready to assess and improve the security of your network environments. Continue exploring NodeZero for further guidance and advanced features to stay ahead of potential threats.

Next Step

Explore the NodeZero Portal

Explore NodeZero Portal capabilities, including penetration testing, compliance, and advanced features.

3. Portal

3.1 NodeZero Portal

The NodeZero **Portal** is your command center for autonomous penetration testing, empowering security teams, IT administrators, and DevOps professionals to enhance their organization's security posture. This intuitive web UI enables you to launch and manage a variety of **tests and assessments**, including internal pentests, external evaluations, and cloud-focused scans. The Portal integrates with key components like the **NodeZero Host** for internal testing, leverages **NodeZero Runners** for scheduling automation, and supports strategic deployment approaches—delivering real-time insights, actionable results, and comprehensive reporting to identify vulnerabilities and track remediation progress.

3.1.1 Essential Functions

- Launch and manage **tests and assessments** (internal, external, OSINT-focused, and more)
- Access comprehensive reports to plan and track remediation efforts
- Monitor real-time progress of all pentests in a centralized interface
- Manage user permissions, integrations, and system configurations
- Integrate with NodeZero components for a complete testing solution

Tests and Assessments

Explore available test types to assess your network's security posture.

Portal Features

Discover additional capabilities and tools within the NodeZero Portal.

Portal Settings

Configure user permissions, integrations, and system settings.

3.1.2 Critical Components

The NodeZero Portal relies on critial components to conduct tests and assessments.

- The **NodeZero Host** is essential for internal pentests, enabling the Portal to assess your network from within.
- The Host's **Runner** automates these tests
- And a well-planned **Deployment Strategy** ensures the Host is placed and configured to effectively identify vulnerabilities.

NodeZero Host

Deploy a NodeZero Host for internal pentests which simulate attacks.

NodeZero Runner Optional

Enable NodeZero Host’s Runner to automate pentests from the Portal.

Deployment Strategy

Plan your Host’s placement and intent to identify vulnerabilities.

3.2 Tests & Assessments

3.2.1 Tests and Assessments

Explore the penetration test types available to assess and strengthen your network’s security posture. Below is one pragmatic sequence of NodeZero tests and related techniques – scroll down for details about supported tests by category.



Set up Runner

Automate pentesting – schedule and manage recurring tests remotely from the Portal UI



Segmentation test

Scan your internal network to identify potential targets for future attacks



Internal test

Simulate an attack on your network from within your own infrastructure



Optional post-test follow-ups

Choose either, both, or neither

- 1-Click Verify remediations
- Assess EDR Effectiveness



External test

Discover and test vulnerable external-facing assets



Templatize tests

Save pentest configurations to easily repeat and schedule tests



Schedule tests

Automate repeated testing from the NodeZero Portal or CLI



Cloud-based tests

Choose any or all

- AWS
- Azure Entra ID
- AD Password Audit

Other tests to consider

Choose any or all

- Insider Threat test
- Phishing test
- Kubernetes test

Infrastructure Attack Surface

Infrastructure Attack Surface tests are NodeZero's autonomous network penetration tests. These all-inclusive tests are deployable from various perspectives. They identify attack paths across on-premises, hybrid-cloud, perimeter, and cloud networks.

Internal Pentest

A pentest run against an internal network. NodeZero is deployed on a Docker host within your private network.

External Attacks

NodeZero runs from the Horizon3.ai cloud and tests your public-facing or external assets.

Kubernetes Pentest

Deploy NodeZero directly into a Kubernetes cluster, either on-prem or in the cloud, to pentest its security controls and vulnerabilities.

Identity Attack Surface

NodeZero's Identity Attack Surface tests find and exploit real-world attack-paths and vulnerabilities in IAM controls and platforms.

AWS Pentest

NodeZero uses a privileged role in your AWS account to discover vulnerabilities and misconfigurations.

Azure Entra ID Pentest

NodeZero runs on a Docker host within your private network to discover and assess attack paths within your hybrid Entra ID environment.

Active Directory Password Audit

Audit your users' Active Directory Passwords. NodeZero will reveal weak, breached, and reused passwords.

Operational Scenario Testing

NodeZero's Operational Scenario tests are used to validate your organization's security readiness by using real-world attack techniques to tests your environment's resilience to operational scenarios.

Segmentation Testing

Discover your internal attack surface. NodeZero enumerates IPS, ports, services and applications within your network.

Phishing Pentest

Use NodeZero to measure the impact of phishing campaigns by injecting the phished credentials into an internal pentest.

Insider Threat Testing

Use NodeZero to validate your organization's security posture against insider threats.

Attack Configurations

NodeZero offers flexible Attack Configurations to customize how penetration tests are conducted. These configurations allow you to choose the specific attack techniques and modules based on your target environment. By adjusting these settings, you can simulate various types of attacks to assess the security posture of your systems and networks.

For more detailed information on choosing attack configuration options, see our Attack Configurations topic.

Back to Portal home

NodeZero Runner

3.2.2 Internal Pentest

Run an Internal Pentest

This guide will help you understand how to configure and run an internal penetration test using NodeZero. It will walk you through the steps to set up the test, define the scope, customize the parameters, and analyze the results to improve your network's security posture.

WHAT IS AN INTERNAL PENTEST?

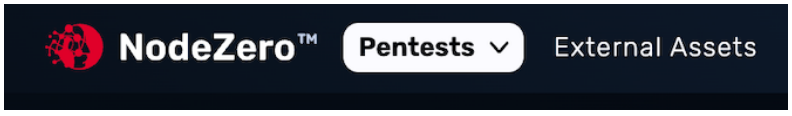
A NodeZero **Internal Penetration Test** is a simulated attack on your organization's internal network designed to identify vulnerabilities that could be exploited by attackers within your own infrastructure. This test mimics the behavior of a malicious actor attempting to exploit weaknesses in your internal systems, applications, and devices. By performing this test, you can gain insights into potential security gaps before they are discovered and exploited by real-world threats.

INSTRUCTIONS

Follow these steps to run an **internal pentest** within your network.

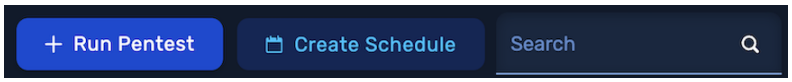
Access NodeZero Portal

After setting up your NodeZero Host, log in to the NodeZero Portal and click the **Pentests** button. Select **Pentests** from the dropdown menu.

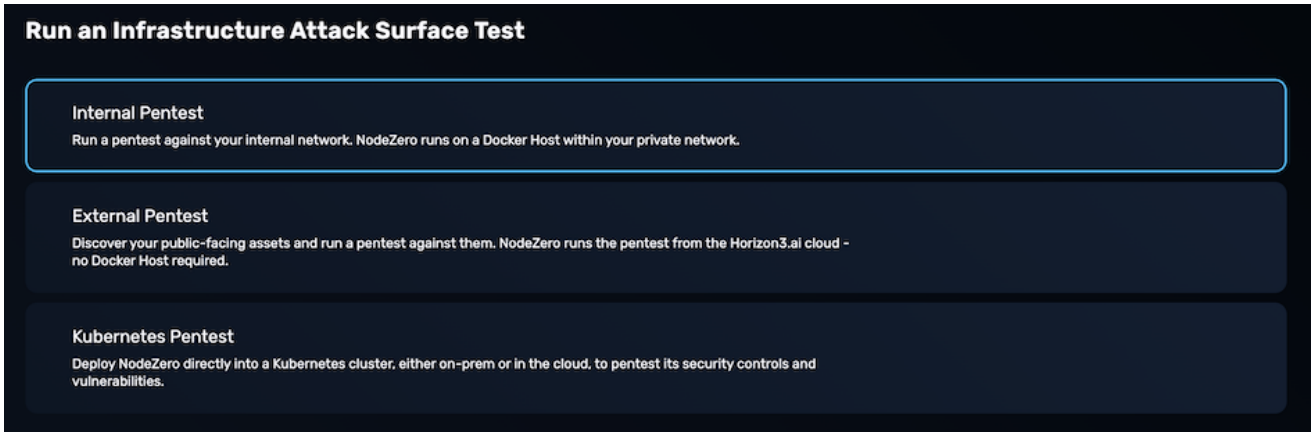


Select Internal Pentest

Choose the option to **+ Run Pentest**.



Select **Infrastructure Attack Test**. Then, select **Internal Pentest**.



Configure internal pentest

Name the internal pentest

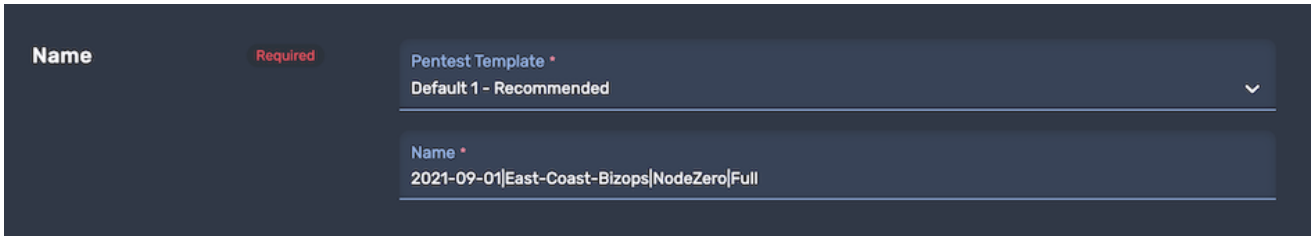
Assign a meaningful name to the Internal Pentest and select an appropriate pentest template.

Naming Conventions

Establish a consistent naming convention to easily identify pentests in your pentest list.

Example Naming Convention: `[date]|[location/network]|[NodeZero source]|[scope]`

Example: `2021-09-01|East-Coast-Bizops|NodeZero|Full` This name indicates that the NodeZero host was placed in the East Coast BizOps network, and the scope covered the entire enterprise.



Select scope

The pentest scope defines the IPs and/or subnets where you want to run the pentest. A broader scope yields more comprehensive results. Unlike traditional vulnerability scanners, NodeZero assesses your environment holistically, using discovered data and context to identify and exploit vulnerabilities, misconfigurations, and poor cybersecurity practices.

- Configure the following scope options:

Scope
Required
The test's scope defines the list of subnets NodeZero will attempt to scan. Subnets are specified using IP or CIDR notation.
Intelligent Scope
Add Full Private IP Space
Auto-expand Scope
Include *
Enter a valid IP: 10.0.0.0/8, 172.16.0.0/12, etc.
Please provide an IP range or use the 'Intelligent Scope' option.
Exclude
Enter a valid IP: 10.0.0.0/8, 172.16.0.0/12, etc.

- **Intelligent scope** This feature utilizes the IP of the NodeZero host from which the test is deployed, enumerating and testing against the /16 subnet it belongs to. NodeZero then expands into adjacent /23 subnets, continuously identifying and testing nearby assets. The cycle repeats until no additional devices are detected. The scope of the test depends on the level of privilege and access NodeZero has, either through capture or granted permissions.

> **Tip**
>
> Great for scenario-based testing - assess what an attacker can view and do from the perspective of your NodeZero host.

- **Add full private IP space:** Adds the following IP address range to the scope (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16).
- Selecting this option allows NodeZero to enumerate your local network looking for any RFC 1918 address, which should not be publicly routable.
- RFC 1918 is a standard published by the Internet Engineering Task Force (IETF) that defines private IP address spaces for use in internal networks. These IP addresses are not routable over the public internet, meaning they are used for local communication within private networks.

> **Tip**
>
> Ideal for comprehensive coverage – understanding what's in your infrastructure and prioritizing from an attacker's viewpoint.

- **Auto-expand scope** If selected, NodeZero will automatically include any additional devices it can discover during the test. While NodeZero can be scoped to specific IP addresses, it's recommended to broaden the scope for better coverage. This approach helps identify potential vulnerabilities across a wider range of devices, improving the overall effectiveness of the test.

> **Note**
>
> NodeZero will only auto-expand within the private (RFC 1918) IP space. Additionally, auto-expand will only work if at least one private IP address is present in Included Scope.

- **Included scope** The **include** section in an internal penetration test lets you specify specific areas or systems within your network that you want to focus the test on. This is useful when you want to target particular assets, such as high-value servers or applications, for in-depth testing. By using the Include section, you ensure that only the designated parts of your infrastructure are tested, while other areas are not affected.
- Inclusions require **Classless Inter-Domain Routing (CIDR)** notation.

  Understanding CIDR Notation CIDR Calculator Tool

> **Tip**
>
> **Example:** If your environment uses `192.168.0.1` with a subnet mask of `255.255.255.0`, add `192.168.0.0/24` to the **Include** section.
>
> **Example**
> For segmented environments, use comma-separated CIDR notation to cover multiple subnets.
> `192.168.0.0/16,172.16.10.0/24,10.0.0.0/8`

  In more complex environments, set the scope to include as many subnets as possible. Consult your Network Administrator for a list of subnets in CIDR notation.
- **Excluded scope** The **exclude** section in an internal penetration test allows you to specify areas of your network or systems that should be excluded from testing. This feature is useful when you want to avoid testing sensitive

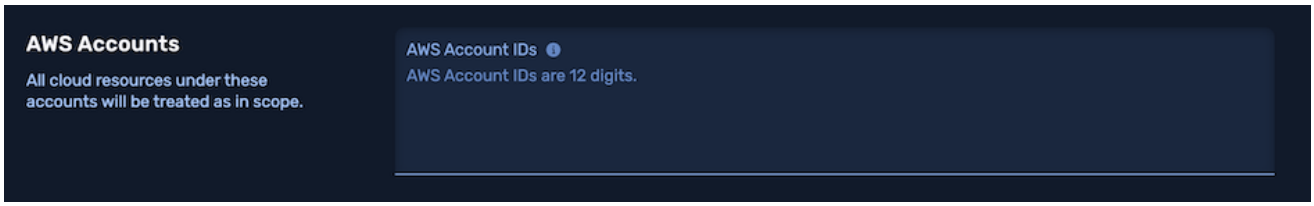
systems, applications, or environments that should not be tested for security reasons, such as production environments or systems with critical data.

Use this section to specify IPs or subnets that NodeZero should avoid scanning or exploiting. Excluded IPs may still appear in the **Out of Scope** list in the pentest results if they are discovered during the test. Exclusions require CIDR notation.

Once you're satisfied with the scope, proceed to the next step.

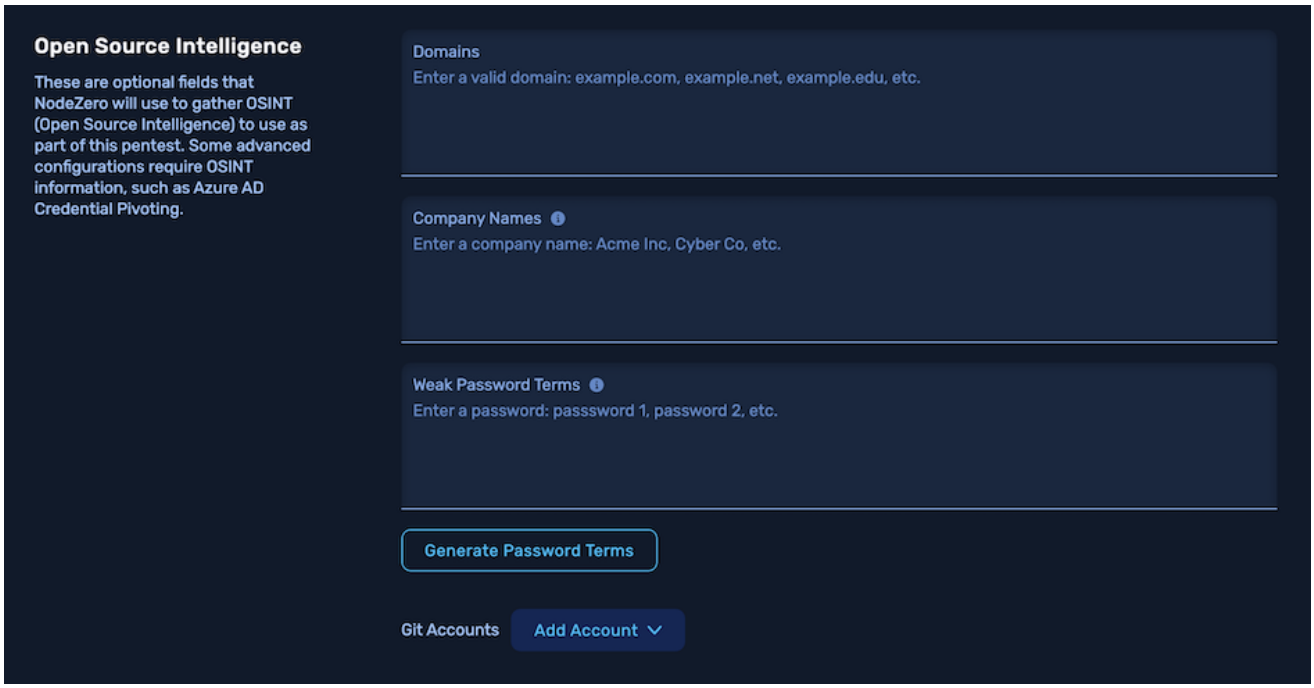
Amazon Web Service (AWS) Accounts

Optionally, add AWS accounts here. All cloud resources under these accounts will be treated as in scope.



Add open-source intelligence

Optionally add details such as domains, company names, weak password terms, or Git account information.



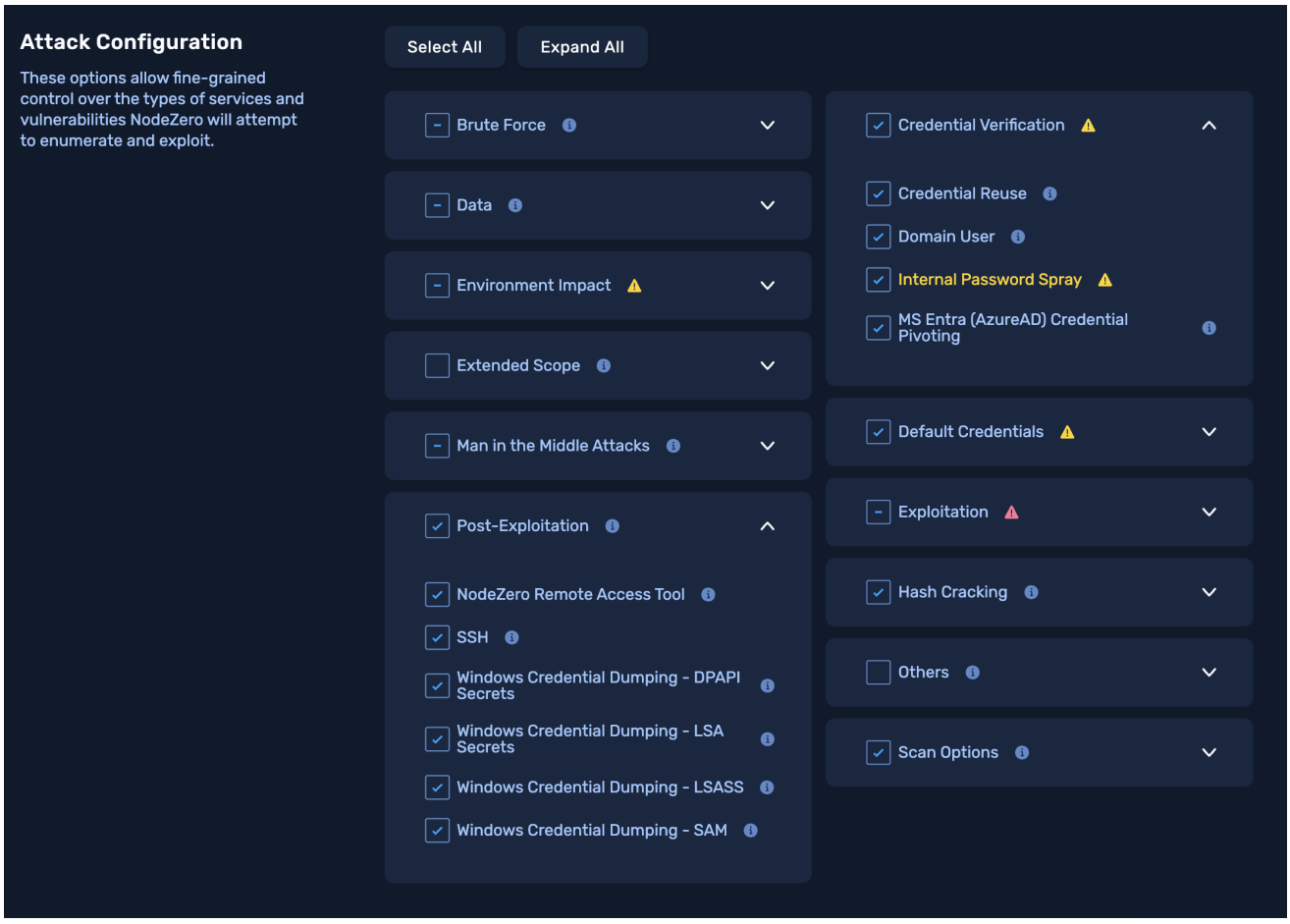
Tripwires (Optional)

> Not required for Quickstart.

If your organization has purchased Tripwires, enable this option to allow NodeZero to deploy tokens in exploitable assets during the pentest. A maximum of 100 tripwires will be deployed per test.

Attack configuration

Choose the specific services and vulnerabilities that NodeZero should enumerate and exploit during the pentest. For more detailed information on choosing attack configuration options, visit the attack configuration options page.



Duration

Specify a minimum or maximum duration to provide more time for certain attacks to execute effectively.

Runner (Optional)

> Not required for Quickstart.

Choose a NodeZero Runner to automatically deploy NodeZero on your Docker host.

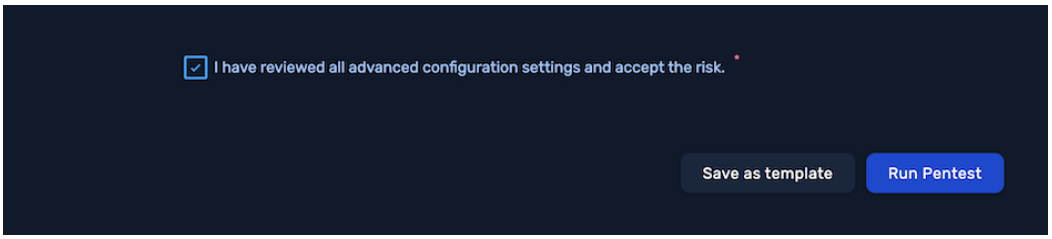


With a Runner assigned, you'll see an option to skip the queue and execute this test immediately.

Review config

Once you've finalized your pentest selections, confirm your review of all advanced configuration settings by checking the designated box.

Finally, click **Run Pentest** to initiate the internal pentest.



Deploy NodeZero

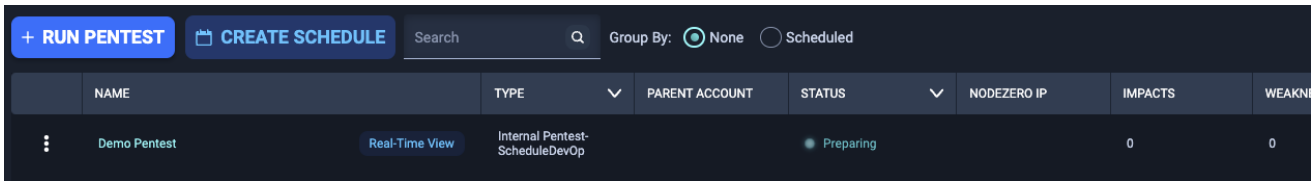
As the pentest provisions, a one-time-use `curl script` is prepared for deployment on your NodeZero host.

Deployment instructions:

1. Copy the launch script
2. Execute the script in the shell of your NodeZero host

   The script will: - Validate the Docker installation. - Download the latest NodeZero Docker image. - Initiate the pentest.

   In the Portal: - The pentest status will update from **Ready** to **Running**. - An email notification will be sent by NodeZero once the internal pentest is completed.



Monitor in Real-Time

Click Real-Time View to track the progress of the pentest, watch identified vulnerabilities, and gain instant feedback on any issues that arise.

Review Test Results

After the pentest is complete, review the findings to identify vulnerabilities, misconfigurations, and potential attack vectors.

Take Action

Prioritize and address the vulnerabilities identified during the pentest. NodeZero provides recommendations and remediation steps to help you secure your environment.

ADDITIONAL OPTIONS

Inject Credentials

Gain deeper insights by running the pentest from an authenticated perspective.

For a more comprehensive test, you can Inject Credentials to simulate an attacker with authenticated access. This gives you deeper insights into the potential impact of compromised credentials.

CONGRATULATIONS!

You've successfully run an internal pentest with NodeZero. You're now ready to assess and improve the security of your network environments. Continue exploring NodeZero for further guidance and advanced features to stay ahead of potential threats.

Go to Portal home

Go to Test Types

3.2.3 External Pentest

External Pentest

NodeZero External Pentest is an easy way to gain an additional perspective on your environment. Due to the nature of an external pentest, you must first authorize the discovered assets.

Before getting started, navigate to the NodeZero Portal and log in with your credentials.

HOW TO RUN AN EXTERNAL PENTEST

To run an external pentest, first create a Scope (formerly called Asset Group), and run Asset Discovery to enumerate external facing assets. When the Asset Discovery completes, you can authorize assets for pentesting and start an external pentest. For specific steps, follow these sections, in order:

1. Create a Scope (Asset Group)
2. Run Asset Discovery
3. Authorize Assets
4. Run an External Pentest

Create a Scope (Asset Group)

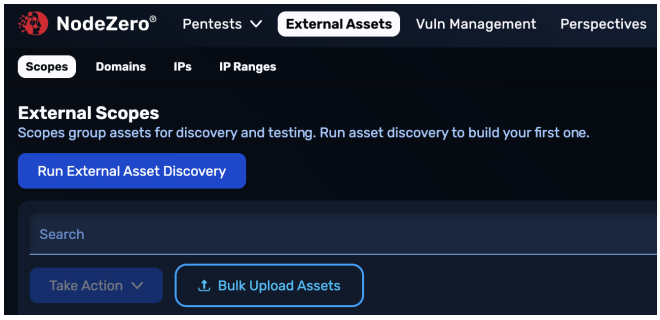
NAVIGATE TO EXTERNAL ASSETS

To run an external pentest, first create a Scope (called an Asset Group in earlier NodeZero Versions). This will define the scope for asset discovery and the external pentest.

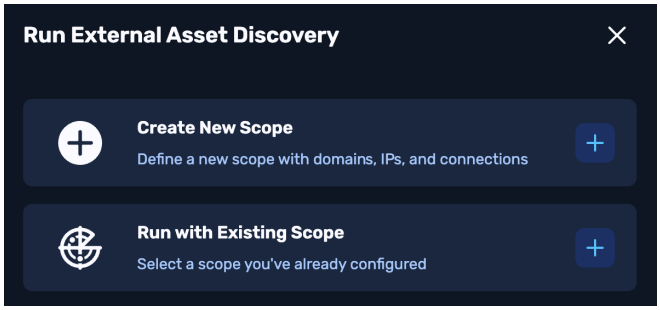
From the NodeZero Portal's top navigation, select **External Assets**.

CONFIGURE THE SCOPE

Unless you want to define your Scope by bulk-uploading assets, kick off UI-guided configuration of your Scope by clicking the **Run External Asset Discovery** button.



From the resulting modal shown below, click the **+** button to the right of **Create New Scope**.



In the resulting **Run External Asset Discovery** modal, use the following steps to define a Scope.

Set Scope's Name and Inputs

Name the Scope and define its inputs (your organization's assets). You can specify registered domains and associated IP addresses to include, and can also specify IPs and subnets to exclude.

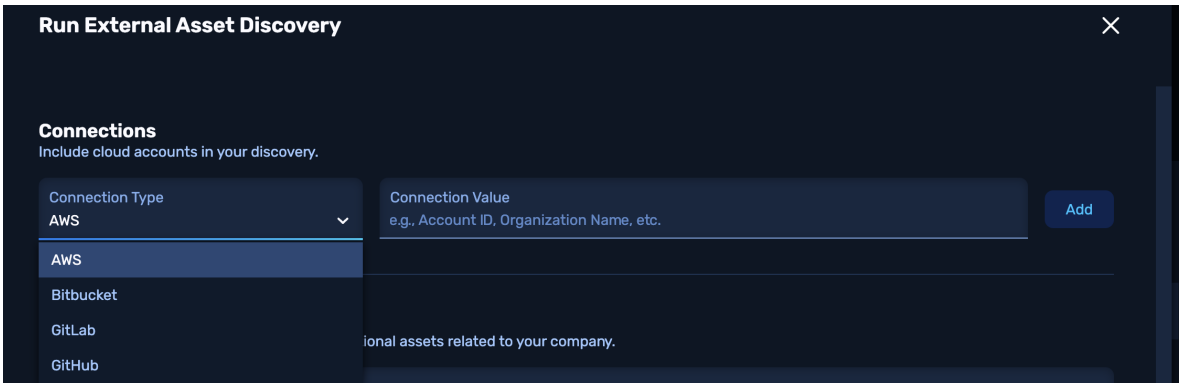
IPs must be public, must be formatted using IP/CIDR notation, and must be comma-separated in each list. Please specify network segments of `/16` or smaller (e.g., `/18`, `/23`, `/24`). See also Add IP Ranges below.

Optionally, toggle on **Auto-Authorize Discovered Assets**.

Add AWS and Git Connections (Optional)

In the **Connections** section (shown below), you have the option to specify your organization's accounts on AWS, GitHub, GitLab, and/or Bitbucket. Listing these related accounts enables NodeZero to confirm ownership of these accounts, to perform a more thorough enumeration of assets.

Use the left **Connection Type** drop-down to select a vendor. Use the right **Connection Value** field to specify your corresponding account ID (e.g., a 12-digit AWS ID) or organization name. Click **Add** to insert each pair into the Scope's configuration.



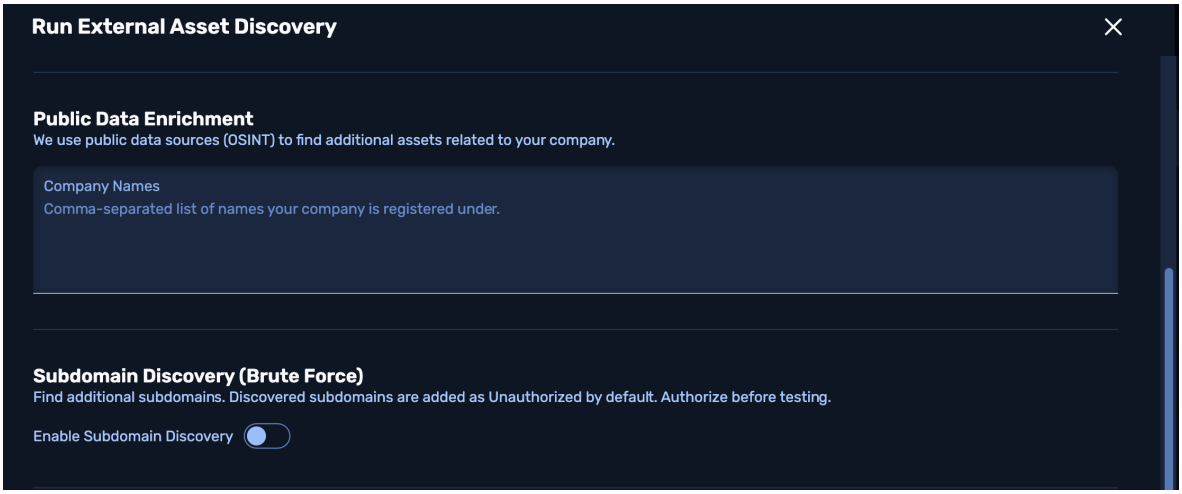
Add OSINT Data

In the **Public Data Enrichment** section, you can add multiple names under which your organization is registered. Using these entries, NodeZero will apply Open Source Intelligence (OSINT) gathering tools and techniques to harvest additional related assets.

You can also specify IPs and subnets to exclude. As in Scope Inputs, both lists should be comma-separated.

Add Subdomain Discovery (Optional)

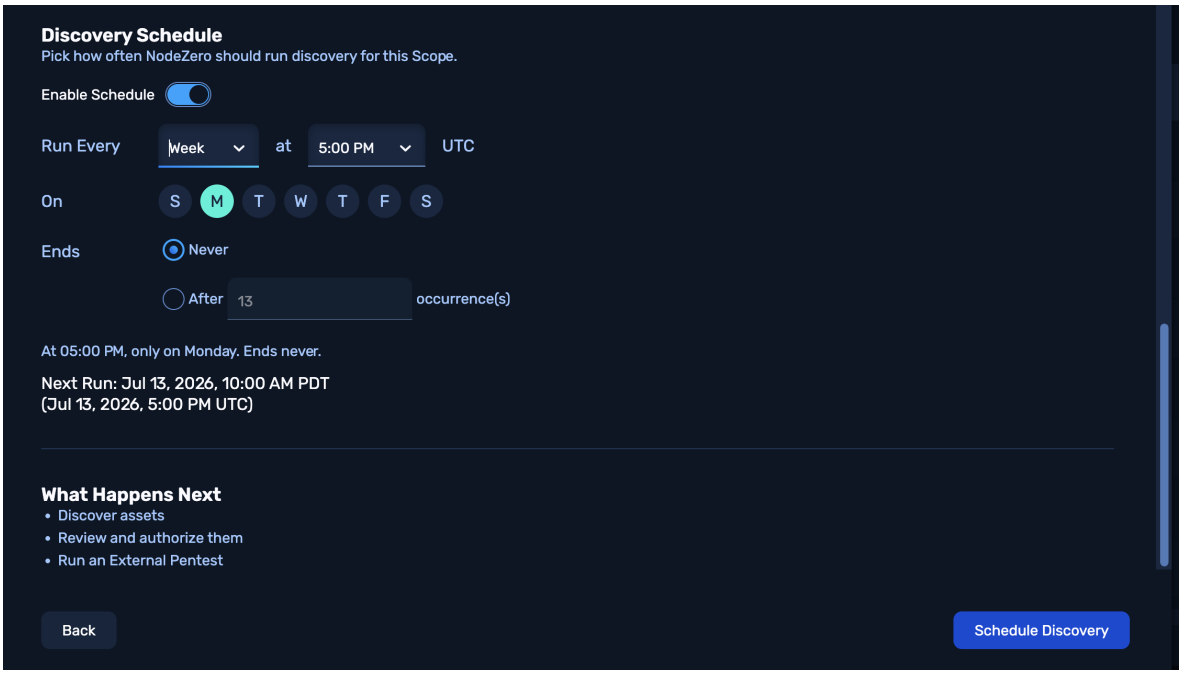
You can enable NodeZero's brute-force discovery of well-known subdomains that might not surface through OSINT discovery. You'll need to explicitly authorize these discovered subdomains for testing.



Enable Schedule (Optional)

You can use the **Discovery Schedule** controls, shown below, to set a cron job that defines repeated discovery.

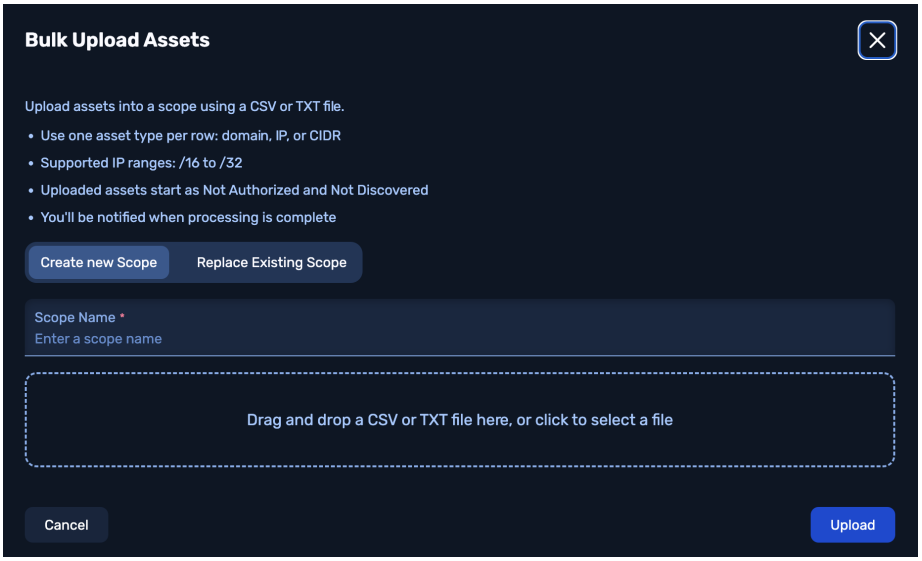
Once satisfied with all of the configuration options above, click **Start Discovery**.



BULK UPLOAD ASSETS

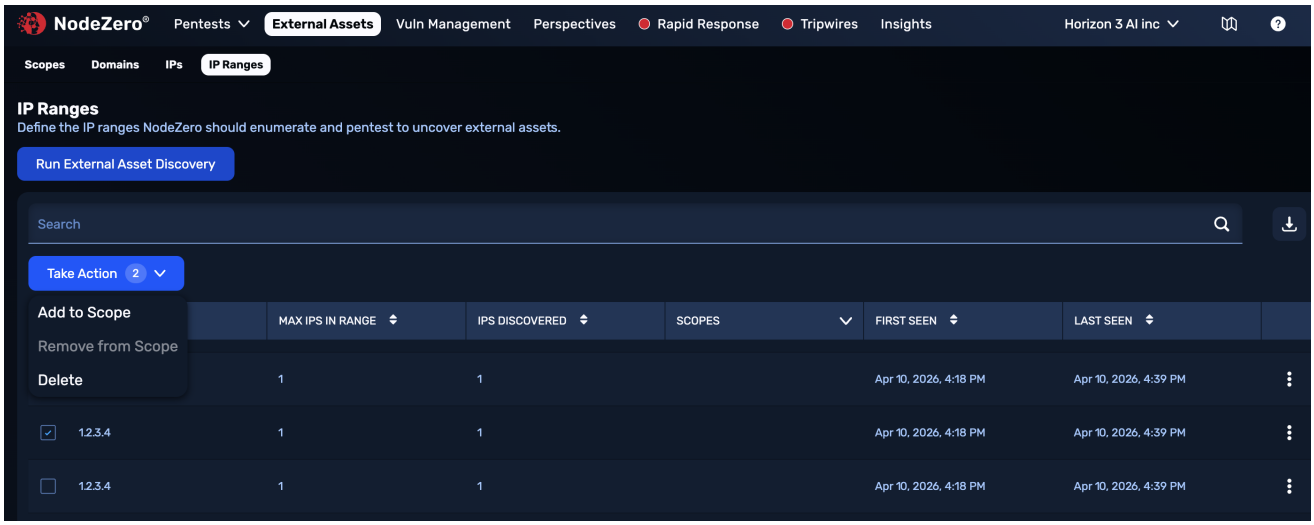
As an alternative to the guided configuration outlined above, you can define (or replace) a Scope by preparing and uploading a comma-separated CSV or text file that lists your assets.

Click **Bulk Upload Assets** to initiate this, and follow the resulting modal's instructions for formatting your file.



ADD IP RANGES

You can use the **IP Ranges** tab to define groups of IPs to add to a Scope. Use the check boxes at left to select ranges, then click the **Take Action** menu's **Add to Scope** option. Next, you'll need to **Run External Asset Discovery** again.



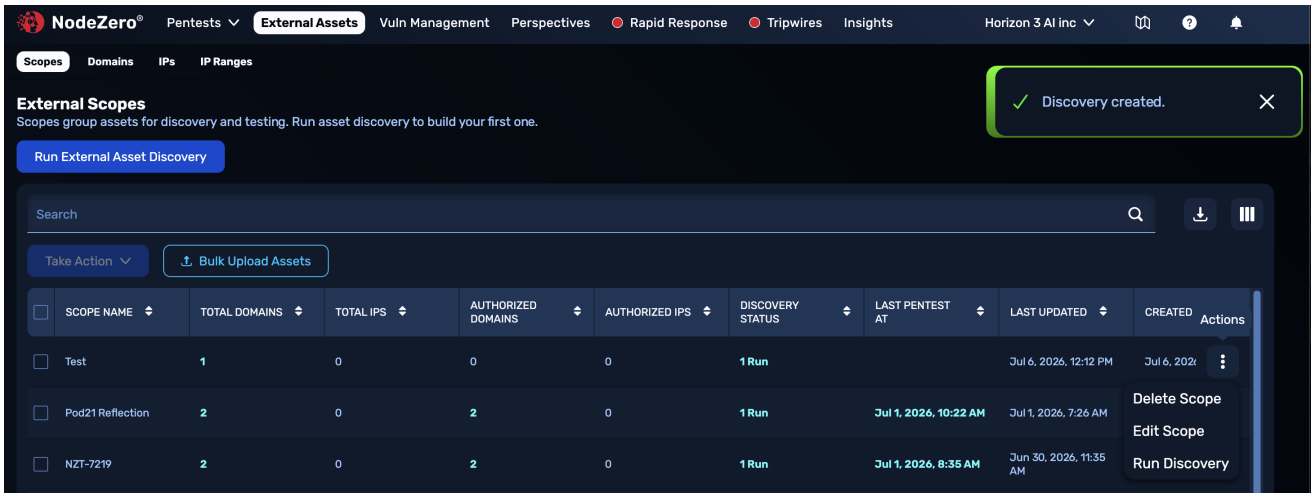
Run Asset Discovery

NodeZero's External Asset Discovery is an enumeration capability that finds all the assets linked to your organization. It leverages DNS, passive website crawling, certificate scraping, and Open Source Intelligence (OSINT)–gathering capabilities and services.

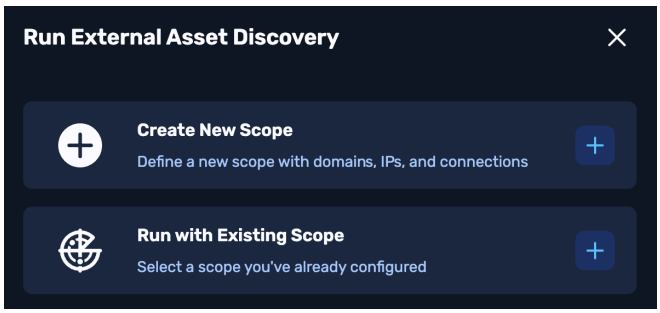
> **Discovery without exploitation**
>
> No exploitation is performed during this operation.

INITIATING DISCOVERY

If you just configured your Scope and selected **Start Discovery**, the NodeZero Portal's **External Scopes** page will display a confirmation banner (shown below) that discovery has begun. As also shown below, you can use an existing Scope's Actions (⋮) menu to initiate discovery, or to edit the Scope's configuration or delete it.



Alternatively, you can click **Run External Asset Discovery** from the upper left and, in the resulting modal, click the **+** button to the right of **Run with Existing Scope**.

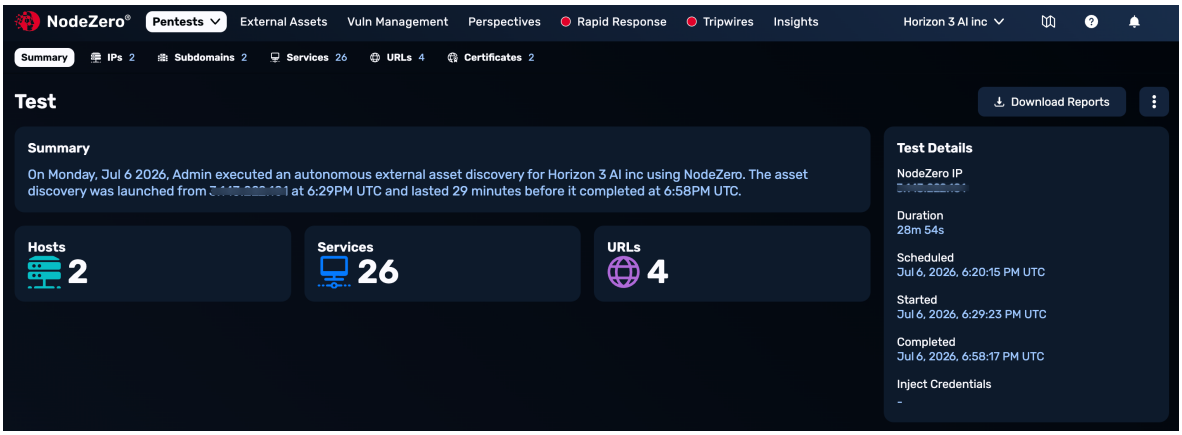


Then, use the **Select a Scope** drop-down and select **Next**.

The modal will expand, enabling you to review, optionally modify, and confirm your Scope's configuration before beginning discovery. (For details about these options, see Create a Scope.)

ASSET DISCOVERY RESULTS

When Asset Discovery completes enumeration, NodeZero sends you an "Asset Discovery complete" email, with a link to display discovery results (as shown below).

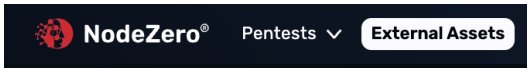


Your next step is to authorize the discovered assets for external pentesting.

Authorize Assets

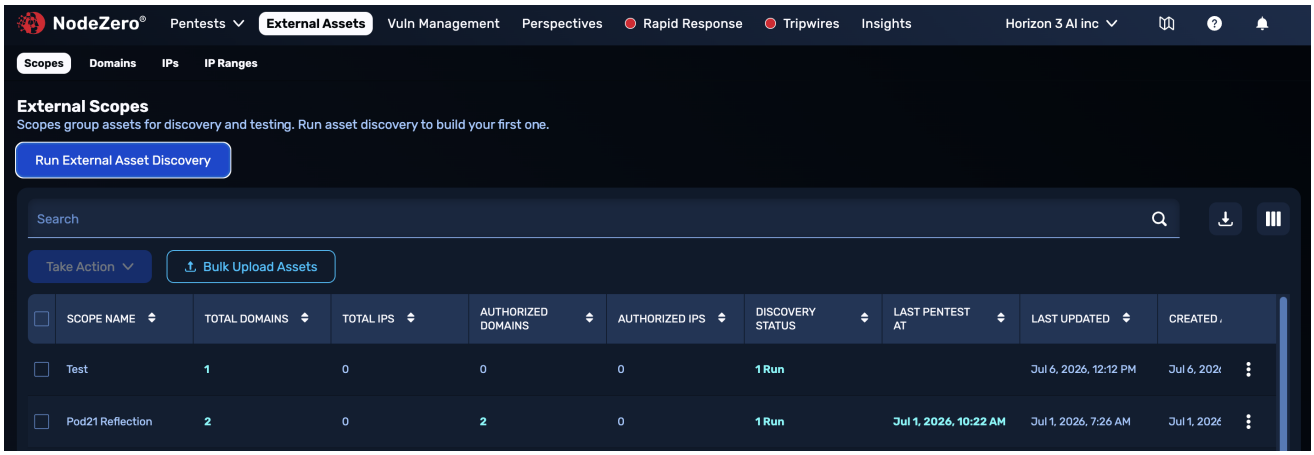
NAVIGATE TO EXTERNAL ASSETS

Now that Asset Discovery has been completed, navigate to External Assets to review and authorize assets for external pentesting.



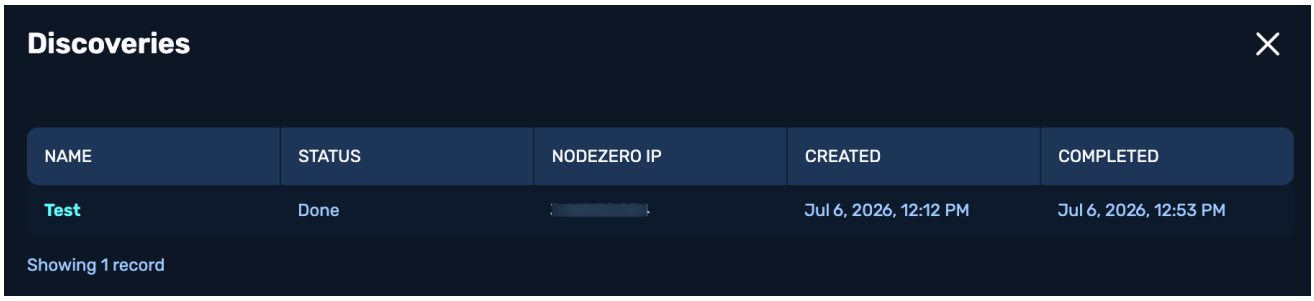
CLICK ON THE SCOPE

Click a Scope of interest whose Discovery Status shows at least `1 Run`.



REVIEW ASSET DISCOVERY

You can check Asset Discovery results and see what has been found.

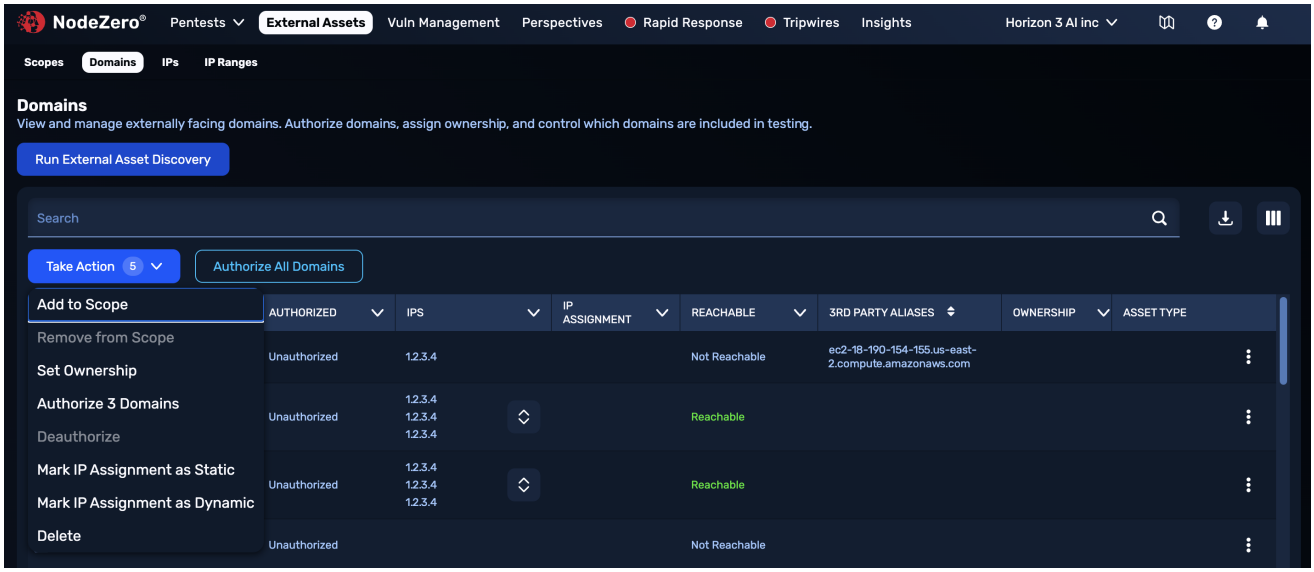


> **Repeat discovery**
>
> Re-run Asset Discovery frequently to maintain the most up-to-date information on the status of your assets. You can automate this using a Scope's Enable Schedule option.

AUTHORIZE DOMAINS

On the External Assets page, use the upper tabs to authorize **Domains** and **IPs** for external pentesting.

On the **Domains** tab, review discovered domains and subdomains identified during Asset Discovery. To authorize assets for a pentest, select the asset(s), click **Take Action**, and then click **Authorize Domain(s)**. You'll need to explicitly authorize automatically discovered subdomains.



> **Authorize assets carefully**
>
> Authorize only the assets you are legally permitted to pentest. You are responsible for accurately defining the scope of the Services for both internal and external testing. See Terms and Conditions for more information.

Asset IP Assignment

During an external pentest, NodeZero uses the asset's IP Assignment to determine whether an asset stays in scope for the pentest. You can mark each IP Assignment as Static or Dynamic, using the **Take Action** menu shown in the preceding screenshot. By default, assets are set to Static.

Static Versus Dynamic IP Resolution

Static or Dynamic indicates each IP address' expected stability over time.

Static IP Resolution indicates that the IP address resolved from the domain name is not expected to change. Assets labeled **Static** will be removed from the scope if the asset resolves to a different IP address during the External Pentest, compared to during the Asset Discovery.

Dynamic IP Resolution indicates that the IP address resolved from the domain name is expected to potentially change over time. Assets labeled **Dynamic** will remain in scope even if the IP address changes between the Asset Discovery and the External Pentest. An example of a Dynamic asset IP would be AWS assets, for which the resolved IP address is controlled by AWS and therefore might change between pentests.

AUTHORIZE IPS

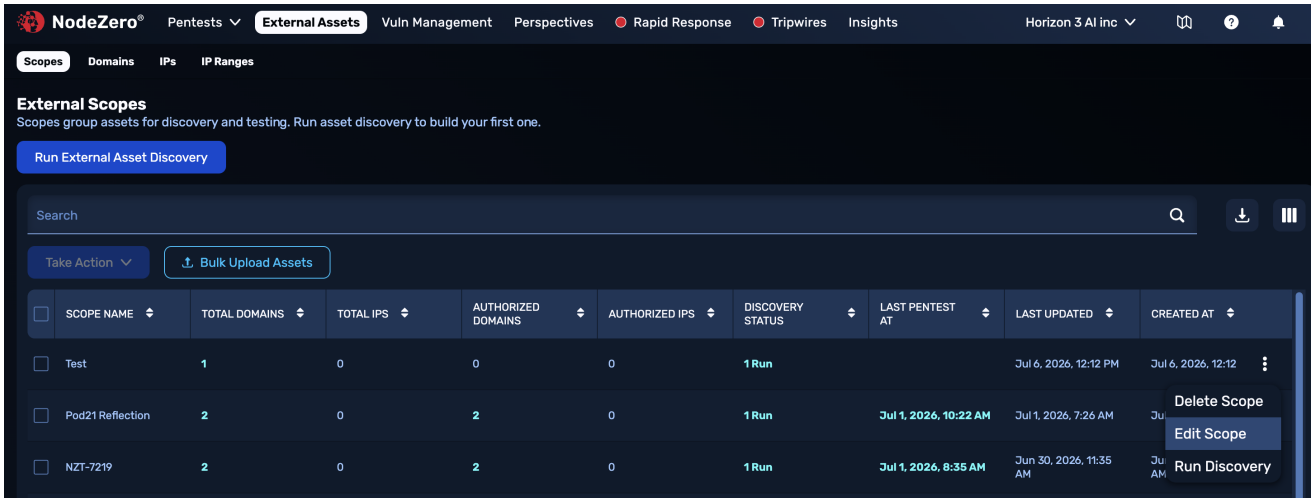
On the **IPs** tab, view reachable IP addresses in the Scope's configuration. To authorize IP addresses for a pentest, select them, click **Take Action**, and then click **Authorize IP(s)**.

IP RANGES FOR DISCOVERY

The **IP Ranges** tab is not used to directly authorize assets. It facilitates populating a Scope with multiple IPs for asset discovery. For details, see Create a Scope.

EDIT CONNECTIONS

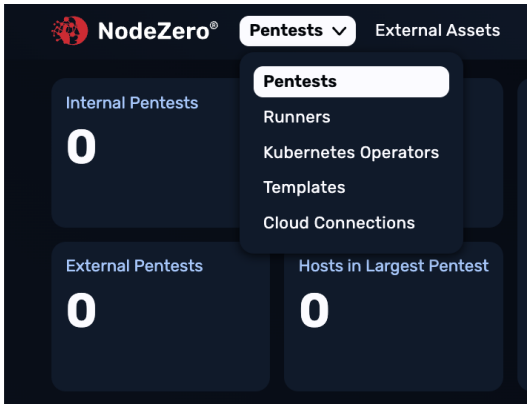
To add or remove connected AWS or Git accounts, edit the Scope's configuration. Click the **Scopes** tab at the upper left, then open the Scope's Actions (⋮) menu and select **Edit Scope**. For details about this and other Scope configuration options, see Create a Scope.



Start an External Pentest

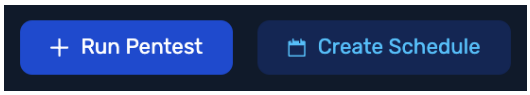
NAVIGATE TO PENTESTS TO RUN AN EXTERNAL PENTEST

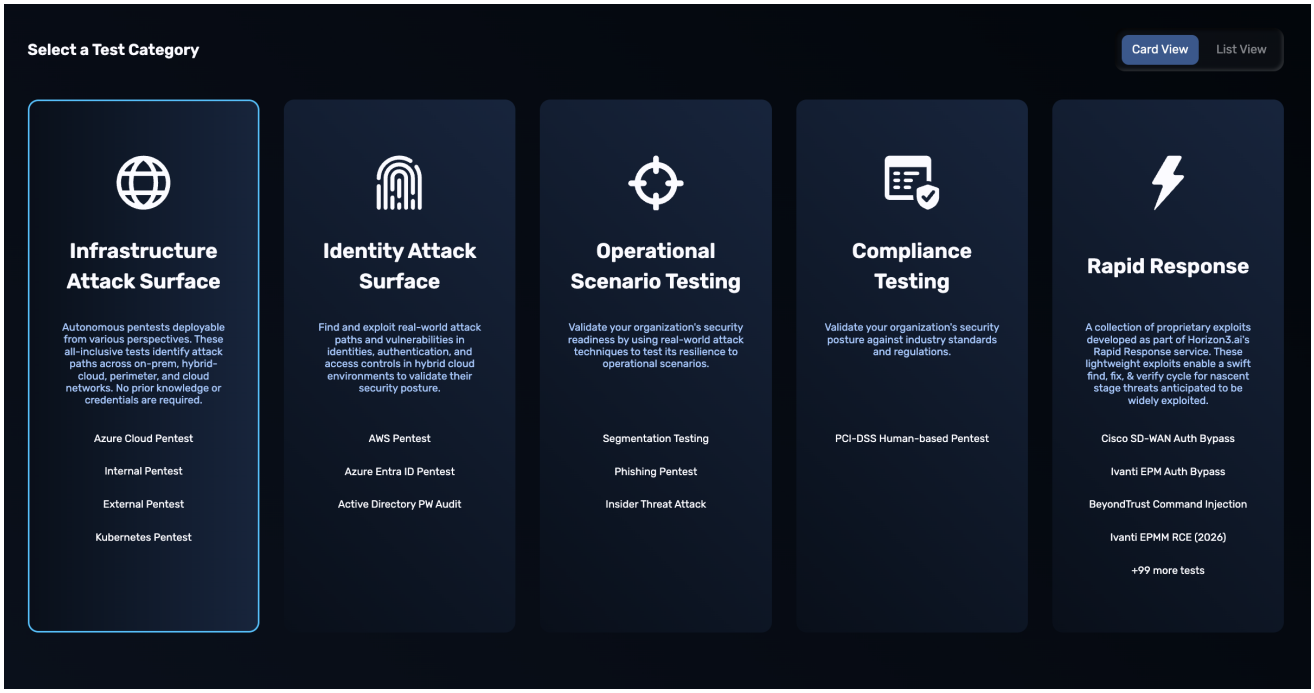
Once a Scope contains the assets authorized for pentesting, you may navigate to the Pentests page to start an external pentest.



CLICK + RUN PENTEST

Click **+ Run Pentest** to open the Pentest Configuration, select **Infrastructure Attack Surface**, then click **External Pentest**.





CONFIGURE THE EXTERNAL PENTEST

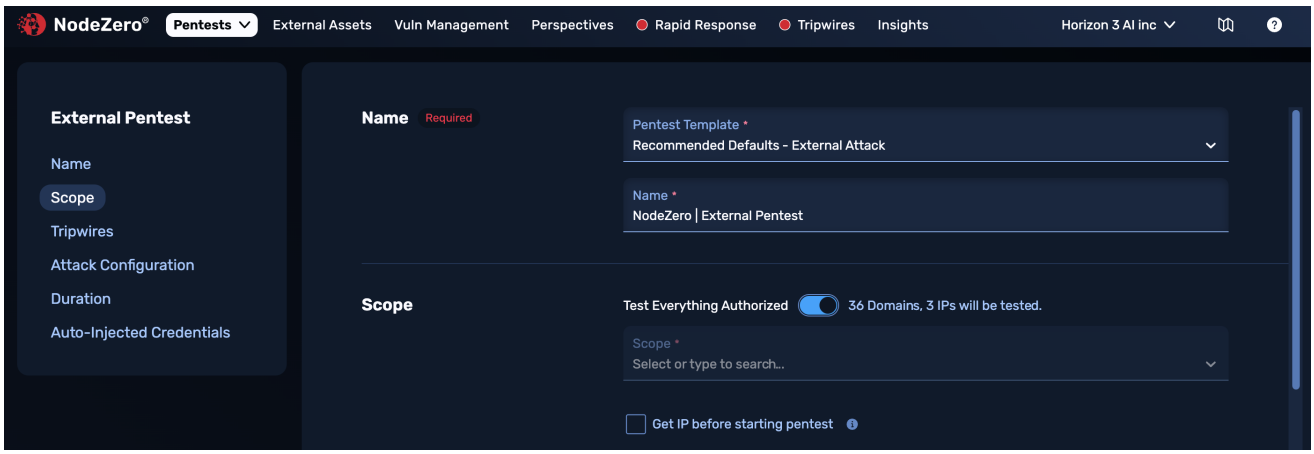
Set a Scope for the external pentest

Select a name and template for the external pentest.

In the **Scopes** panel, the **Test Everything Authorized** option enables testing authorized assets across multiple Scopes. (Test reports will enable breakdowns by Scope.) The adjacent indicators show the total numbers of domains and IPs that are authorized and reachable across all those scopes. Toggle this off if you prefer to select an individual Scope that includes authorized assets.

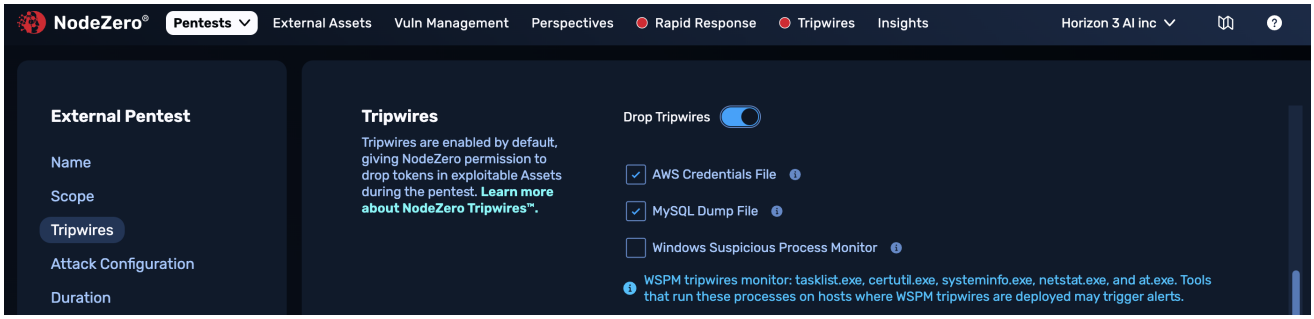
Select **Get IP before starting pentest** where the test's own IP needs to be allowlisted before running the test. Details about this option:

Because this is an external pentest, an IP from outside the perimeter network will be used to attack the network. Some organizations will need to allowlist this IP prior to starting the test, to simulate a breach. NodeZero will email the IP to the user who kicks off the test, and that user will need to add this IP to their allowlist. After adding the IP, resume the test from the Real-Time View.



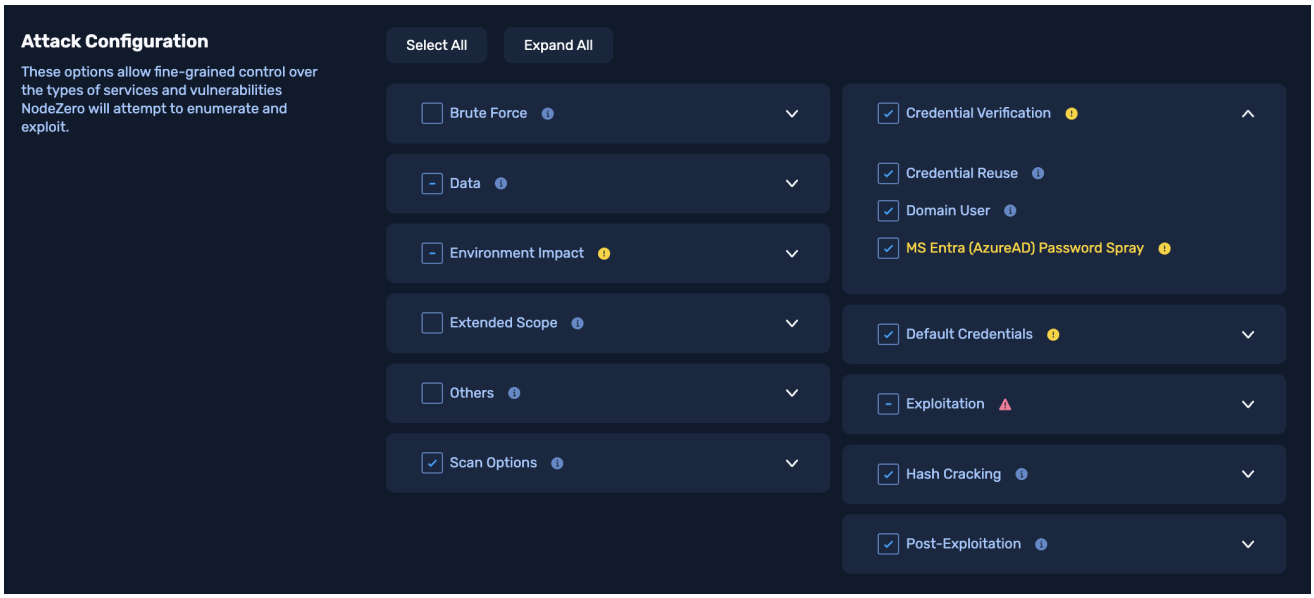
Tripwire Options

Use the **Tripwires** section to configure any Tripwire honeytokens you want to include in your test.



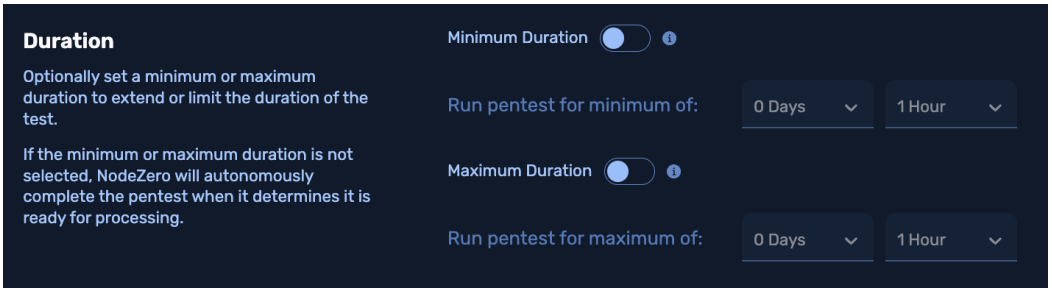
Advanced Configuration Options

Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. Expand each category to see the options within that category. When you're done, scroll to the next section.



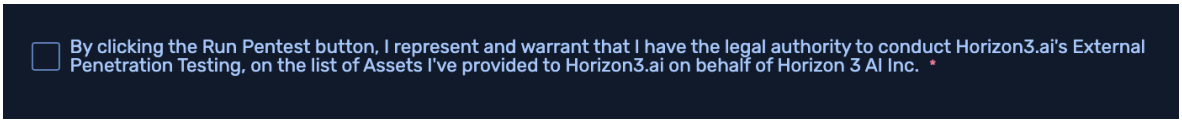
Additional Pentest Options

Optionally, set a minimum or maximum amount of time to allow some attacks to have more time, or limit their overall run time.



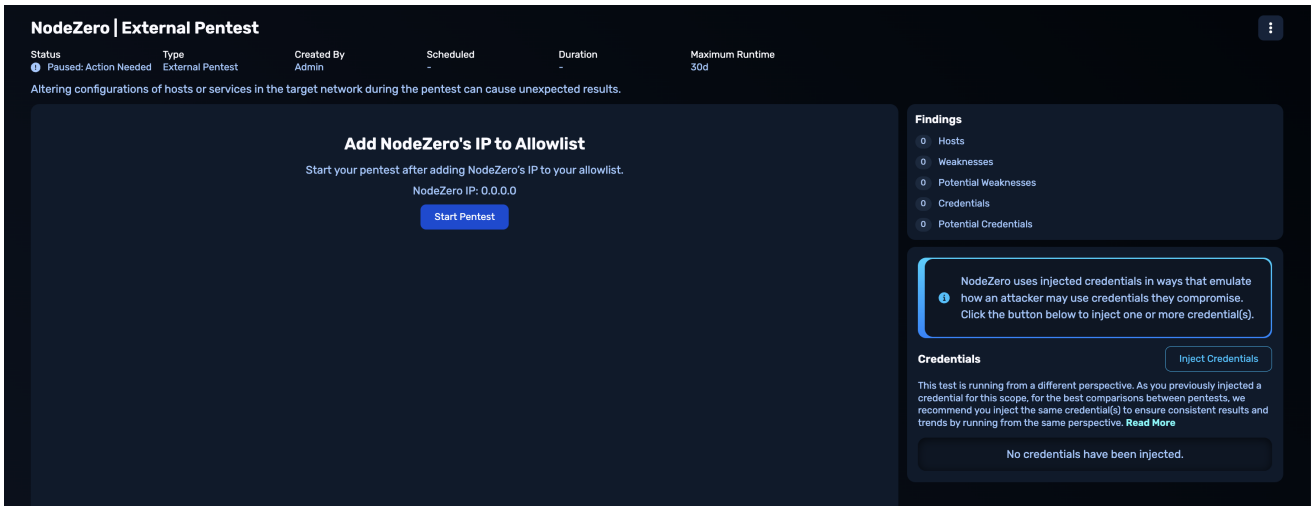
Review the External Pentest Configuration

Once satisfied with your pentest selections, scroll to the bottom of the page and check the box to indicate you represent, and have, the legal authority to conduct Horizon3.ai's External Penetration Testing on the list of authorized assets. Then click **Run Pentest**.

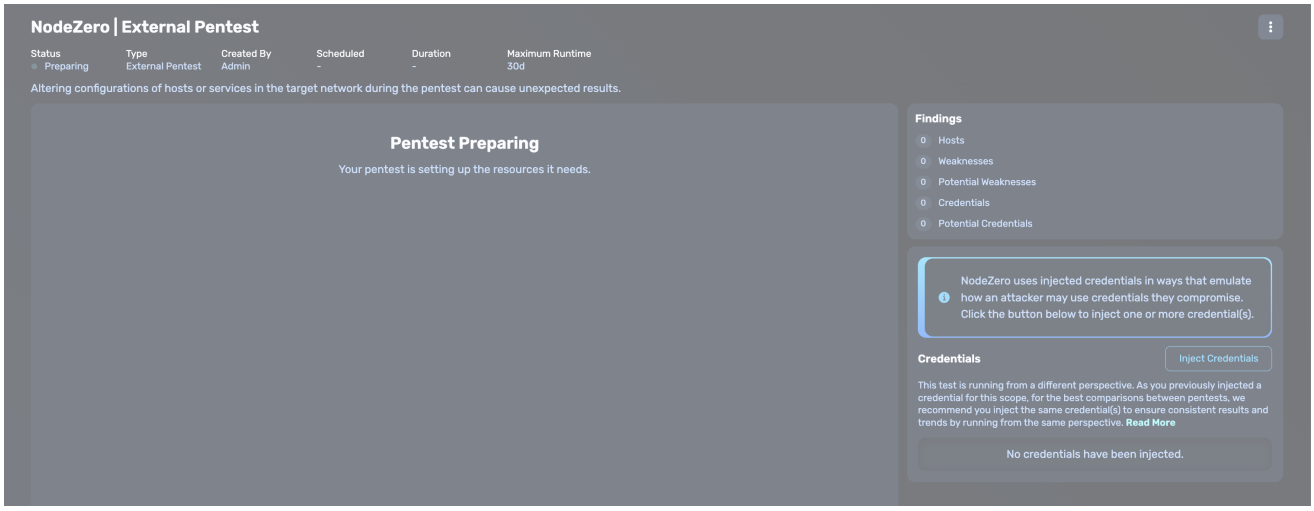


Getting NodeZero IP

If you select **Get NodeZero IP**, the pentest will start in a paused state, so that you can add the NodeZero IP to your allowlist. Once you've completed that step, return to this page and resume the pentest.



Running the External Pentest



NodeZero can also run pentests from an authenticated perspective. Go to the Real-Time View and Inject Credentials to see the impact an attacker would have by leveraging compromised credentials!

You've started an External Pentest

NodeZero sends an email once the external pentest completes.

3.2.4 Kubernetes Pentest

Kubernetes Pentest

The NodeZero Kubernetes Pentest runs from inside your Kubernetes cluster to test the security of your cluster by identifying security misconfigurations and weaknesses that exist in your cluster.

? Why should I run a NodeZero Kubernetes Pentest?
▶ How to Run a Kubernetes Pentest

WHY SHOULD I RUN A NODEZERO KUBERNETES PENTEST?

NodeZero runs from inside your cluster to help simulate the scenario that an attacker has gained a foothold on your cluster.

The foothold could be because an attacker found a user's kubeconfig file in a git repository or maybe an attacker compromised a web application running inside a pod in your cluster.

By running the Kubernetes pentest from inside your cluster and fixing the identified weaknesses, your organization can reduce the risk of the risk and impact of these attacks against your Kubernetes cluster.

HOW TO RUN A KUBERNETES PENTEST

To conduct a Kubernetes Pentest against your cluster, you must first install the NodeZero Operator. The NodeZero Operator will be the launch point for all the Kubernetes Pentests you run against your cluster. You only need to install the operator once per cluster allowing you to continuously test the security of your Kubernetes clusters.

Once installed, the operator can be used to install a Runner or a single pentest.

- Install the NodeZero Operator
- Install a Kubernetes Runner
- Run a Kubernetes Pentest

INSTALL THE NODEZERO OPERATOR

Installation Requirements:

The NodeZero Operator and Kubernetes Pentest is supported on Kubernetes Cluster versions 1.24.0 and above.

You will need to have a terminal with kubectl and helm installed. Make sure kubectl is configured to use the context of your Kubernetes cluster.

The NodeZero Operator will need access to the following domains over port 443/TCP (Horizon3.ai's NodeZero Gateway):

US-Based

- gateway.horizon3ai.com
- api.gateway.horizon3ai.com
- registry.gateway.horizon3ai.com
- interact.gateway.horizon3ai.com

EU-Based

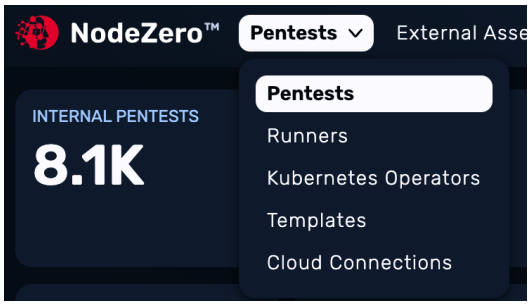
- gateway.horizon3ai.eu
- api.gateway.horizon3ai.eu
- registry.gateway.horizon3ai.eu
- registry.gateway.horizon3ai.com
- interact.gateway.horizon3ai.eu

Does the NodeZero Gateway feature support the NodeZero Operator?

The NodeZero Operator and NodeZero Kubernetes Pentests are supported by NodeZero Gateway.

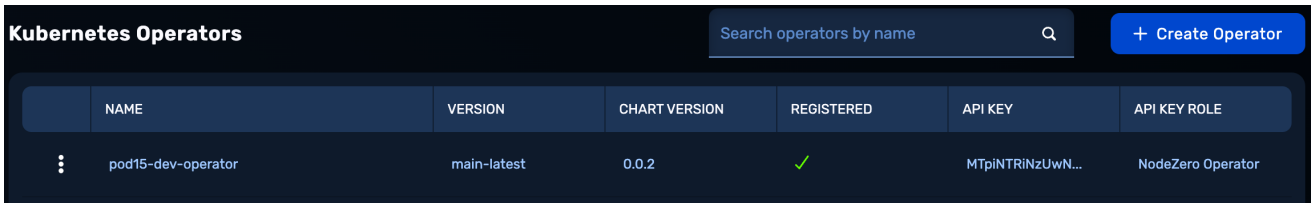
Navigate to Kubernetes Operators

From the Pentest drop-down, select `Kubernetes Operators`.

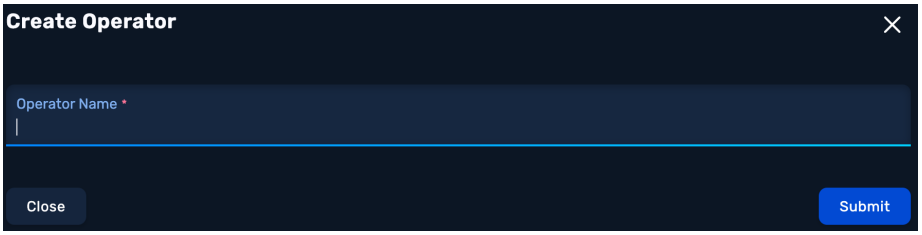


Create Operator

Click the `+ Create Operator` button.



Input a descriptive name and click `Submit`.

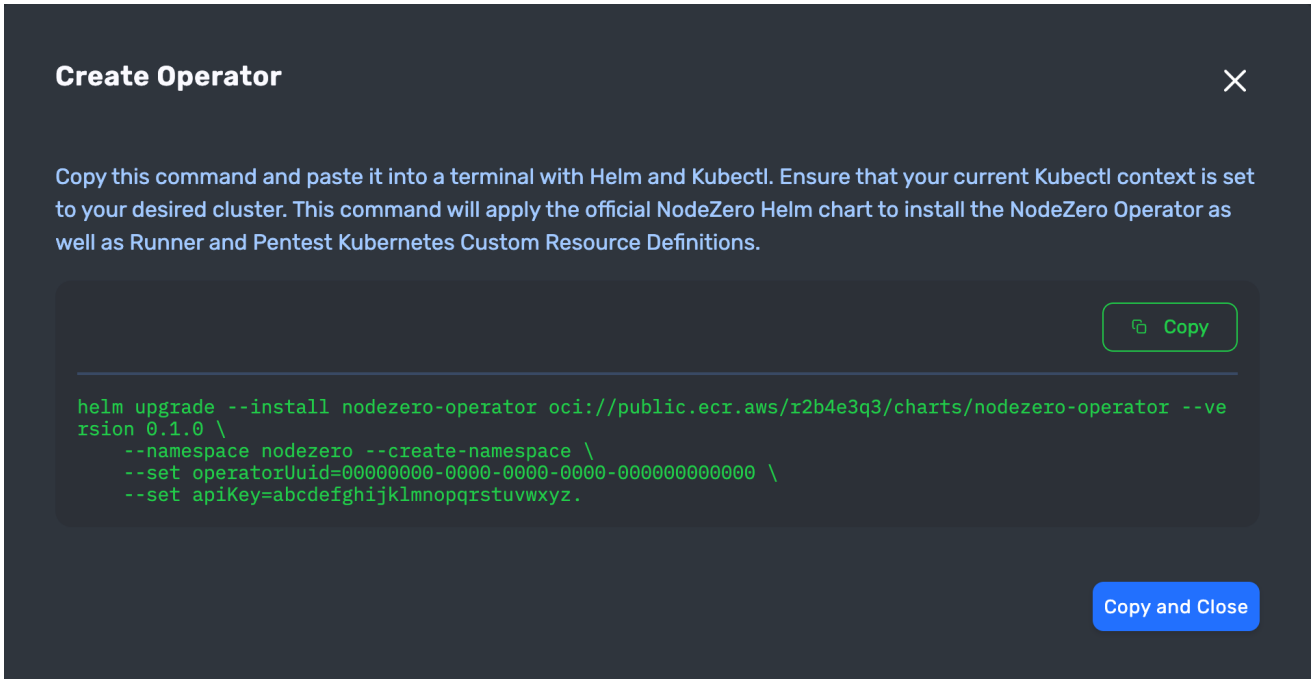


Install Operator

Follow the installation directions by copying the `helm upgrade` command and pasting into a terminal with Helm and Kubectl.

What permissions are needed to install the NodeZero Operator?

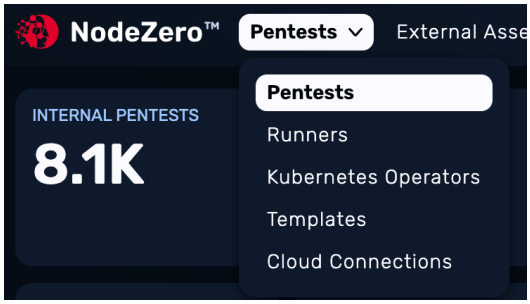
An example ClusterRole CRD with the needed permissions can be found here.



INSTALL A KUBERNETES RUNNER

Navigate to Runners

From the Pentest drop-down, select `Runners`.



Create Runner

Click the `Install Runner` button.

Specify a name for the Runner, select the `Use Kubernetes Runner` toggle, specify a namespace, and then click `Submit`.

Runner per namespace

A Kubernetes Runner lives inside a namespace, meaning that any pentests that use that Runner will run from that namespace. If desired, you can install a Runner in each namespace.



Install Runner

Copy this command and paste it into a terminal with Kubectl. Ensure that your current Kubectl context is set to your desired cluster. This one-time command will install the Runner using the NodeZero Runner Kubernetes Custom Resource Definition.

What permissions are needed to install the NodeZero Runner?

An example ClusterRole CRD with the needed permissions can be found here.

RUN A KUBERNETES PENTEST

You can deploy NodeZero directly into any Kubernetes cluster, manage or self-managed, to pentest its security controls and vulnerabilities. Follow these steps to run a kubernetes pentest within your network.

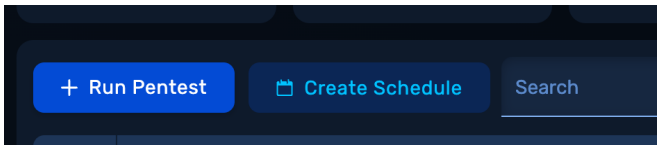
Navigate to Pentests

Once you've installed a NodeZero Operator in your cluster, navigate to Pentests to start a Kubernetes pentest.



Click + Run Pentest

Click + Run Pentest to open Test Categories screen, and select Kubernetes Pentest from the "Infrastructure Attack Surface" category.



Configure the Kubernetes Pentest

Name the Kubernetes Pentest

Name the Kubernetes Pentest and select a pentest template. For best results, etermine and follow a naming convention that will allow you to quickly find a pentest from your pentest list.

An example format: `[date]|[library]|[NodeZero Src]|[scope]`

This format might give you a name like: `2021-09-01|NodeZero|East-Coast-Bizops-Cluster|Full` – which indicates that the NodeZero host was placed in the East Coast Bizops cluster, and the cluster CIDR ranges were explicitly added to scope.

Select the Kubernetes Scope

Since the goal of the Kubernetes Pentest is to test your cluster, all Kubernetes resources are treated in scope including nodes, pods, and services. This means you cannot exclude any Kubernetes resources from testing.

The Kubernetes pentest scope is a set of IPs and/or subnets (in CIDR notation) that you explicitly want NodeZero to test.

Depending on where your NodeZero host is deployed, reachability to the provided scope can be affected by how much privilege, access, and authentication NodeZero can capture or is given.

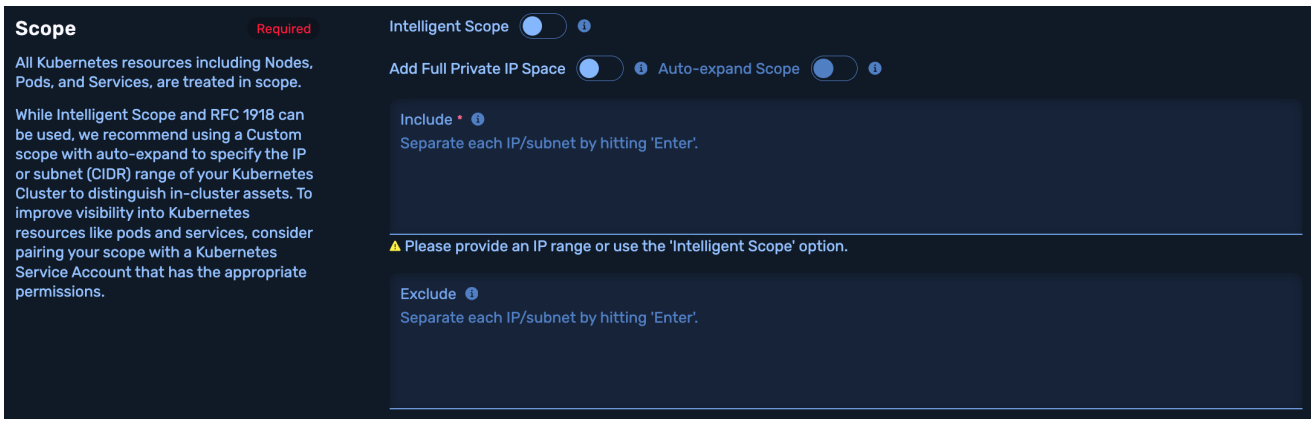
Unlike vulnerability tools that scan for device fingerprints and services, NodeZero enumerates devices in scope and then chains together context, exploitable vulnerabilities, misconfiguration, insecure or weak controls, and any data or loot that it captures, to identify attack paths by priority of impact while providing proof. Typically, the larger the scope, the more findings you'll see, as NodeZero has much more to chain together.

There are three options for the Include section (defined below):

- Intelligent Scope
- Full RFC 1918 (192.168.0.0/16, 172.16.10.0/24, 10.0.0.0/8)
- Custom IP(s) or Subnet(s)
- Auto-expand (Optional)

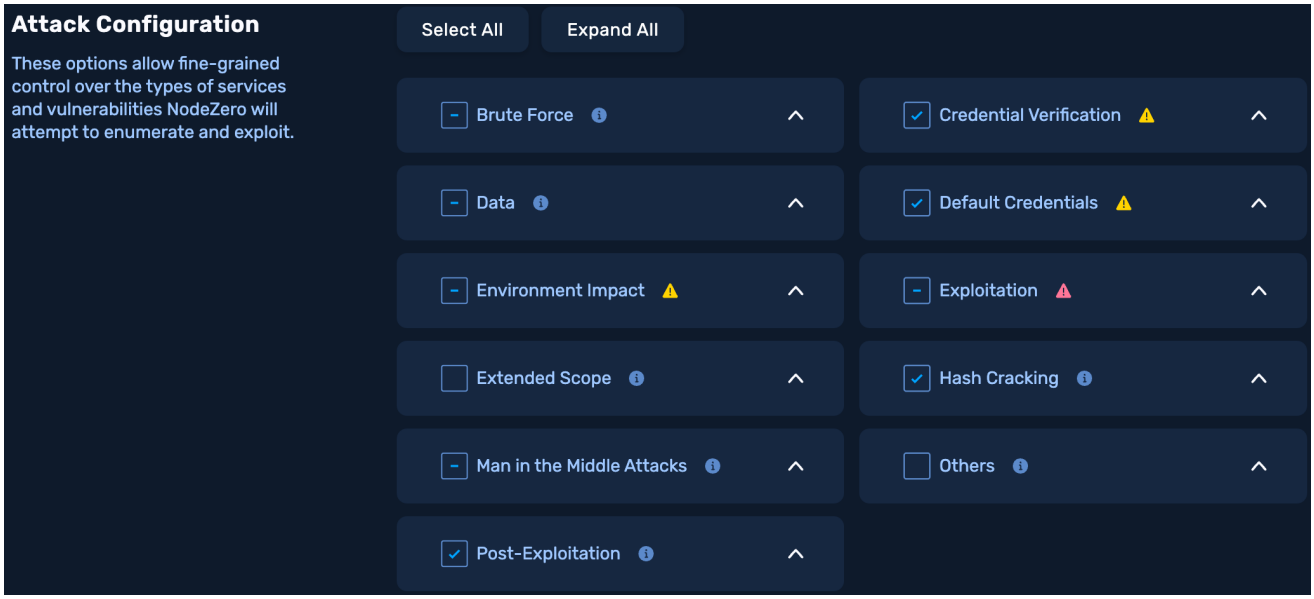
While Intelligent Scope and RFC 1918 can be used, we recommend using a Custom scope with auto-expand to specify the IP or subnet (CIDR) range of your Kubernetes Cluster to distinguish in-cluster assets. Without specifying this NodeZero has a hard time determining what is inside or outside your cluster and therefore what is in scope and should be tested. To improve visibility into Kubernetes resources like pods and services, consider pairing your scope with a Kubernetes Service Account that has the appropriate permissions.

We also recommend providing the CIDR ranges of the internal network or VPC that your cluster resides in to see if NodeZero can reach outside the cluster to your other hosts.



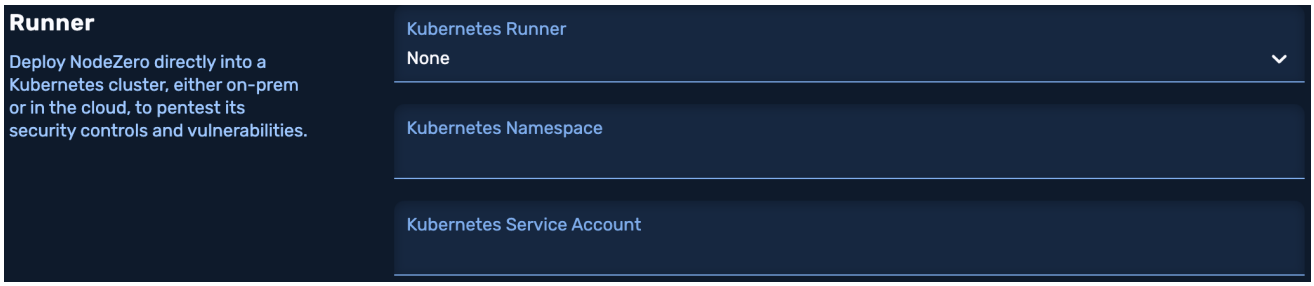
Attack Configuration

Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. For more information on choosing attack configuration options, visit the attack configuration options page.



Runner

If you would like to deploy the pentest using a Runner, feel free to select one! Please note that Kubernetes pentests can only use Kubernetes Runners.



Kubernetes Settings

You can specify a `Kubernetes Namespace` and a `Kubernetes Service Account` that you would like NodeZero to use.

These are powerful settings that enable you to simulate the scenario where an attacker compromises a pod in your cluster.

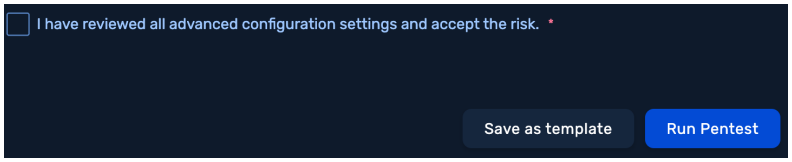
To simulate this scenario simply specify the namespace and service account that one of your pods is configured with.

We recommend simulating pods that have ports accessible from outside the cluster as this is one of the more likely entry points for an attacker into your cluster.

> **Runner per namespace**
>
> Since a Kubernetes Runner lives inside a namespace, when you select a Runner, the pentest must run from that namespace, and you will not be able to customize that field.

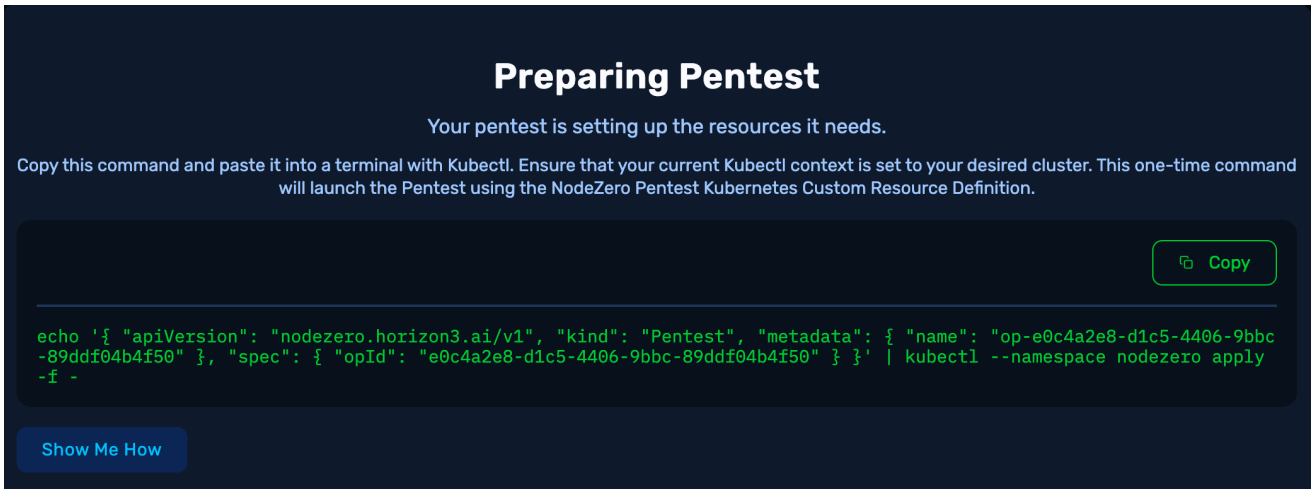
Review the Kubernetes Pentest Configuration

Once satisfied with your pentest selections, check the box to indicate you've reviewed all advanced configuration settings. Then click Run Pentest, which launches the Kubernetes pentest.



Deploy NodeZero

While the pentest is provisioning, its companion one-time-use software module, NodeZero, is made ready for deployment on your NodeZero Host.

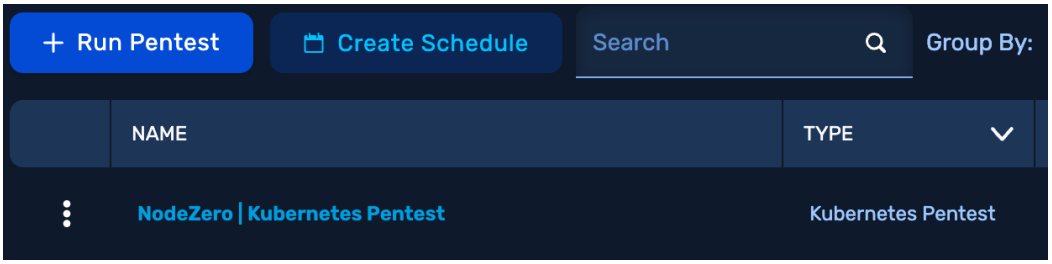


Copy this command and paste it into a terminal with Kubectl. Ensure that your current Kubectl context is set to your desired cluster. This one-time command will launch the Pentest using the NodeZero Pentest Kubernetes Custom Resource Definition.

What permissions are needed to deploy the pentest?

An example ClusterRole CRD with the needed permissions can be found here.

Real-Time View will show up, from where you can Inject Credentials and monitor pentest progress.



You've started a Kubernetes Pentest

NodeZero sends an email once the Kubernetes Pentest completes.

UNINSTALL THE NODEZERO OPERATOR

If you've used the `helm upgrade` command outlined above in Install Operator, you have the option to later uninstall the operator as follows:

1. Verify releases with: `helm ls -n <the-namespace>`. (This will typically be: `helm ls -n nodezero`.)
2. Uninstall the operator with a command of the form: `helm uninstall nodezero-operator -n nodezero`.

Operator ClusterRole CRD

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: nodezero-operator-requirements
rules:
- apiGroups: [""]
  resources: ["namespace", "secrets", "serviceaccounts", "services"]
  verbs: ["*"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["*"]
- apiGroups: ["admissionregistration.k8s.io/v1"]
  resources: ["mutatingwebhookconfigurations", "validatingwebhookconfigurations"]
  verbs: ["*"]
- apiGroups: ["apiextensions.k8s.io"]
  resources: ["customresourcedefinitions"]
  verbs: ["*"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["clusterrole", "clusterrolebinding", "role", "rolebinding"]
  verbs: ["*"]
- apiGroups: ["networking.k8s.io/v1"]
  resources: ["networkpolicies"]
  verbs: ["*"]
```

Pentest & Runner ClusterRole CRD

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: nodezero-pentest-runner-permissions
rules:
- apiGroups: ["nodezero.horizon3.ai"]
  resources: ["pentest", "runner"]
  verbs: ["*"]
```

3.2.5 AWS Pentest

AWS Pentest

The NodeZero AWS Pentest is a gray-box–style pentest that uses a role with read-only permissions to comprehensively assess the security of your AWS accounts.

? Why Should I Run a NodeZero AWS Pentest?
▶ How to Run an AWS Pentest

WHY SHOULD I RUN A NODEZERO AWS PENTEST?

An AWS pentest is a security test focused on determining exploitable weaknesses that exist in an AWS account – weaknesses that an attacker could use to harm the organization.

The NodeZero AWS pentest is a gray-box–style pentest because without being able to see what exists in an AWS account, it is impossible to assess the account's security. With read-only permissions, NodeZero gains a comprehensive view of your AWS account, and is able to determine the exploitable weaknesses that could expose your organization to the worst impacts.

While a black-box pentest is the most realistic, a gray-box approach is often necessary to find the most critical weaknesses that, given enough time, an attacker will find. NodeZero conducts the AWS pentest in compliance with AWS Penetration Testing policies.

HOW TO RUN AN AWS PENTEST

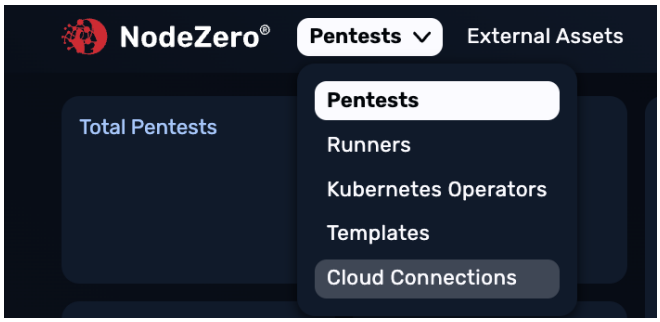
To conduct an AWS pentest, you must first set up an AWS cloud connection. An AWS cloud connection is what enables NodeZero to assume a role in your AWS account. These cloud connections are long-lived, meaning you need to set up only one cloud connection for each of your AWS accounts. An AWS Pentest can use more than one connection.

- Create an AWS Cloud Connection
- Create AWS Cloud Connections in Bulk
- Start an AWS Pentest

Create an AWS Cloud Connection

1. Navigate to Cloud Connections

Navigate to the Cloud Connections page, you will need to create a cloud connection for each of your AWS accounts you would like to test.



2. Click Create Cloud Connection

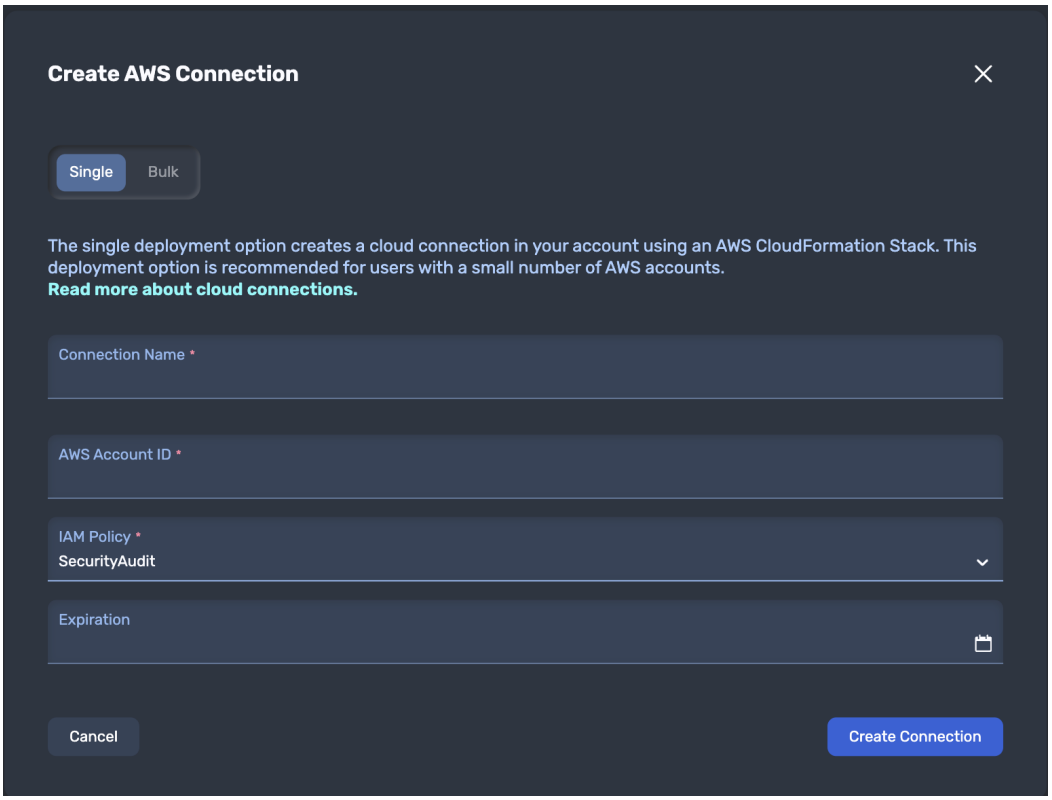
On the Cloud Connections page, click `+ Connection` to open the connection creation form.

3. Configure the Cloud Connection

Name the Cloud Connection, enter an AWS account ID, select the IAM policy you want the connection to use and, if desired, add an expiration date. Click `Submit`.

Which IAM policy should I choose?

- SecurityAudit: An AWS Managed Policy that only grants access to read security configuration metadata. This allows NodeZero to analyze IAM and security configurations of cloud resources.
- ReadOnlyAccess: An AWS Managed Policy that provides read-only access to all AWS services and resources. This allows NodeZero to analyze IAM, security configurations of cloud resources, and sensitive information stored in cloud resources such as S3 buckets and Lambda functions.



4. Launch the CloudFormation Stack

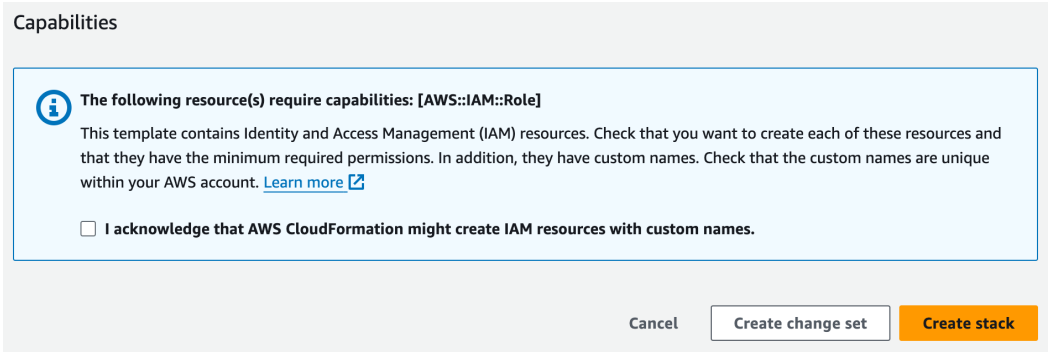
What permissions are needed to deploy the CloudFormation stack?

- cloudformation:CreateStack
- iam:GetRole
- iam:CreateRole
- iam:PutRole
- iam:AttachRolePolicy
- lambda:InvokeFunction

Log in to the AWS account you specified in the previous step.

Click the CloudFormation `Launch Stack` button. 

You will be redirected to a prepopulated AWS "Quick create stack" form. Select the `I acknowledge` check box and click `Create Stack`.

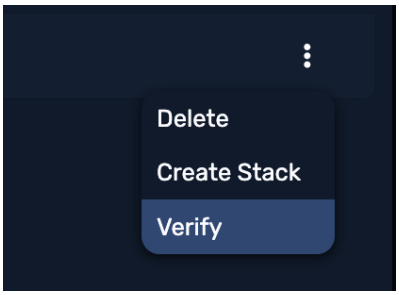


AWS CLI

Instead of using the `Launch Stack` button, you can use the AWS CLI and the CloudFormation CLI command provided in the `Create Connection` form.

5. Verify that the Connection Was Created Properly

After the CloudFormation stack has successfully been created, click the `Verify Connection` button to confirm that the role was created properly. Cloud connections will be automatically re-verified once every 24 hours. You can manually re-verify a connection by using the action menu in the connection table.



Create AWS Cloud Connections in Bulk

Advanced option

The bulk connection setup process is intended for experienced users familiar with AWS IAM and AWS CloudFormation StackSets. The setup process uses AWS CloudFormation StackSets to set up the cloud connections in all of your selected AWS accounts from a single account.

1. Navigate to Cloud Connections

Navigate to the `Cloud Connections` page. You will need to create a cloud connection for each of your AWS accounts you would like to test.

2. Click Create Cloud Connection

On the `Cloud Connections` page, click `+ Connection` to open the connection creation form, and select `Bulk`.

3. Configure the Cloud Connection

Name the Cloud Connection, and provide a list of comma-separated AWS account IDs. Select the IAM policy you want the connection to use and, if desired, add an expiration date. Click `Submit`.

Which IAM policy should I choose?

- `SecurityAudit`: An AWS Managed Policy that only grants access to read security configuration metadata. This allows NodeZero to analyze IAM and security configurations of cloud resources.
- `ReadOnlyAccess`: An AWS Managed Policy that provides read-only access to all AWS services and resources. This allows NodeZero to analyze IAM, security configurations of cloud resources, and sensitive information stored in cloud resources such as S3 buckets and Lambda functions.

4. Run the CloudFormation StackSets Deployment Script

Configure a terminal with the AWS CLI, and authenticate to your AWS StackSets administrator account. In the terminal, run the CloudFormation StackSets deployment script that you copied from the module displayed above.

5. Verify the Connections Were Created Properly

After the deployment script runs and both the StackSet and StackSet instances have successfully been created, click the `Verify Connections` button to confirm that the role was created properly.

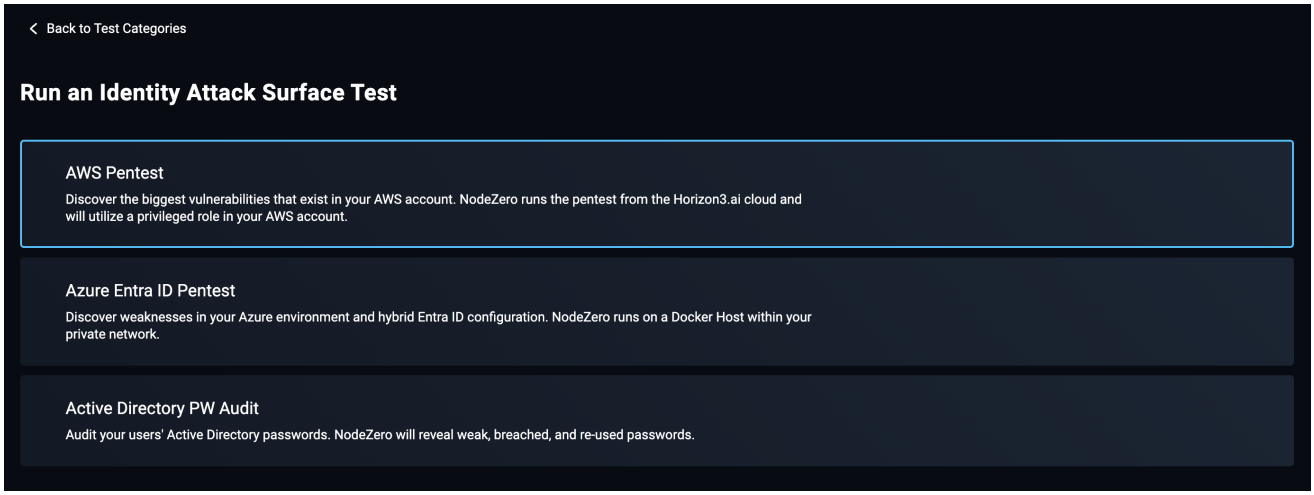
Cloud connections will be automatically re-verified once every 24 hours.

Start an AWS Pentest

1. Navigate to Pentests to Run an AWS Pentest

Once a Cloud Connection is created and verified, you may navigate to the Pentests page to start an AWS pentest. Click `+ Run Pentest` to open the Pentest Configuration and select the `Identity Attack Surface` test category.

Then select `AWS Pentest`.

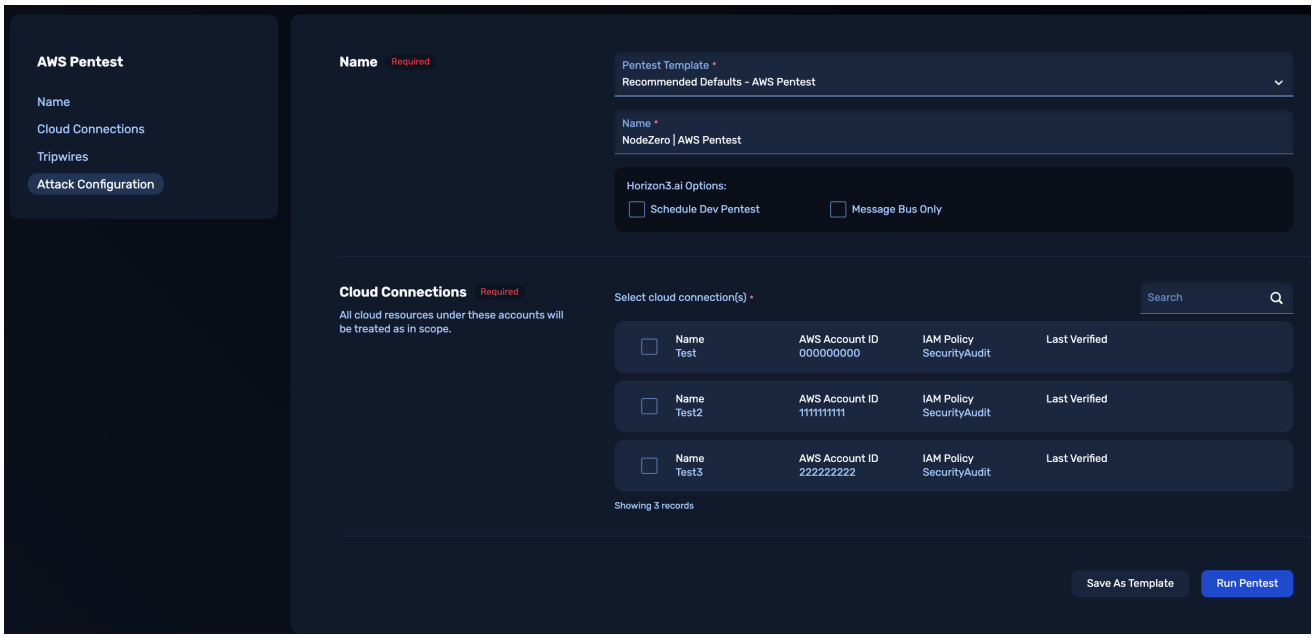


2. Configure the AWS Pentest Select the Cloud Connections

Name the AWS Pentest, select a pentest template, and select the cloud connections you would like to use.

Verified connections

Only verified connections will appear when running a pentest.



Advanced Configuration Options

Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. Click `Next`.

Review the AWS Pentest Configuration

Once satisfied with your pentest selections, check the box to indicate you represent and have the legal authority to conduct Horizon3.ai's AWS Penetration Testing. Then click `Run Pentest`.

Clean Up a Cloudformation StackSet

> **Advanced option**
>
> This is used to clean up the AWS resources created during the bulk cloud connection process that uses AWS Cloudformation Stacksets. The commands shown below make use of the AWS CLI and will need to be ran authenticated to the management account where the Stackset was provisioned.

1. Identify the Stackset Name

Run the following command to view a list of the Stacksets that are in your account and identify the name of your NodeZero stackset. It will be prepended with `NodeZeroStackSet`.

```
aws cloudformation list-stack-sets
```

example output:

```
{
    "Summaries": [
        {
            "StackSetName": "NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111",
            "StackSetId": "NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111:11111111-1111-1111-1111-111111111111",
            "Status": "ACTIVE",
            "DriftStatus": "NOT_CHECKED"
        }
... TRUNCATED
```

2. Delete All Associated Stackset Instances

In order to delete the Stackset, you will need to delete all its associated instances. To do this you will need the same list of aws account ids that was provided to the bulk connection setup process. Provide the Stackset name identified above and the list of space delimited account ids to the command.

```
aws cloudformation delete-stack-instances \
  --stack-set-name $STACKSET_NAME \
  --no-retain-stacks --regions us-east-1 --accounts $ACCOUNT_IDS
```

example:

```
aws cloudformation delete-stack-instances \
  --stack-set-name NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111 \
  --no-retain-stacks --regions us-east-1 --accounts 1111111111111 2222222222222 3333333333333
```

3. Verify the Stackset Instances Have Been Deleted

List the Stackset instances to verify they have been deleted for that specific Stackset.

```
aws cloudformation list-stack-instances \
  --stack-set-name $STACKSET_NAME
```

example:

```
aws cloudformation list-stack-instances \
  --stack-set-name NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111
```

Once deleted, you should see the following output:

```
{
    "Summaries": []
}
```

4. Delete the Stackset

Once all the Stackset instances have been deleted, you can delete the Stackset itself.

```
aws cloudformation delete-stack-set \
  --stack-set-name $STACKSET_NAME
```

example:

```
aws cloudformation delete-stack-set \
  --stack-set-name NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111
```

A successful deletion will not yield any output.

5. Verify Stackset deletion

You can verify that the Stackset was properly deleted by attempting to describe it.

```
aws cloudformation describe-stack-set \
  --stack-set-name $STACKSET_NAME
```

example:

```
aws cloudformation describe-stack-set \
  --stack-set-name NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111
```

If it was properly deleted, you should see a similar error message:

```
An error occurred (StackSetNotFoundException) when calling the DescribeStackSet operation:
  StackSet NodeZeroStackSet-SecurityAudit-Role-11111111-1111-1111-1111-111111111111 not found
```

3.2.6 Azure Entra ID

Azure Entra ID Pentest

The NodeZero Azure Entra ID Pentest is a gray-box style pentest that uses an Azure credential and a privileged Active Directory (AD) credential to enumerate and exploit attack paths to compromising your Entra ID tenant. The Azure Entra ID Pentest runs from a Docker container within your private enterprise network.

? Why should I run a NodeZero Azure Entra ID Pentest? ▶ How to Run an Azure Entra ID Pentest

---

WHY SHOULD I RUN A NODEZERO AZURE ENTRA ID PENTEST?

An Azure Entra ID pentest is a security test focused on determining misconfigurations within your Active Directory and Entra ID hybrid environment that could lead to a full tenant compromise.

The NodeZero Azure Entra ID Pentest is a grey-box style pentest requiring a privileged AD credential with DCSync permissions, and an initial Entra user credential. The privileged AD credential allows NodeZero to simulate an internal Domain Compromise that could enable common attack techniques such as Entra Connect credential dumping, and Azure Seamless SSO Silver Ticket attacks. By providing an initial Entra credential - NodeZero is able to simulate an initial credential compromise and enumerate the Entra ID environment. This initial collection also allows NodeZero to highlight possible privilege escalation misconfigurations for other users within the tenant.

HOW TO RUN AN AZURE ENTRA ID PENTEST

From NodeZero's "Run Pentest" page, the Azure Entra ID Pentest can be found under the Identity Attack Surface category.

Scope

The Azure Entra ID Pentest requires a Domain Controller IP Address. NodeZero will connect to the DC utilizing the privileged domain credential (see below) to query AD via LDAP(S) and gather details about the domain's hybrid Entra ID setup. This includes the location of the Entra Connect application. Additionally, NodeZero will perform a DCSync attack against the DC to simulate a Domain Compromise and collect the Credential for the AZUREADSSOACC$ Machine Account, if present. This credential will be utilized to perform Kerberos Silver Ticket Attacks, if possible.

NOTE: The host running Entra Connect is automatically consider "In Scope" for the Azure Entra ID pentest once it is enumerated.

Privileged Domain Credential

The Azure Entra ID pentest requires a Privileged Domain Credential with DCSync privileges -- much like the NodeZero AD Password Audit. This allows NodeZero to simulate a Domain Compromise, enabling NodeZero to evaluate several common attack vectors and demonstrate how an on-premise compromise could enable an attacker to compromise your Entra ID tenant.

Azure Credential

An initial Azure Entra ID user credential is required for the Azure Entra ID pentest. As of October 2024, Microsoft has mandated the use of Multifactor Authentication for Azure services. NodeZero now utilizes the Device Code Flow (aka OAuth Device Authorization Grant) to request an initial credential for the Azure Entra ID Pentest. During configuration, users will be asked to provide their Entra tenant ID instead of a username/password. Once the pentest begins, users will receive an email notification that NodeZero is ready for them to return to the Portal and complete the authentication and authorization steps required for NodeZero to access the tenant. Please see Injecting Azure MFA Credentials. This allows NodeZero to simulate an initial access vector to your Entra environment. Since Entra allows user's to enumerate a significant portion of the tenant's configuration by default, this credential does not need to be privileged.

Once NodeZero enumerates the environment utilizing this credential, it will attempt to find and exploit a path to compromising the tenant using the credential as a starting point. Additionally, NodeZero will show any paths to full tenant compromise it enumerates from other users utilizing Azurehound.

3.2.7 AD Password Audit

AD Password Audit

NodeZero AD Password Audit is an easy way to audit the strength and similarity of user passwords in an Active Directory environment. Read below for motivations, necessary environment configurations, and instructions on running an AD Password Audit.

- Motivations for AD Password Audit
- How to Configure Your Environment
- How to Run an AD Password Audit

MOTIVATIONS FOR AD PASSWORD AUDIT

Why, and how often, should you run an AD Password Audit?

Reasons to Perform a Password Audit

Weak passwords factor into a number of commonly used attacker techniques:

MITRE Tactic	MITRE Technique / Sub-Technique
TA0001: Initial Access	T1078.002: Valid Accounts: Domain Accounts
TA0006: Credential Access	T1110.0001: Brute Force: Password Guessing
TA0006: Credential Access	T1110.0002: Brute Force: Password Cracking
TA0006: Credential Access	T1110.0003: Brute Force: Password Spraying
TA0006: Credential Access	T1110.0004: Brute Force: Credential Stuffing
TA0006: Credential Access	T1621: Multi-Factor Authentication Request Generation

Attackers take advantage of weak passwords for both initial access and lateral movement.

On the perimeter, it's common for attackers to target endpoints not enabled for MFA. For cases that MFA is enabled, having a user's password becomes the first stage to subsequent attacks such as MFA fatigue attacks.

On internal networks, attackers generally don't need to worry about MFA as there are many endpoints and protocols (e.g. SMB, DCOM/RPC, NTLM) available for abuse that don't support MFA.

Reasons to Run a Password Audit Regularly

It's important to audit passwords regularly because the state of the passwords and accounts in Active Directory changes over time.

- Employees join or leave an organization.
- Employees change their passwords. Many organizations have policies in place that require employees to rotate their passwords.
- New data breaches in the wild can expose passwords previously thought to be safe.

> **Audit passwords regularly**
>
> We recommend running an AD Password Audit at least once every three (3) months.

HOW TO CONFIGURE YOUR ENVIRONMENT

This section reviews the prerequisites to running an AD Password Audit. Instructions are given for configuring your environment to ensure the audit can run successfully.

Already configured your environment?

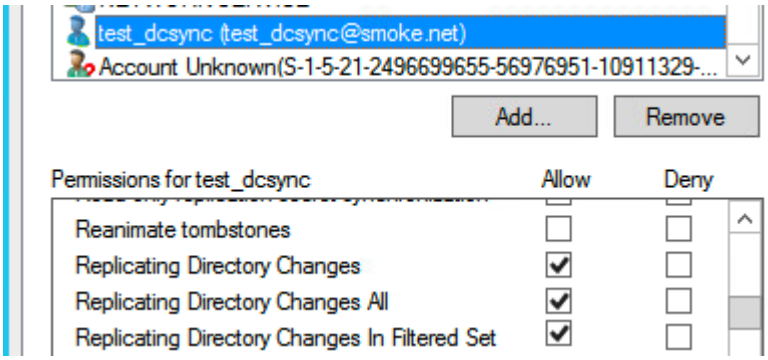
Jump to the section on How to Run an AD Password Audit.

Privileged Account

Prior to running an AD Password Audit, you must have the credentials for a domain user account with DCSync privileges. A domain admin account can be used, but we recommend setting up a separate account with the minimum privileges required. This account must **not** be in the Protected Users security group.

The minimum privileges required are:

```
Replicating Directory Changes
Replicating Directory Changes All
Replicating Directory Changes in Filtered Set
```



Configuring EDR to Allow DCSync

EDR software running on the domain controller may block DCSync from taking place. This is a good thing in general as it prevents attackers from using this well-known attack technique. However, for the sake of the AD Password Audit, DCSync needs to be allowed. We recommend consulting your EDR vendor's documentation for instructions on how to adjust the EDR configuration to enable DCSync. Once you finish running an AD Password Audit, the configuration should be reset to block DCSync again.

Testing DCSync

You can manually verify that DCSync will work using a NodeZero container. On a host where you have already run a NodeZero pentest before, start up an old NodeZero container. (The following instructions cover Docker, but work the same way with Podman.)

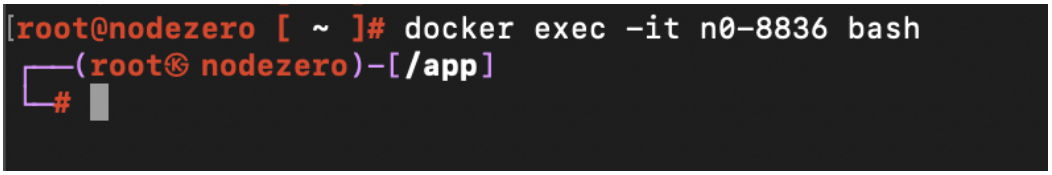
First, run `docker ps -a` to list any existing containers.

```
[root@nodezero [ ~ ]# docker ps -a
CONTAINER ID   IMAGE                                                                COMMAND                  CREATED        STATUS                     PORTS     NAMES
0acc5a87e919   206961115674.dkr.ecr.us-east-2.amazonaws.com/h3/n0:naveen1729.latest   "/docker-entrypoint.…"   28 hours ago   Exited (0) 27 hours ago              n0-8836
root@nodezero [ ~ ]#
```

Then run `docker start n0-xxxx` to start an existing container (substitute `n0-xxxx` for your container's name).

```
[root@nodezero [ ~ ]# docker start n0-8836
n0-8836
[root@nodezero [ ~ ]#
```

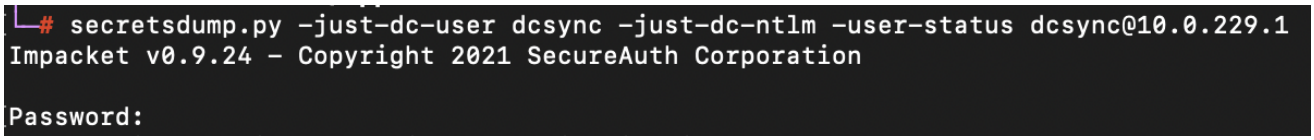
Then run `docker exec -it n0-xxxx bash` to get a shell inside the container.



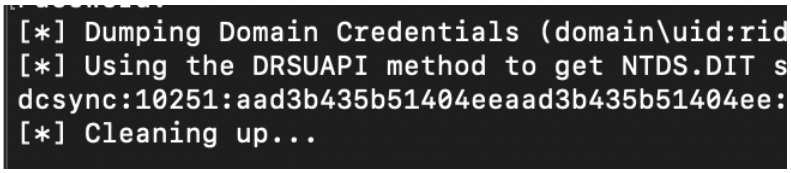
Next, using the privileged account from the previous step, run the `secretsdump` command to perform a DCSync operation (substitute in your account's username and domain controller IP).

```
secretsdump.py -just-dc-user <your_dcsync_user> -just-dc-ntlm -user-status <your_dcsync_user>@<domain_controller_ip>
```

Enter the password for the privileged account.



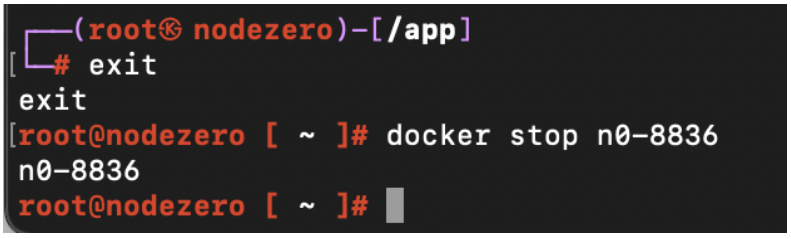
If the DCSync is successful you should see a successful retrieval of the NTLM hash for just the privileged account.



If the DCSync is not successful, you should recheck your account's permissions and EDR configuration.

Finally, after testing, exit the shell and stop the NodeZero container. Note, even if you don't shut it down, the container will automatically shut itself down after ten minutes.

```
docker stop n0-xxxx
```



HOW TO RUN AN AD PASSWORD AUDIT

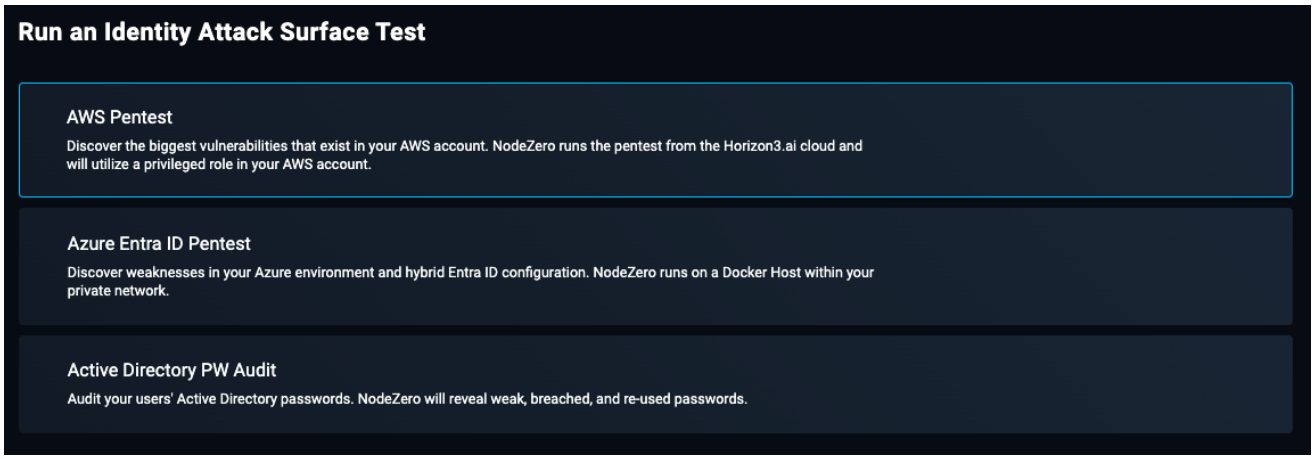
Navigate to Pentests

Navigate to Pentests to start a pentest.

Click `+ RUN PENTEST`

Click `+ RUN PENTEST`, then select the Identity Attack Surface category.

Select `Active Directory PW Audit` to open the configuration for this pentest.



Configure the AD Password Audit

Name and Domain Controller IP Address

Specify the template, the pentest name, and the IP address of a domain controller for the domain you would like NodeZero to audit.

IP access

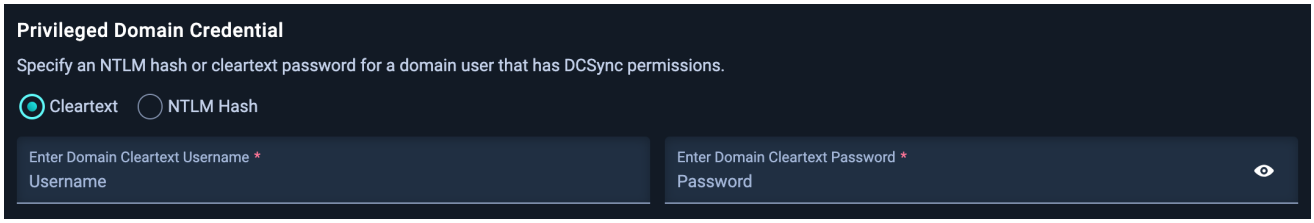
Make sure the IP address is accessible from where your Docker host is deployed.

Privileged Credential

Specify an NTLM hash or cleartext password for a domain user that has DCSync permissions.

DCSync permissions

NodeZero needs DCSync permissions in order to extract the NTDS.dit file from the domain controller. The NTDS.dit file stores Active Directory information about user objects and includes the NTLM hashes for all the users. These NTLM hashes are what NodeZero will attempt to crack. For more information on setting up a credential with the needed permissions, see How to Configure Your Environment



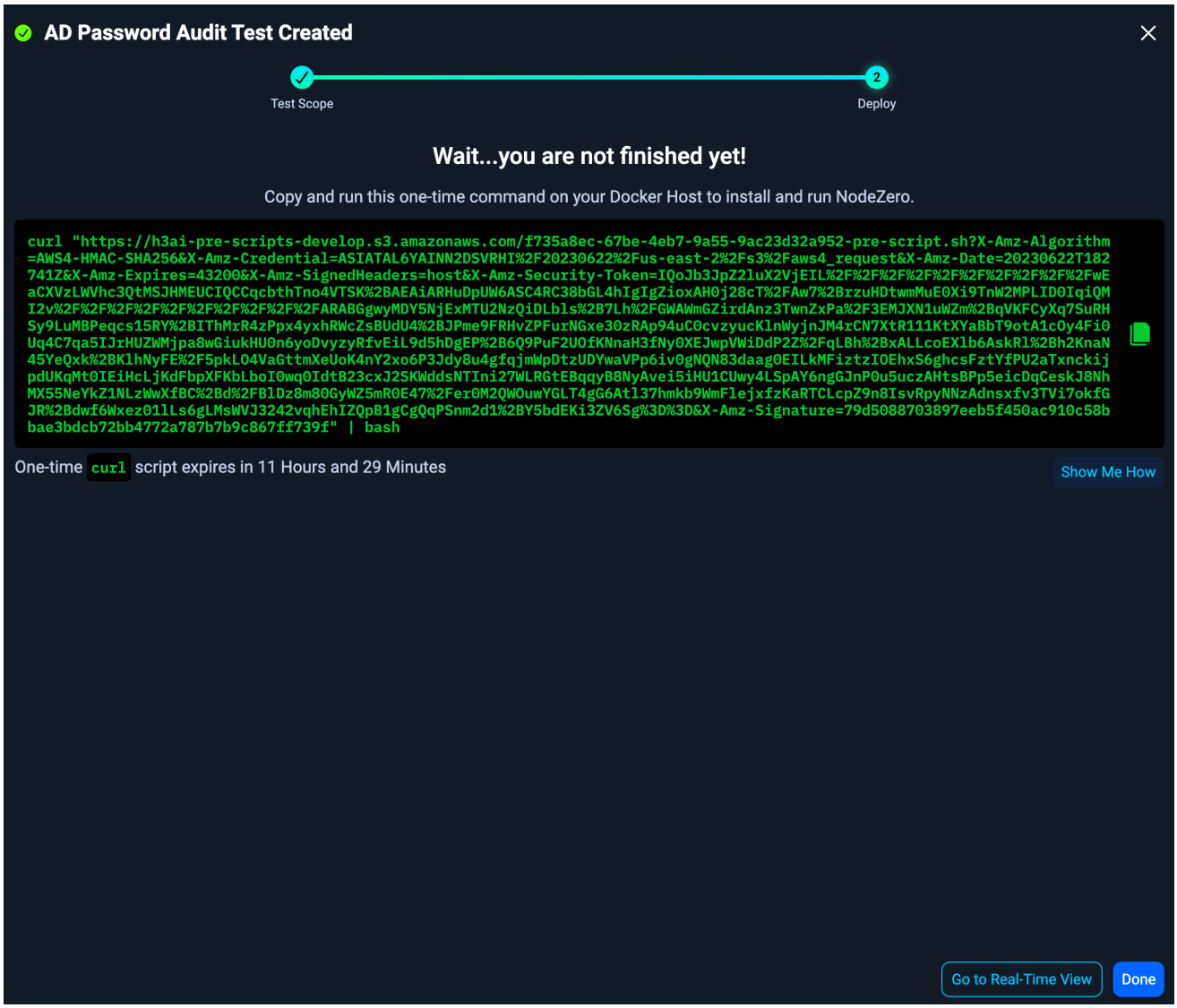
Add Open-Source Intelligence

The Open Source Intelligence step is important for providing NodeZero context about your company - context that an attacker may be able to leverage to guess weak passwords.

NodeZero uses company domains you provide to look up dark web data associated with your company. Company Name(s) and Weak Password Terms are used as an input to password cracking. If you have any well-known weak passwords at your company that you want to weed out, you can add them into the “Weak Password Terms” section to ensure NodeZero attempts them (and variations of them) while cracking.

Deploy NodeZero

When you're done configuring the pentest, scroll to the bottom of the page and click Run Pentest. This creates the test and brings you to the Real-Time View, where you can copy the NodeZero script to run on your host.



Run the curl script on your NodeZero host to kick off the AD Password Audit. Or, if you are using a NodeZero Runner, the audit will kick off automatically on its own.

Don't have a NodeZero host?

See our documentation in Set Up NodeZero Host.

The script will validate the Docker installation, download the most up-to-date NodeZero Docker image, and begin the pentest. In the Portal, you will see the status of the pentest transition from Ready to Running.

Successful launch!

Upon completion of the AD Password Audit, NodeZero sends an email to the user that launched it.

HOW NODEZERO CONDUCTS AN AD PASSWORD AUDIT

User passwords are stored as NTLM hashes within the Active Directory database file, called NTDS.dit, on domain controllers. To conduct an AD Password Audit, the NodeZero Docker container first performs a remote operation

called DCSync to connect to a domain controller and pull these password hashes. NodeZero requires a privileged credential to perform this operation.

User password hashes are transmitted by the NodeZero Docker container to the Horizon3.ai cloud and stored within a secured ephemeral private VPC network for the duration of the audit. The VPC is allocated in real time when the audit is started and destroyed after the audit completes. All sensitive data, including password hashes, are destroyed at the end of the audit.

Password Cracking

During the audit, NodeZero attempts to crack password hashes using a variety of methods:

- ✅hecking for basic commonly used weak passwords
- ✅hecking for passwords that resemble usernames
- ✅hecking for passwords that resemble well-known contextual terms about a company such as a company's name or domain name
- ✅hecking for passwords that match or are similar to passwords in dark web data associated with the company
- ✅hecking for passwords that match or are similar to any known breached passwords
- ✅hecking for passwords that match any user-provided custom password terms

Password Reuse

NodeZero also analyzes the population of passwords for password reuse, i.e., cases where two or more users are using the exact same or similar passwords. Password reuse is important to minimize. When users use similar passwords, attackers can easily take a single account compromise and turn it into a multi-account compromise, potentially gaining more privileges and access to data.

Cracking Methodology

To maximize cracking efficiency as real-world attackers do, NodeZero uses a combination of probabilistic algorithms and curated wordlists whose contents are continually updated. Password cracking results are not static, and might vary between runs.

Reports

After completing the AD Password Audit, you can download the Fix Action Report by clicking the **Download Reports** button in the summary page's upper-right corner.

Viewing Disabled Users

To see disabled users in the AD Password Audit results, go to the "AD Password Audit" page. Click on the 'kebab' menu (three vertical dots) located in the upper-left corner of the results table. From the dropdown menu, select 'View Disabled Users'.

No Passwords were found

If the AD Password Audit fails to display any data upon its completion, verify you have properly configured the injected credential with privileged access. For instructions on configuring the credential, see How to Configure Your Environment.

REFERENCES

1. Metrics That Matter: An Attacker's Perspective on Assessing Password Policy
2. The Undeniable Effectiveness of Password Spray
3. NIST Special Publication 800-63B

3.2.8 Segmentation Test

Segmentation Test

The NodeZero segmentation test scans the internal network for potential targets to run attacks against. This test does not run any attacks; it just scans for IPs, ports, services, and applications on the network.

WHY SHOULD I RUN A NODEZERO SEGMENTATION TEST?

Here are three primary use cases for running a segmentation test:

- See what NodeZero finds in a network before running tests that perform exploits.
- Provide a list of assets on the network at the time the test is run.
- Quick test to familiarize you (or other users) with how results are reported.

The following sections outline how to configure and run the test.

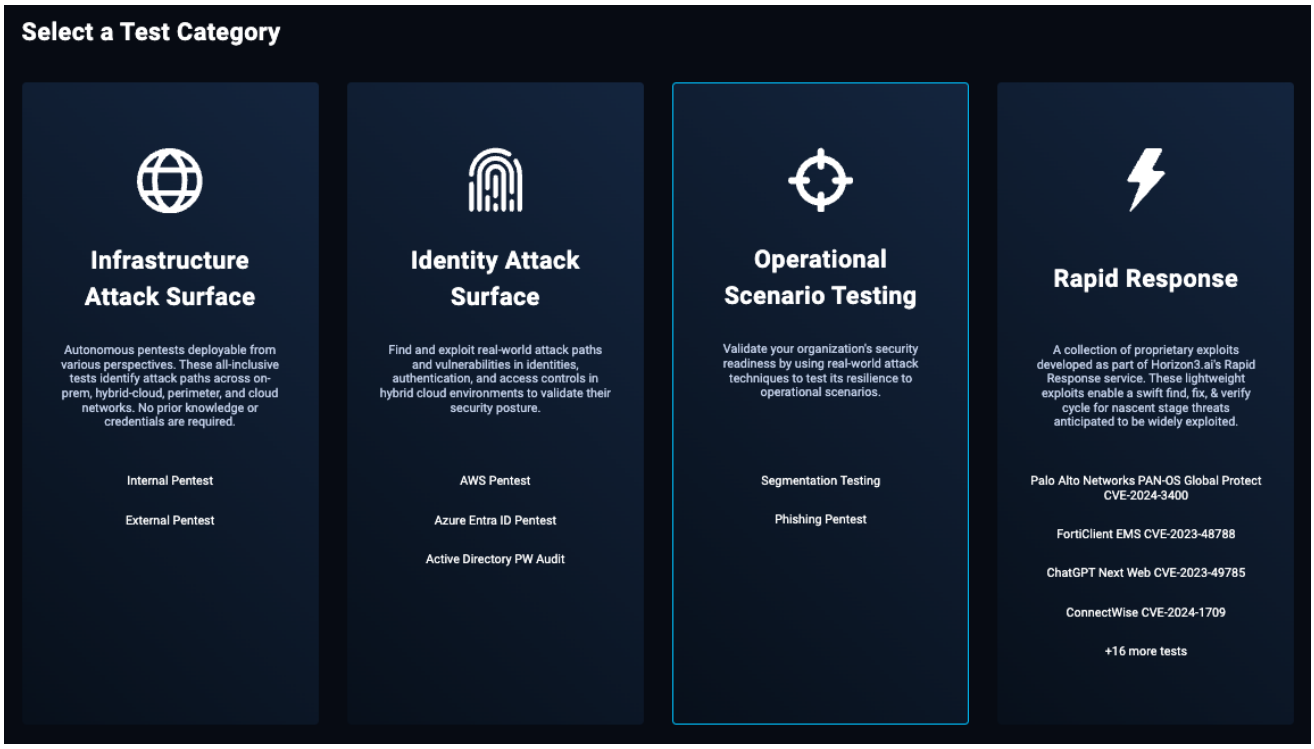
(OPTIONAL) SET UP A RUNNER

Creating a Runner for the segmentation test will enable you to run the test directly from the NodeZero Portal, and will also facilitate running the test repeatedly. For configuration steps, see NodeZero Runner.

CREATE A NEW PENTEST

Open the NodeZero Portal and navigate to the **Pentests** tab.

Click **+ RUN PENTEST**, then select the **Operational Scenario Testing** category.



Select **Segmentation Testing** to open the configuration for this pentest.

CONFIGURE A SEGMENTATION TEST

You can fill out three sections for this test: **Pentest Template**, **Name**, and **Scope**.

Select a Template and Name

Select a **Pentest Template** to use for the test, and give the test a **Name**.



Set a Scope

The scope is the set of IPs and/or subnets (in CIDR notation) within which you want to run the test. This should be set to the network of interest.

If you are unclear on CIDR notation, here is a reference and a calculator app to assist you:

- CIDR Notation Article
- CIDR Calculator

If your environment uses `192.168.0.1` and the subnet mask is `255.255.255.0`, then you'll add the following to the Include section: `192.168.0.0/24`

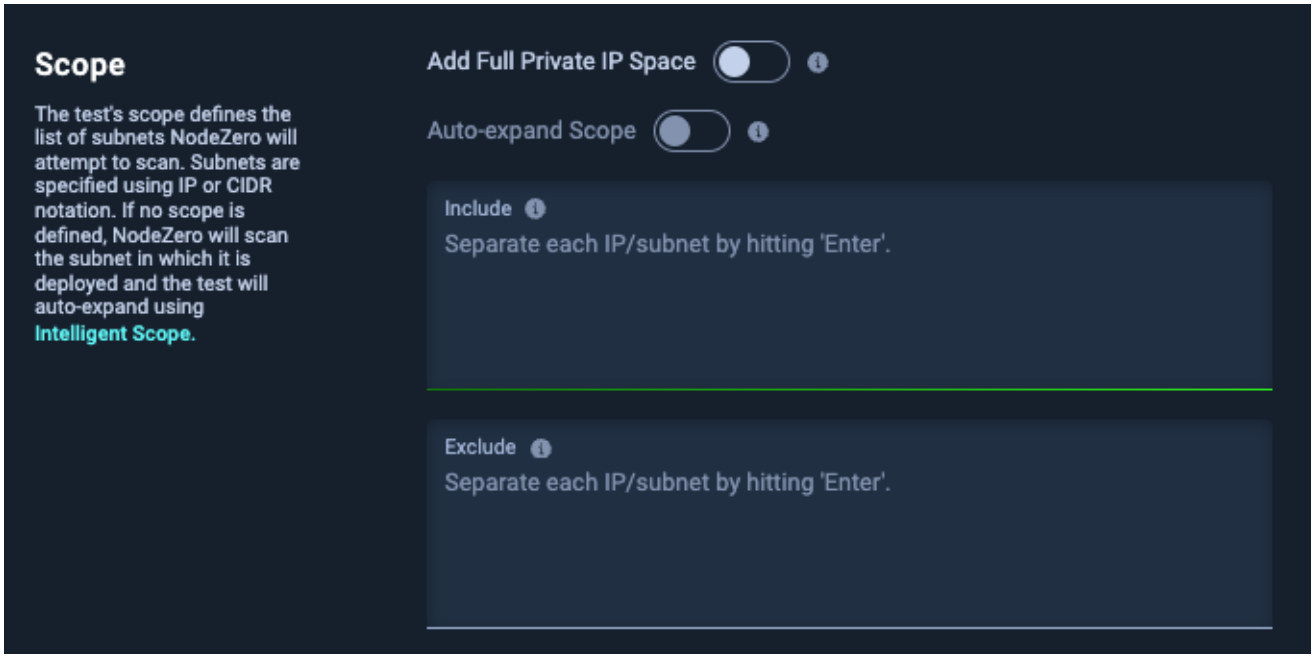
For properly segmented environments, use comma-separated CIDR notation. For example: `192.168.0.0/16,172.16.10.0/24,10.0.0.0/8`

Most internal networks will use the designated private IPs to scan all of these. To automatically add the entire private IP space to the scope, enable the **Add Full Private IP Space** toggle.

If you are running NodeZero in a more complex environment, set the scope to cover as many subnets as possible. You should ask your network administrator for a list of CIDR-notated subnets.

The **Exclude** section stops NodeZero from scanning or exploiting a set of IPs or subnets that you supply (in CIDR notation). NodeZero might still discover these IPs via various techniques within the pentest, but NodeZero will not touch them. Within the pentest results, these IPs might show up in the **Out of Scope** list.

When satisfied with your scope, click **Next**.



(Optional) Select a Runner

To assign the test to a Runner that you've already created, select that Runner from the drop-down shown here.



Run the Test

Finally, click **Run Pentest** to generate the start script to run. If you've assigned a Runner, the Runner will automatically pick up the script and start the test on your behalf. If you're not using a Runner, log in to the NodeZero host, copy/paste the script onto the command line, and execute the script to start the test.

You've started an Segmentation Test

NodeZero sends a notification email once the segmentation test completes.

3.2.9 Phishing Test

Phishing Impact Test

The NodeZero Phishing Impact Test is designed to help you measure and understand the impact of successful phishing attack within your organization, starting with your most phish-prone employees.

WHY SHOULD I RUN A NODEZERO PHISHING IMPACT TEST?

No matter how secure we make our network perimeter, attackers will find ways to exploit the humans that use our networks to gain initial access. Hopefully, your organization provides education and training to employees to reduce the likelihood of successful phishing attacks, but inevitably some slip through the cracks.

So the question is not "what happens if someone gets phished?", but "what happens when someone is phished?".

This is where NodeZero's Phishing Impact Tests come in!

This test is intended to supplement your organization's simulated phishing tools to not just identify who my be susceptible to phishing, but to also understand what happens when they get phished.

NodeZero will automatically capture the credentials of simulated phishing attack victims and use them to pentest your internal network. The report from this test can be use to assess the business risk of a successful phishing attack, and identify security controls that can be put in place to mitigate this risk.

HOW DOES A NODEZERO PHISHING IMPACT TEST WORK?

The Phishing Impact Test works a lot like an Internal Pentest, but enables NodeZero to automatically capture and use credentials captured through your organization's Phishing Simulation Tool.

At a high-level:

- NodeZero is deployed within your organization's network
- NodeZero generates a JavaScript payload to be copied into the landing page for your simulated phishing campaign
- You deploy your simulated phishing campaign, sit back, and wait for a bite
- When a user is phished, the credentials are captured and directly injected into NodeZero's ephemeral infrastructure
- NodeZero uses the phished credentials for perform an authenticated pentest within your network
- NodeZero generates a report to convey the impacts of the phishing attack

HOW TO RUN A NODEZERO PHISHING IMPACT TEST

Before You Begin

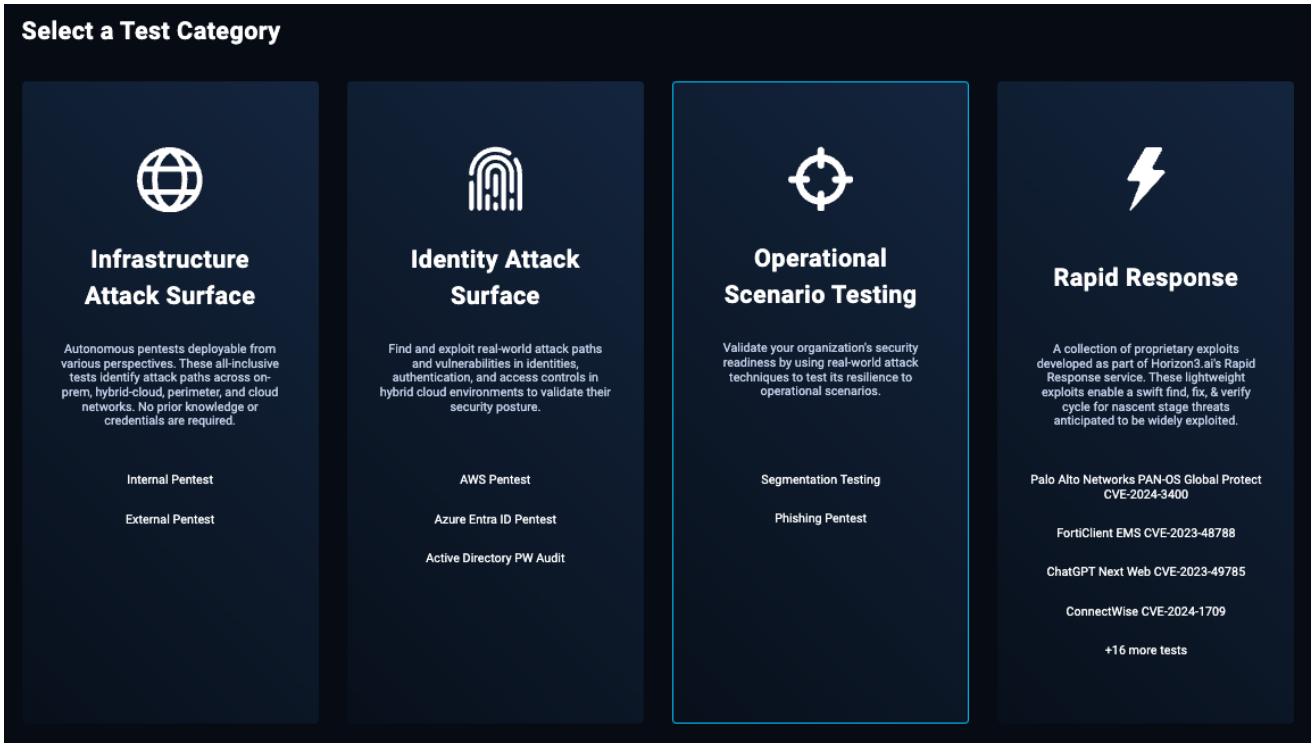
- You will need access to both the NodeZero Portal and your organization's Phishing Simulation Tool (e.g. KnowBe4, ProofPoint, etc.)
- Plan out the details of your phishing campaign
- How long will the campaign last?
- What is the target group for this phishing campaign?
- What type of phishing email(s) am I sending?
- What landing page(s) will the phishing emails link to?

> **Campaign duration**
>
> NodeZero currently supports phishing campaigns of up to 28 days.

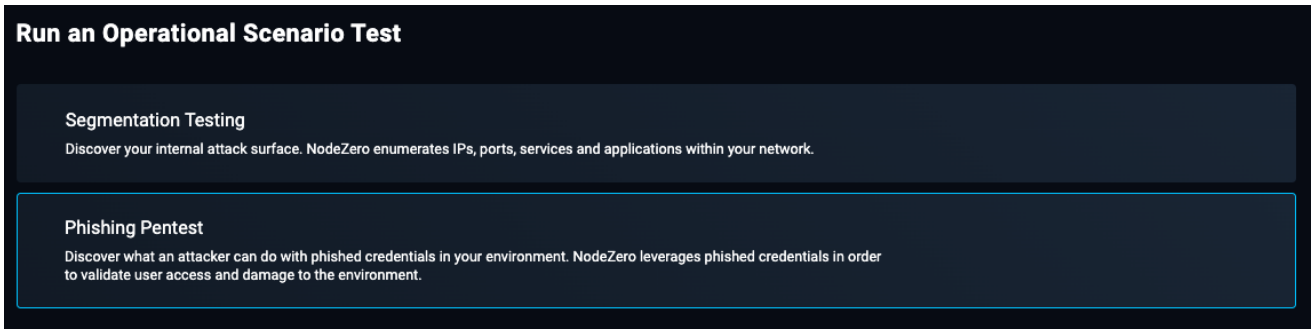
Create a new Pentest

Open the NodeZero Portal and navigate to the Pentests tab.

Click `+ RUN PENTEST`, then select the Operational Scenario Testing category.



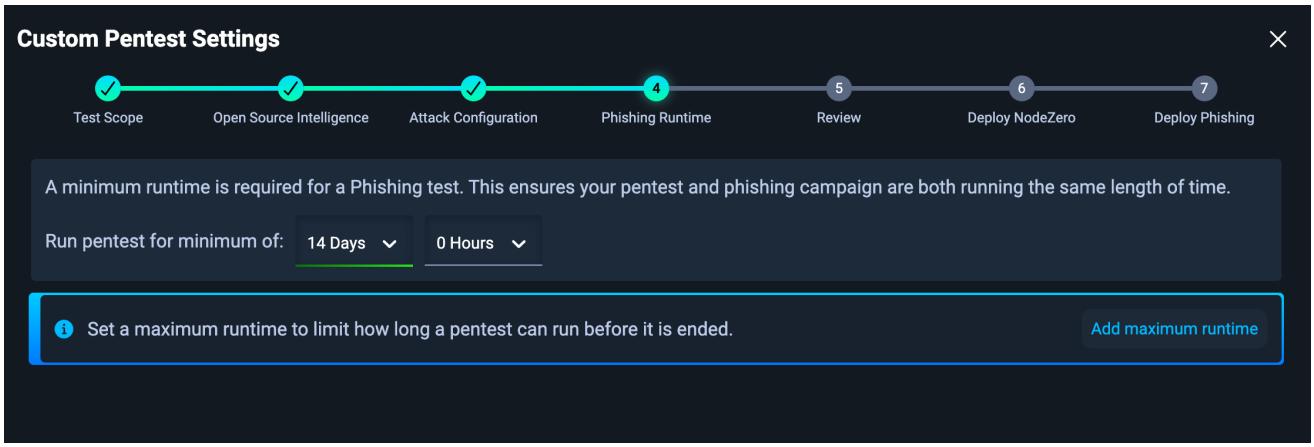
Select `Phishing Pentest` to open the configuration for this pentest.



Configure the Phishing Impact Test

The pentest configuration for a Phishing Impact Test is very similar to an Internal Pentest

A key difference from the Internal Pentest is that you must set a **minimum** phishing campaign runtime. This should be **at least as long** as your phishing campaign within your Phishing Simulation Tool.



Selecting target networks

Consider your phishing targets when configuring your pentest scope. Include networks in which your phishing targets likely have access. Also consider including networks to which they definitely should **not** have access. NodeZero can help validate whether your assumptions and security controls are correct.

Deploy NodeZero

Run the curl script on your NodeZero host to kick off the Phishing Impact Test. Or, if you are using a NodeZero Runner, the test will begin automatically.

Don't have a NodeZero host?

See our documentation in Set Up NodeZero Host.

This script will validate the Docker installation, download the most up-to-date NodeZero Docker image, and begin the test.

Configure the Phishing Landing Page

Copy NodeZero Phishing Script and embed at the bottom of the HTML

The NodeZero Phishing Script should be copied to the bottom of the HTML file right above the closing `</body>` element. In some cases phishing templates are missing the `<body>` tag. In such cases putting the script at the bottom on the HTML will suffice.

The example HTML page below shows where to paste the script, and can be used to try it out yourself.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Horizon3.ai Phishing Impact Test Page</title>
</head>
<body>
<style type="text/css">
  * { box-sizing: border-box; }
  body { font-family: "Helvetica Neue"; font-size: 15px; color: #000; }
  h1 { font-size: 34px; color: #000; font-weight: 200; }
  form { width: 100%; max-width: 356px; margin: 0 auto; display: block; }
  input[type="text"],input[type="password"] { border: 2px solid rgba(0, 0, 0, 0.4); background-color: rgba(255, 255, 255, 0.4); color:
#000; font-size: 15px; padding: 6px 8px; margin-bottom: 12px; }
  input[type="text"]:focus, input[type="password"]:focus { border-color: #0078d7; background-color: #fff; }
  input[type="text"], input[type="password"], input[type="submit"] { width: 100%; }
  #logon { background-color: #0078d7; border: 0; color: #fff; padding: 8px 12px; text-align: center; width: 100%; font-size: 15px; }
  #logon:hover { background-color: #004e8c; cursor: pointer; }
</style>
<center>
  <img alt="Horizon3.ai logo." height="25" src="https://p7i3u3x3.rocketcdn.me/wp-content/uploads/2022/06/
```

```
Horizon3ai_Logo_Tagline_Horizontal_RGB-WhiteTxt.png.webp" style="width: 200px; height: 50px; margin: 2% auto 0 auto; background-color:
#000;" width="100" />
  <h1>Sign in</h1>
  <p>Use your Horizon3.ai account.<br /></p>
  <form action="/pages/19906e20e8a60c71badf8f94151c19a2" id="loginform" method="post">
    <input id="username" name="username" placeholder="Email" required="" type="text" />
    <input id="password" name="password" placeholder="Password" required="" type="password" />
    <input id="logon" name="logon" type="submit" value="Sign in" /> 
  </form>
</center>

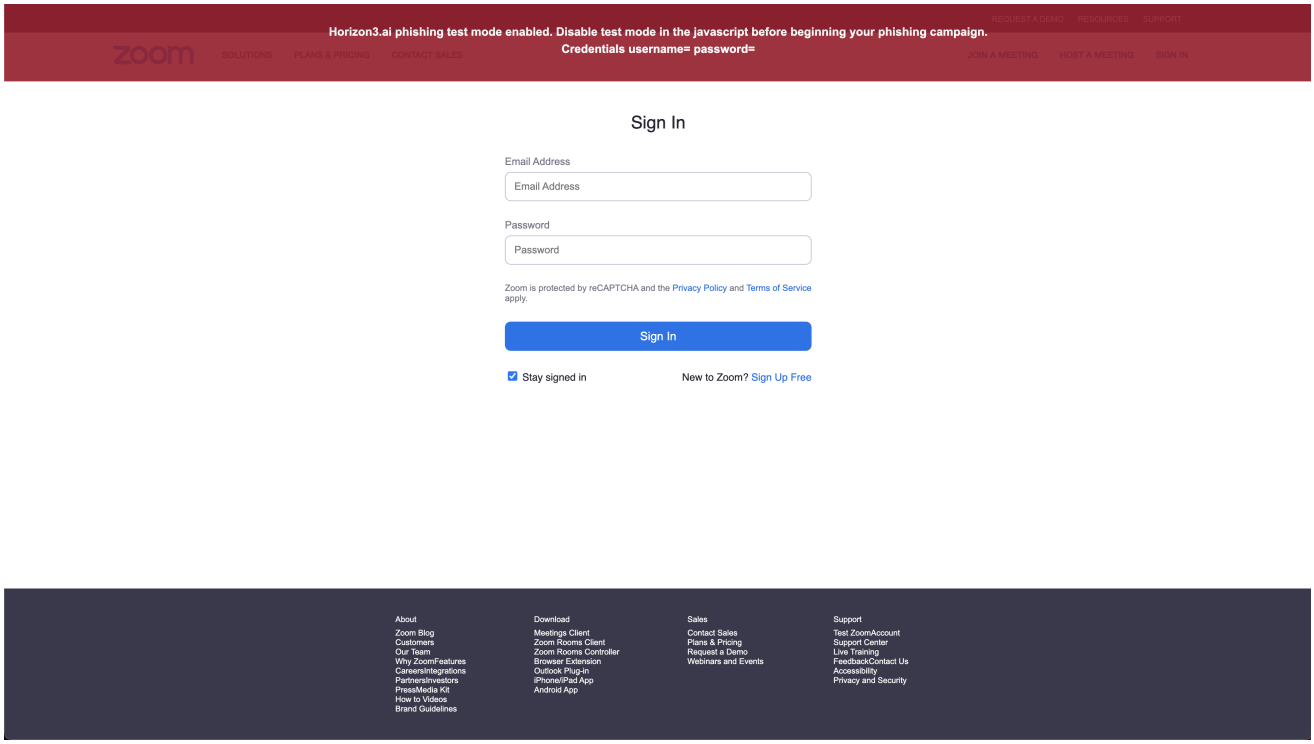
<!-- PASTE NODEZERO PHISHING SCRIPT HERE -->

</body>
</html>
```

Some phishing templates may already contain JavaScript and there could be interference between the scripts. Carefully review any existing scripts to identify conflicts. Remove any unnecessary `<script>` tags in the template. In most cases, the only `<script>` tag should be the NodeZero Phishing Script.

Preview the landing page

Open the landing page to preview and test it. Notice that there is a red banner explaining Horizon3.ai Phishing Impact test mode enabled. If you do not see a red banner check the script for `const testModeEnabled = true`.



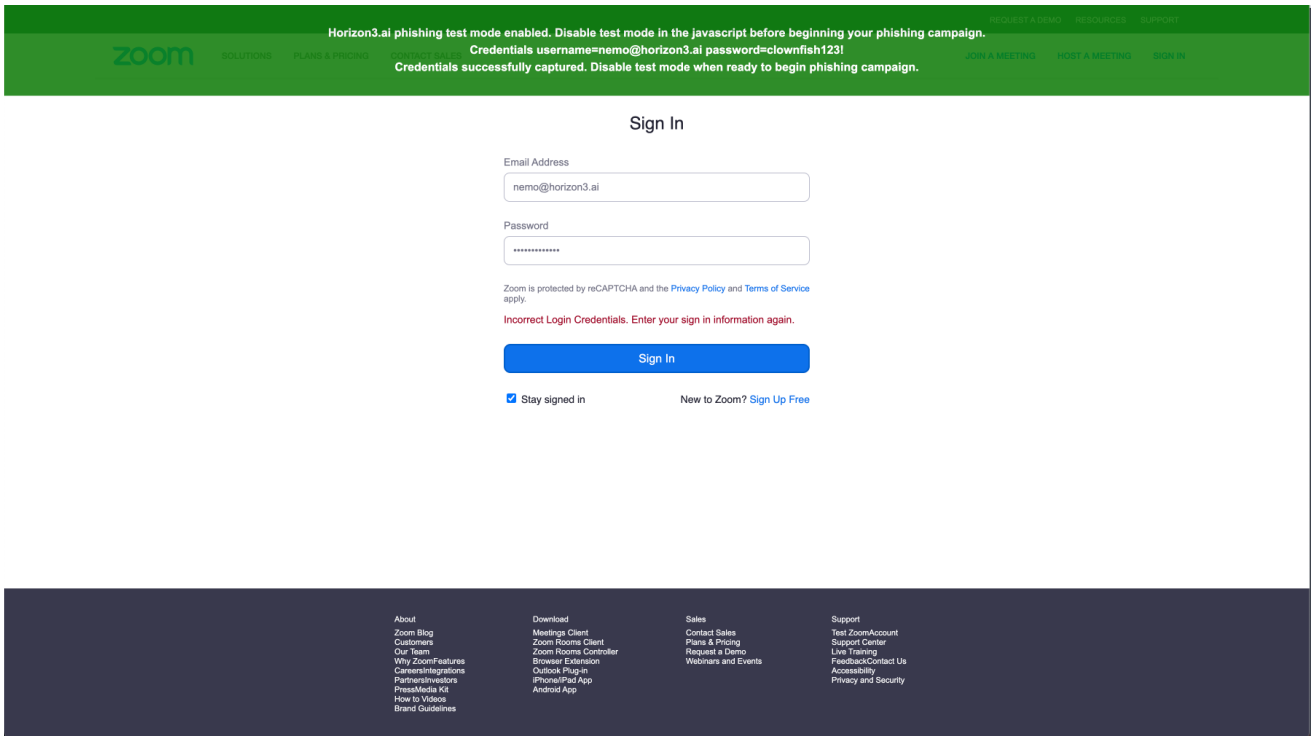
Test the landing page credential detection

From the landing page preview, enter a username and password. The credentials will be reflected to you in the red test banner as you type.

If you do not see credentials when typing into the input fields, then there is an issue with the script identifying the right inputs.

Test the landing page credential submission

From the landing page preview, submit a set of credentials. If the script is working properly the banner will turn green. At this point, you have validated that the script is correctly capturing and submitting credentials to NodeZero.



Additionally, you may notice a red form validation error that reads `Incorrect Login Credentials. Enter your sign in information again.` This is added by the Horizon3.ai phishing script in an attempt to trick the user into submitting another variation of credentials that can also be used during the Pentest. You can disable this feature if you like by changing the following variable to false `const showIncorrectCredentialsError = false`.

Disable test mode

Revisit the landing page source code and change the line

```
const testModeEnabled = true
```

with

```
const testModeEnabled = false
```

Refresh the landing page preview and ensure that the banner is no longer present. If you do not change this variable, then the red banner will show to phished users.

Your landing page is now armed and ready for your phishing campaign.

Having issues with the NodeZero Phishing Script?

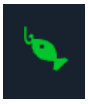
See our documentation in How to use the NodeZero Phishing Script.

Launch your Phishing Campaign

Launch the Phishing Campaign in your Phishing Simulation Tool.

As phished credentials are received, they will appear in Real-Time View and NodeZero will automatically use them.

You can view the status of credentials in the Real-Time View using the icons next to each credential. Credentials states are shown below:

Icon	State	Description
	Pending	The phished credential has been submitted but has not yet been received by NodeZero. This state may occur when the pentest first receives the credential.
	Received	The phished credential has been received by NodeZero. If NodeZero has attempted to use the credential but authentication was unsuccessful, it will remain in the Received state.
	Confirmed	The phished credential has been used by NodeZero to successfully authenticate within the target environment.

FAQ

Is it safe to send phished credentials to NodeZero?

Yes. Horizon3.ai takes the security of phished credentials seriously.
Once phished, credentials are securely transferred to NodeZero's ephemeral environment, which is dedicated to a single pentest and is destroyed once the pentest is completed. Sensitive parts of credentials (e.g. plaintext passwords, hashes, private keys, etc.) are never stored in persistent databases.

How many credentials can I phish in a single Phishing Impact test?

NodeZero supports phishing up to 100 total credentials per Phishing Impact test. Credentials that exceed this limit may show up in the user interface but will not be used by NodeZero. If you find that your Phishing Impact tests are exceeding this limit, consider reducing the number of emails in your phishing campaign or reach out to support to discuss a limit increase.

The NodeZero Phishing Script

Setting up your first phishing pentest requires understanding of the NodeZero Phishing Script, how it works, and what it can do. This page helps answer those questions.

HOW IT WORKS

The NodeZero Phishing Script works by identifying the username and passwords inputs on the login page and submitting them to NodeZero.

The phishing script is currently set up to work on login pages that have a form containing a username/email, password, and a submit button. It does not capture other fields.

Since login pages vary we need a flexible way to capture what input is the username, password, and submit. The script leverages an allowlist of words within the script to automatically find the username, password, and submit inputs. If the script cannot detect these fields then it will require light updates and troubleshooting to the script.

TROUBLESHOOTING THE NODEZERO PHISHING SCRIPT

This section provides tips for troubleshooting problems with the NodeZero Phishing Script. If you have additional issues or questions please reach out to Horizon3.ai support.

The red test banner is not going away

Open the landing page source code and ensure it contains the following line.

```
const testModeEnabled = false
```

If the value is set to `true`, change it to `false`.

If you cannot find the line on the page, regenerate and replace the phishing script into the login page. Then, find the line and change the value from `true` to `false`.

After updating the source code, be sure to press the Refresh button on your browser to reload the page.

Credentials are not showing up on the red banner in test mode

If you notice that the script is not reflecting credentials as you type, then the script is not detecting the form inputs on the landing page. Try the following approach.

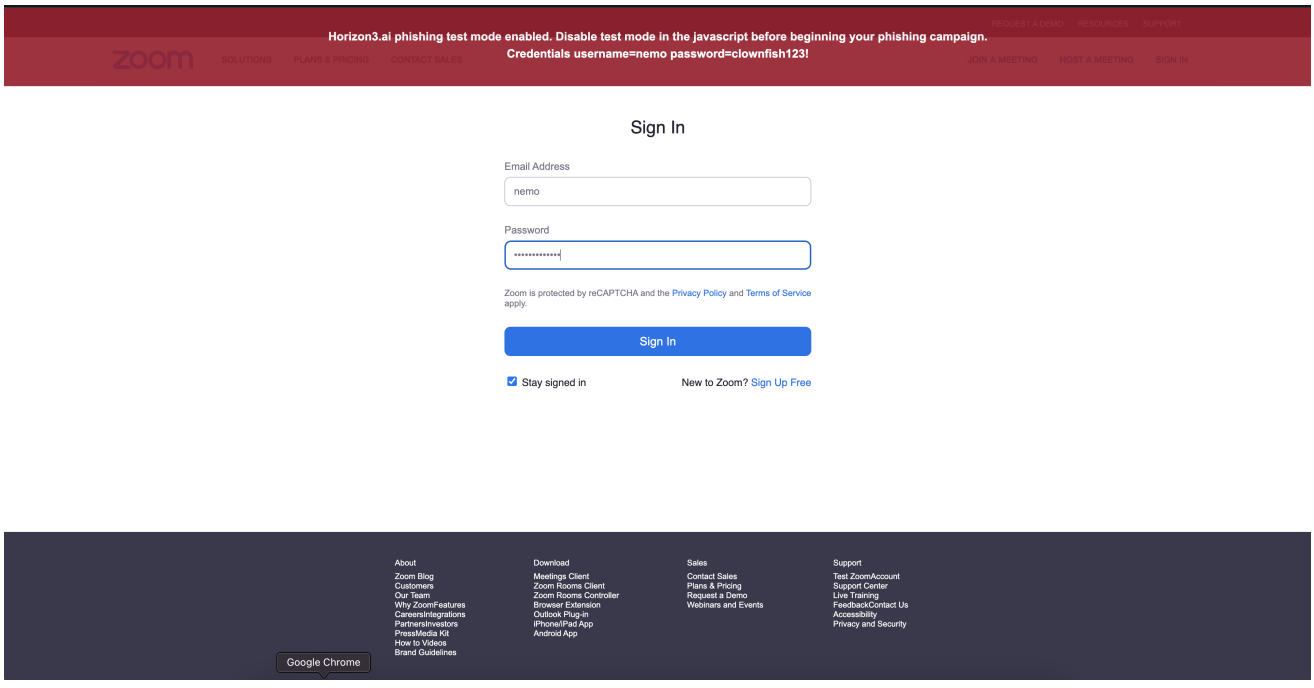
The way the script finds the username, password and submit inputs is with the following wordlists within the script.

```
const lowercaseUsernameTerms = ['email', 'username', 'uname']
const lowercasePasswordTerms = ['password']
const lowercaseSubmitTerms = ['submit', 'sign in', 'log in']
```

Let's imagine that the username field is not being correctly captured. In order to troubleshoot you can open the developer tools in your browser and inspect the input element. In this case the HTML attributes on the input are `user` and our wordlist for `lowercaseUsernameTerms` does not include `user` we can add this item to the wordlist and refresh the page and test again.

```
const lowercaseUsernameTerms = ['email', 'username', 'uname', 'user']
```

Type into the username field and see if the credential reflects back. If so, you're good to proceed on to the next steps.



The banner does not turn green when submitting credentials

If you notice the test mode banner does not turn green and indicate success after submitting credentials, there may be an issue with the API Key used by the script. The generated phishing script is specific to a single phishing test and cannot be reused.

Try regenerating and replacing the phishing script in the login page.

I don't want to show the `Incorrect Login Credentials` message

Open the landing page source code and ensure it contains the following line.

```
const showIncorrectCredentialsError = false
```

If the value is set to `true`, change it to `false`.

If you cannot find the line on the page, regenerate and replace the phishing script into the login page. Then, find the line and change the value from `true` to `false`.

After updating the source code, be sure to press the Refresh button on your browser to reload the page.

3.2.10 Insider Threat Test

Run an Insider Threat Pentest

You can deploy NodeZero to validate your organization's security posture against insider threats.

HOW TO RUN AN INSIDER THREAT PENTEST

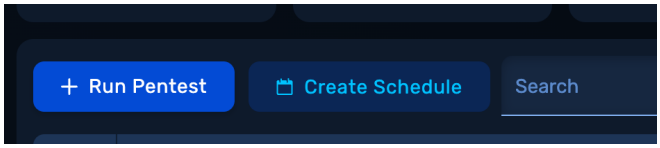
Navigate to Pentests to Run an Insider Threat Pentest

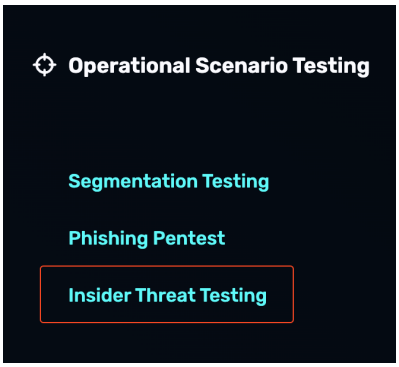
Once you've established a NodeZero Host that meets to requirements, navigate to Pentests to start an insider threat pentest.



Click + Run Pentest

Click **+ Run Pentest** to open Test Categories screen, and from the **Operational Scenario Testing** category, select **Insider Threat Testing**.



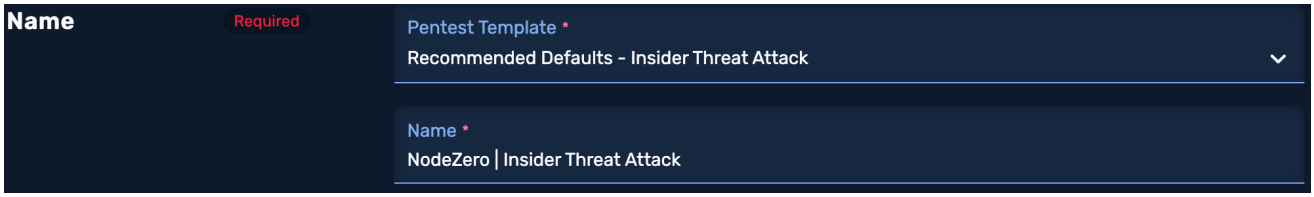


Configure the Insider Threat Pentest

Name the Insider Threat Pentest

Name the Insider Threat Pentest and select a pentest template.

> **Pro Tip: Naming Scheme**
>
> Determine and follow a naming convention to easily identify pentests in your pentest list.
>
> **Example naming convention:** [date]|[library]|[NodeZero Src]|[scope]
>
> **Example following this convention:** 2021-09-01|NodeZero|East-Coast-Bizops|Full: This indicates that the NodeZero host was placed in the East Coast Bizops network and the scope was the entire enterprise.



Select a Scope

The pentest scope is a set of IPs and/or subnets (in CIDR notation) that you want NodeZero to test.

Depending on where your NodeZero host is deployed, reachability to the provided scope can be affected by how much privilege, access, and authentication NodeZero can capture or is given.

Unlike vulnerability tools that scan for device fingerprints and services, NodeZero enumerates devices in scope and then chains together context, exploitable vulnerabilities, misconfiguration, insecure or weak controls, and any data or loot that it captures, to identify attack paths by priority of impact while providing proof. Typically, the larger the scope, the more findings you'll see, as NodeZero has much more to chain together.

There are three options for the Include section (defined below):

- Intelligent Scope
- Full RFC 1918 (192.168.0.0/16, 172.16.10.0/24, 10.0.0.0/8)
- Custom IP(s) or Subnet(s)
- Auto-expand (Optional)

If you are running NodeZero in a more complex environment, we recommend setting the scope to cover as many subnets as possible. Be sure to ask your Network Administrator for a list of CIDR annotated subnets.

The Exclude section stops NodeZero from scanning or exploiting a set of IPs or subnets. The IPs within this section may be discovered by NodeZero via various techniques within the pentest, but NodeZero will not touch them. They may show up in the Out of Scope list within the pentest results. Note that this parameter also requires CIDR notation.

Additional CIDR notation reference and calculator app:

- CIDR Notation Article
- CIDR Calculator

Intelligent Scope - Uses the IP of the NodeZero Host you are deploying the test from and enumerates then tests against the /16 subnet it's a part of. NodeZero then moves into /23 subnets to continue identifying other potential assets in its vicinity, then repeats the cycle until it can't see additional devices. The blast radius varies depending on the level of privilege and access NodeZero captures or is given.

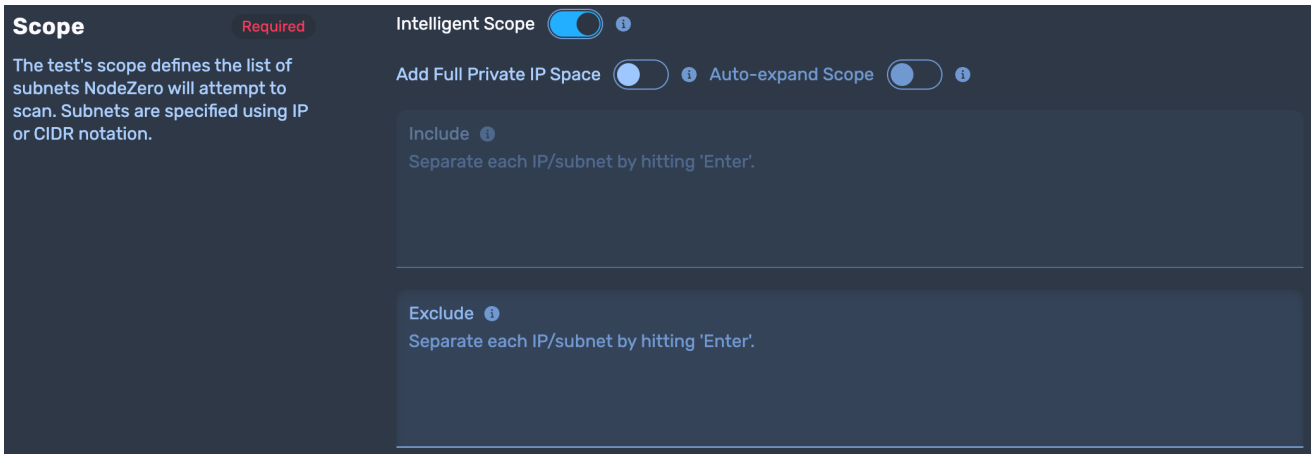
> **Pro Tip: Intelligent Scope**
>
> Great for scenario testing – what can an attacker see and do from the position of my NodeZero host?

Full RFC 1918 (192.168.0.0/16, 172.16.10.0/24, 10.0.0.0/8) - The most thorough approach as it explicitly instructs NodeZero to attempt to enumerate and test every IP in the private IP range.

> **Pro Tip: Full RFC 1918**
>
> Great for ensuring complete coverage – what exists in my infrastructure today, and what are its priorities from an attacker's perspective?

Custom IP(s) or Subnet(s) - Provide any IP(s) or subnet(s) in CIDR notation for NodeZero to test from its position. The option to auto-expand allows NodeZero to include in its test any additional devices that it's able to enumerate While NodeZero can be scoped to singular IPs, it's recommended to cast wider nets to maximize coverage and identify potential vulnerabilities across the network more effectively.

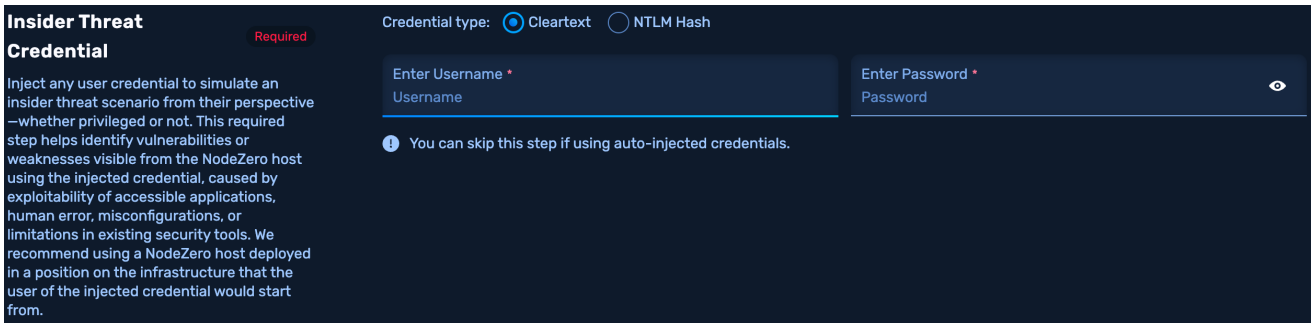
> **Pro Tip: Custom IP(s) or Subnet(s)**
>
> Great for testing small scopes and validating controls – what could an attacker chain together in the scope of focus and, if auto-expand is turned on, how could they pivot out to other segments?



When satisfied with your scope, scroll down.

Insider Threat Credential

Inject any user credential to simulate an insider threat scenario from their perspective - whether privileged or not. This required step helps identify vulnerabilities or weaknesses visible from the NodeZero host using the injected credential, caused by exploitability of accessible applications, human error, misconfigurations, or limitations in existing security tools. We recommend using a NodeZero host deployed in a position on the infrastructure that the user of the injected credential would start from.

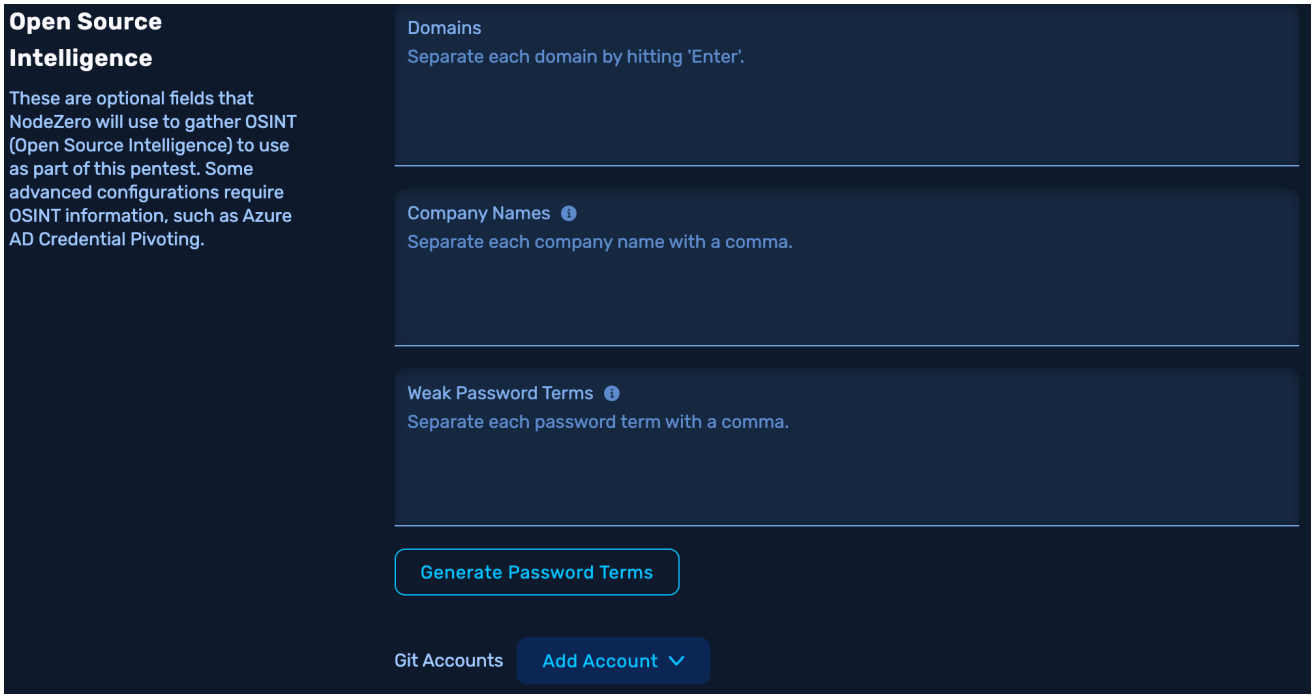


AWS Accounts

Optionally add AWS accounts here. All cloud resources under these accounts will be treated as in scope.

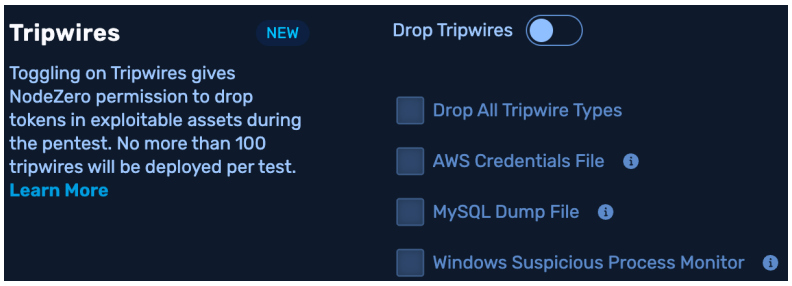
Open Source Intelligence

Optionally add Domains, Company Names, Weak Password Terms, or Git accounts



Tripwires

If your organization has purchased Tripwires, use this option to allow NodeZero permission to drop tokens in exploitable assets during the pentest. No more than 100 tripwires will be deployed per test.

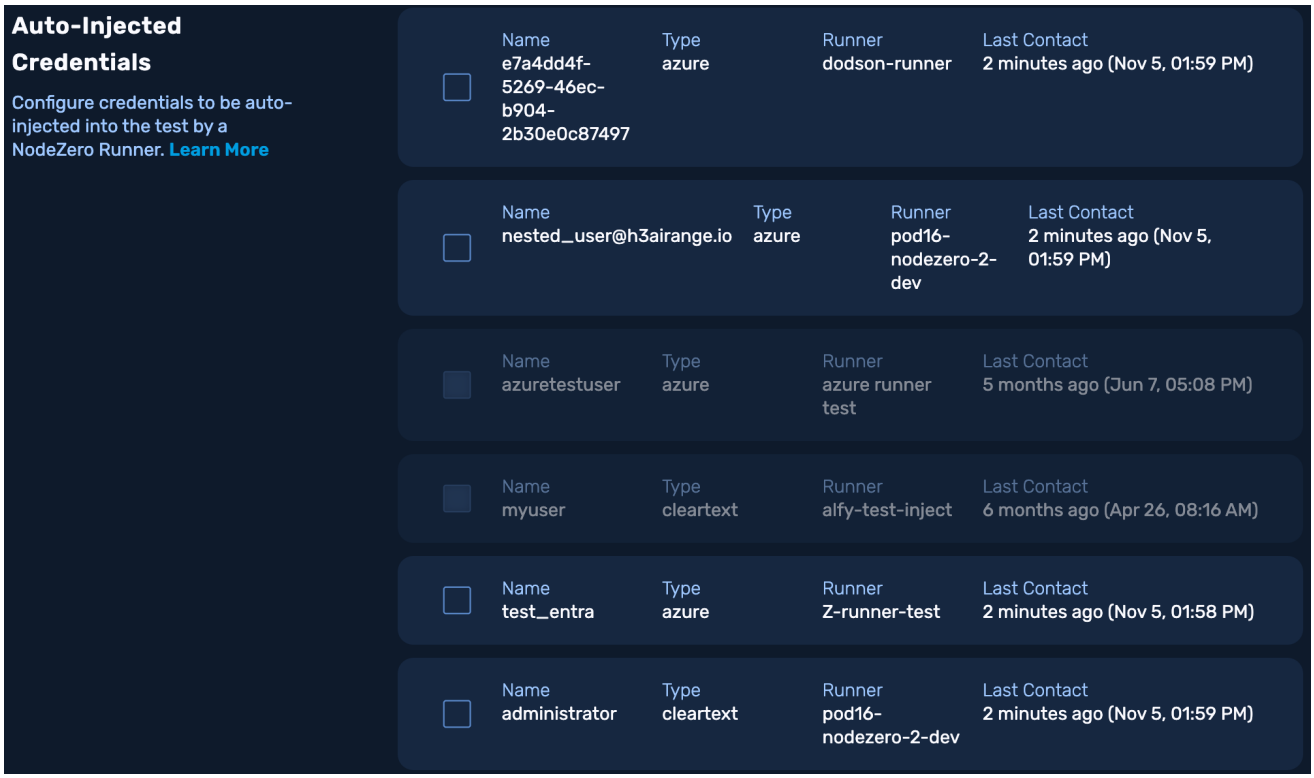


Attack Configuration

Select the types of services and vulnerabilities NodeZero will attempt to enumerate and exploit. For more information on choosing attack configuration options, visit the attack configuration options page.

Auto-Injected Credentials

Configure credentials to be auto-injected into the test by a NodeZero Runner.

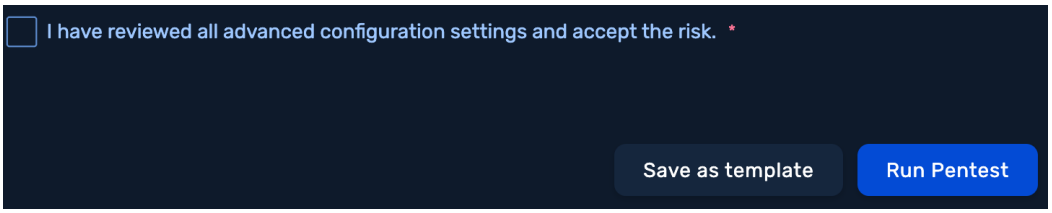


Runner

Use a NodeZero Runner to automatically deploy NodeZero on your Docker host.

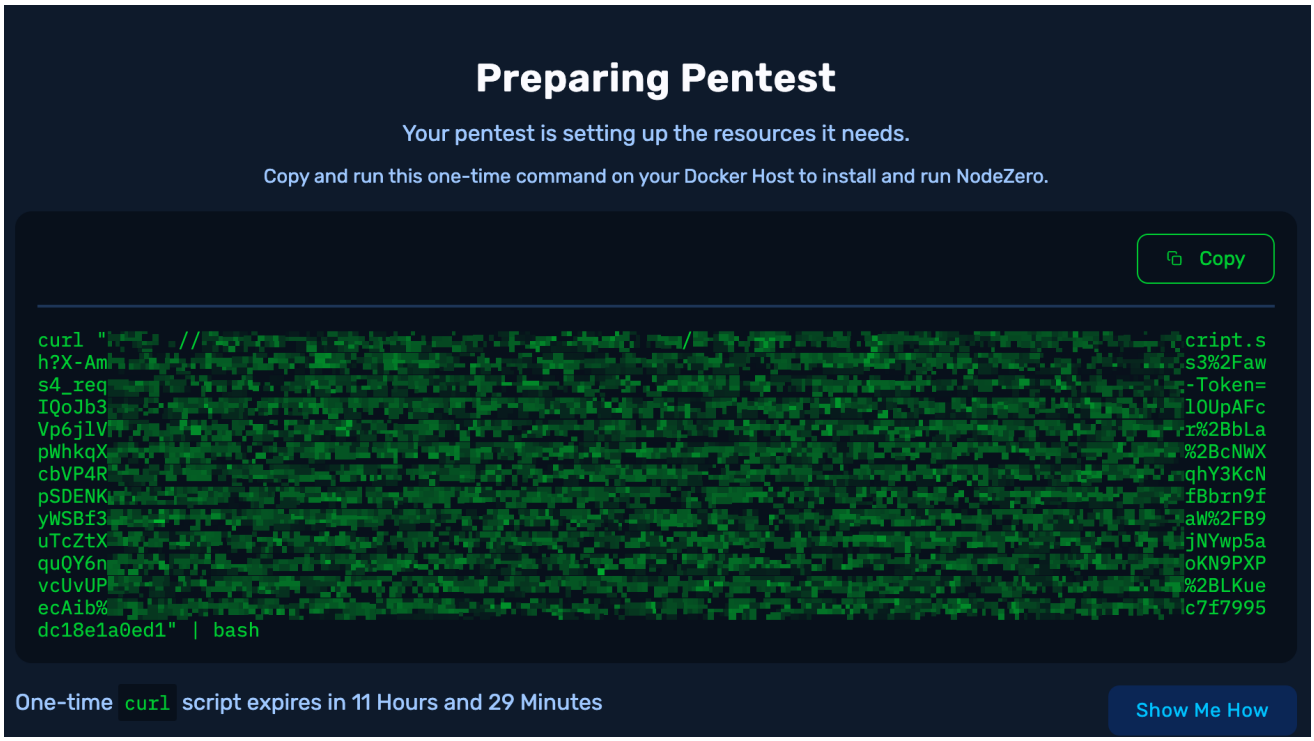
Review the Insider Threat Pentest Configuration

Once satisfied with your pentest selections, check the box to indicate you've reviewed all advanced configuration settings. Then click **Run Pentest** to generate your curl script to launch your pentest.

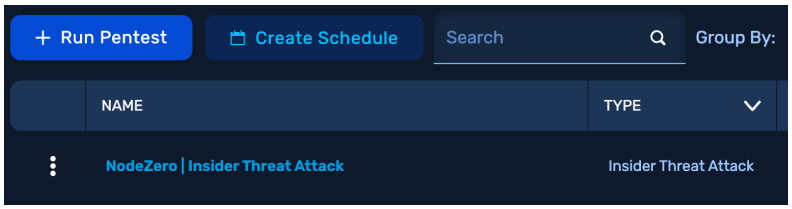


Deploy NodeZero

While your pentest provisions and the ephemeral architecture is being spun up, copy and paste the one-time use bash script into your chosen NodeZero Host. If you selected a Runner, you can skip this step and NodeZero will automatically deploy the test once it's ready.



Real-Time View will show up. From here, you can inject credentials and monitor pentest progress.



You've started an Insider Threat Pentest

NodeZero sends an email once the Pentest completes.

Tip

NodeZero can also run pentests from an authenticated perspective. Go to the Real-Time View and inject credentials to see the impact an attacker would have by leveraging compromised credentials!

3.2.11 1-Click Verify

Run 1-Click Verify

1-Click-Verify enables you to quickly and conveniently schedule a pentest to verify that specific weaknesses have been remediated. This supports, and helps to facilitate, the Hack-Fix-Verify-Repeat loop.

> **Compatible test types**
>
> The 1-Click Verify option is available only on results from Internal, Phishing, and Insider Threat tests.

HOW DOES IT WORK?

1. Select the Weaknesses

Under the **Weaknesses** tab, we've added a 1-Click-Verify functionality to all weaknesses that are 1-Click verifiable. Note that not all weaknesses are 1-Click verifiable. 1-Click verifiable weaknesses are those that can be reliably verified even when the pentest scope is limited to only a single host or set of hosts affected by the weakness. Weaknesses that require complex chains involving multiple hosts and credentials would not be considered 1-Click verifiable.

If a weakness is 1-Click verifiable, you will see a check box on its row. Select the check boxes for all the 1-Click verifiable weaknesses that you want to include in your 1-Click Verify run.



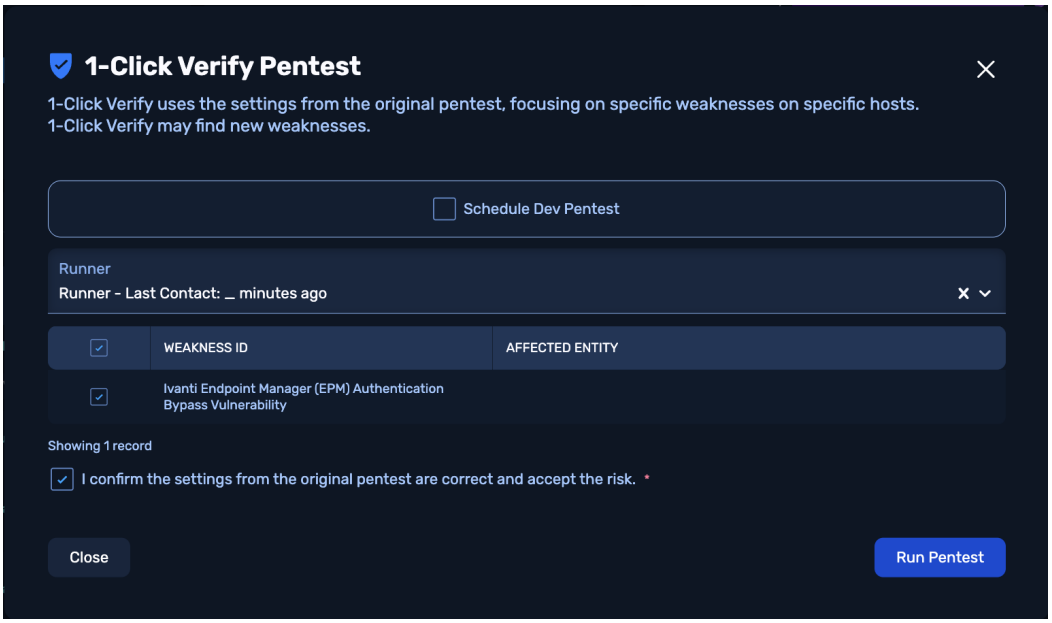
2. Verify the Weaknesses

Once you have selected all the weaknesses you would like to test, click `Take Action` and select `1-Click Verify` from the submenu.

3. Run 1-Click Verify Pentest

In the resulting modal, you can remove any previously added weaknesses, select the Runner you want to use for the test, and acknowledge the configuration settings. All settings are carried over from the original pentest that discovered these weaknesses.

Once you are ready to kick off the 1-Click Verify test, select the confirmation check box, then click the `Run Pentest` button.



4. Check the Results

The newly scheduled 1-Click Verify pentest appears in the Pentests list with its own record. Its name defaults to `1-click verify for weakness CVE-XXXX-XXXX`.

When the 1-Click Verify pentest completes, the tested weaknesses should no longer appear in the results. In most cases, this indicates that remediation was successful. However, from a testing perspective, the only definitive statement we can make is that NodeZero did not rediscover the weakness during this verification run.

WHY RUN A 1-CLICK VERIFY TEST?

A common usage pattern is to first run a full-scope pentest across your environment to identify weaknesses across many hosts. Remediation work then begins with the highest-priority issues. When you are ready to validate specific fixes, 1-Click Verify allows you to narrow the scope of a follow-up pentest to only the hosts associated with the selected weaknesses. Because of this smaller scope, the verification pentest typically completes significantly faster than the original full-scope test. This enables rapidly confirmation of your remediation results, supporting an efficient Find–Fix–Verify cycle.

WHAT PROBLEM DOES IT SOLVE?

Rapid, targeted verification of your remediation efforts enables you to make incremental and verifiable progress toward the goal of eliminating weaknesses and improving the security posture of your network.

1-Click Verify can be a fast path to verifying and documenting the corrections you've made during a compliance audit, such as a PCI DSS (Payment Card Industry Data Security Standard) audit. Once you've verified that your remediation is complete, simply download the 1-Click Verify report as evidence to submit to your assessor.

3.2.12 Real-Time View

Real-Time View (RTV)

NodeZero's Real-Time View provides live monitoring of each pentest's progress and impact – what NodeZero is doing, why it matters, and how each step contributes to risk discovery. Within the NodeZero Portal, RTV provides a dynamic view of events as the test unfolds, in graphical and tabular formats.

These portal views of the event log enable you to correlate NodeZero test events against your SIEM or SOAR, to distinguish and deconflict NodeZero's benign testing from malicious attacks.

ACTION LOG TIMELINE & TABLE

In the default view, the **Action Log Timeline** provides a dynamic view of pentest events over time (UTC). The table at the bottom summarizes corresponding event metadata. A button at the top right enables you to Inject Credentials as the test proceeds.

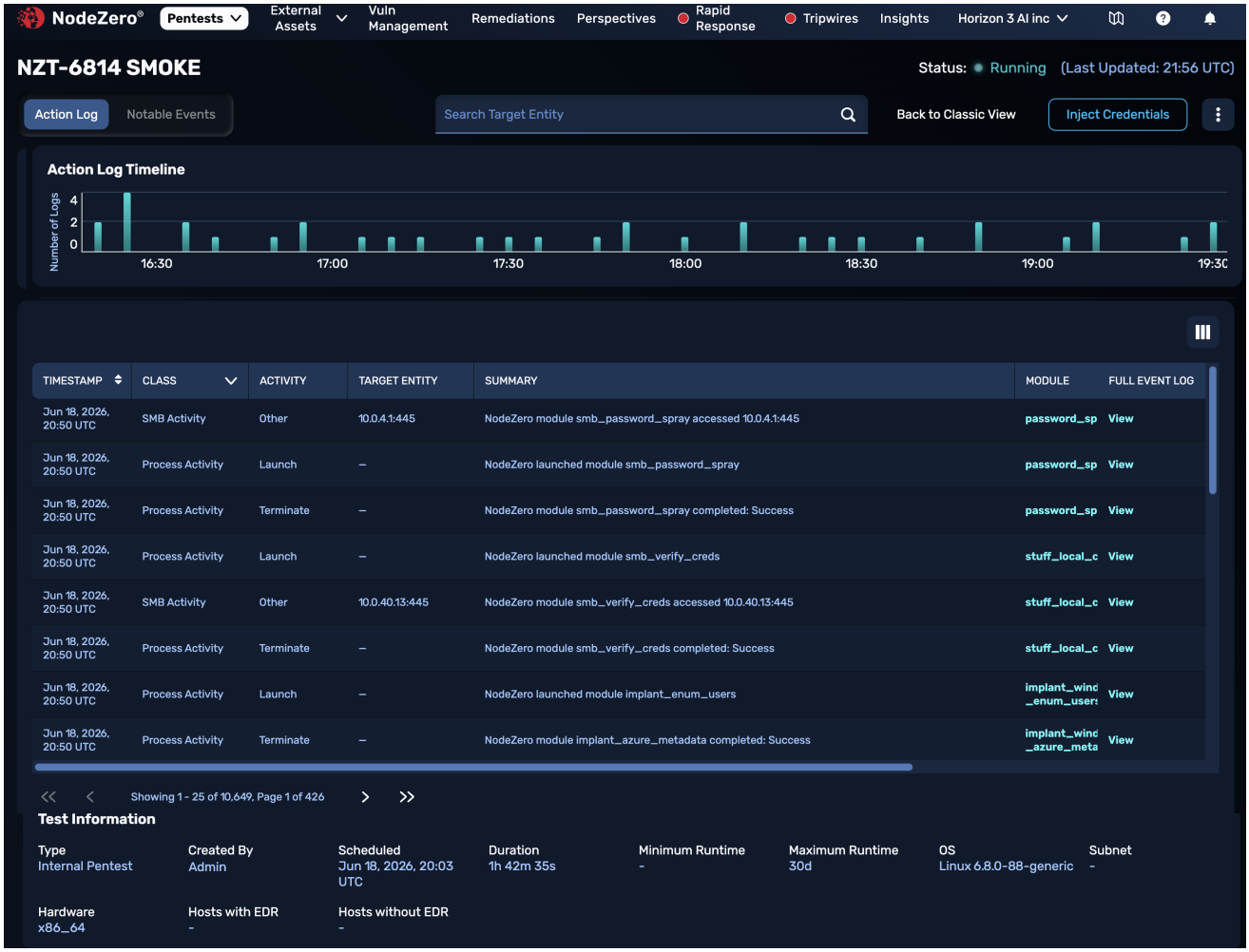


Table Metadata

For each event, the table shows the timestamp, type of activity, module targeted, status, severity, and a summary of the test action. You can sort by timestamp, and you can toggle columns off to simplify the display.

Filtering the Display

The upper search bar provides the option to filter the display by Target Entity. For example, you can paste in an IP address (or IP substring) of interest copied from your security tool, and check whether it matches against a NodeZero test event. Both the timeline and the table will condense to show only matching events.

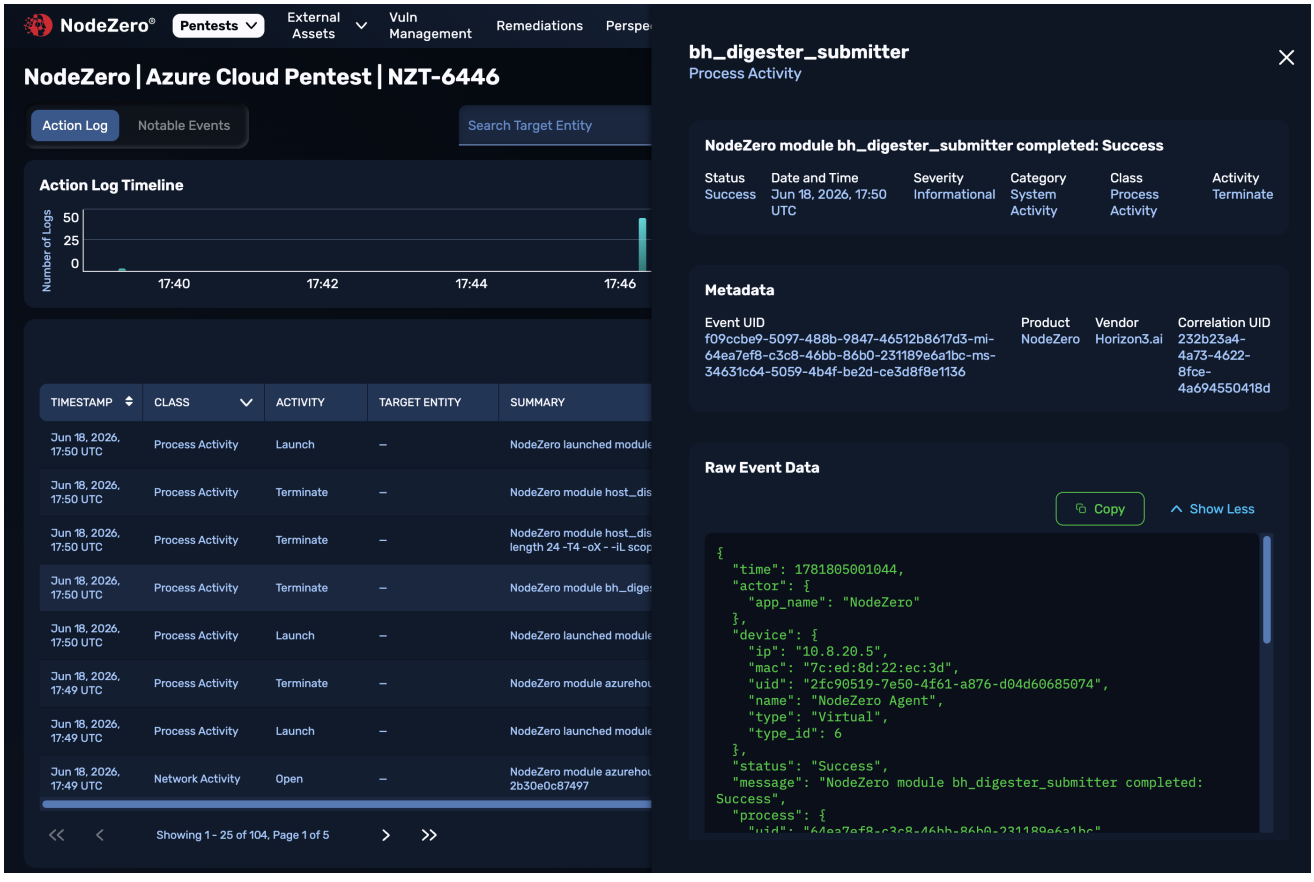
In the upper timeline, you can click and drag to set time boundaries of interest, and the table will refresh to display only events that occurred within those times. To remove your time restrictions, click the ⊗ close box below the timeline to restore the full 8-hour display.

In the table's **MODULE** column, you can click a module of interest to narrow the table's display to only events associated with that instance of the selected module. To restore the display of all events, click the ⊗ close box above the table.

EVENT LOG DETAILS

In the **FULL EVENT LOG** column, click **View** beside any event to open a drawer with detailed metadata about that event. Basic information includes the module accessed, product, vendor, timestamp, severity, activity, Event UID, and Correlation UID.

The **Raw Event Data** panel displays event details in OCSF (Open Cybersecurity Schema Framework) format. Use the **Show More** button to expand the display. You can use the **Copy** button to copy the whole blob, in order to feed it as an artifact into your other security tools.



TEST INFORMATION

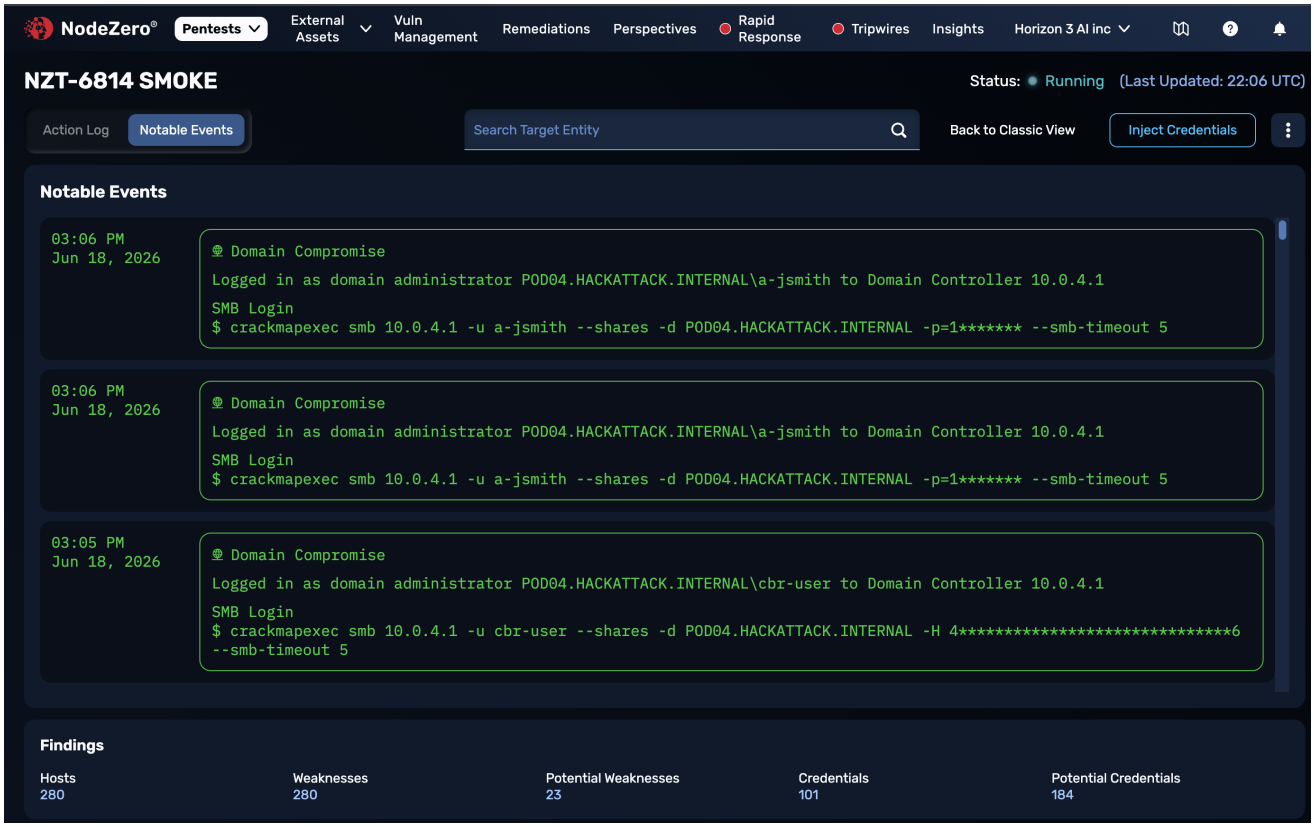
Below the table is a concise summary of metadata for the whole pentest.

NOTABLE EVENTS

At the upper left, click the **Notable Events** tab to replace the timeline and raw logs with a focused display of high-impact events from the running test. This "so what?" view isolates key events like credential discovery, host compromise, lateral movement, privilege escalation, and domain compromise.

Here, you can quickly understand what was compromised, to facilitate fast triage, mid-test actionability, and deconfliction. Use these findings to drill into the full Action Log for audit details.

The **Findings** panel at the bottom of this view displays counts of Hosts, Weaknesses, Potential Weaknesses, Credentials, and Potential Credentials discovered during the test.



3.3 Features

3.3.1 Portal Features

Explore the powerful features of the NodeZero Portal, designed to enhance your security operations and streamline vulnerability testing. These tools help you optimize your security posture and improve efficiency.

Vulnerability Management Hub

Powerful set of tools designed to help security teams seamlessly Fix and Verify weaknesses uncovered during autonomous pentests.

MCP Server

Supercharge your AI with the attacker's perspective using the NodeZero MCP Server

Attack Configuration

Customize the behavior of pentests to match your environment and testing needs.

Bloodhound

Gain insights from Active Directory data to better understand security risks.

Injecting Credentials

Execute pentests from an authenticated perspective to simulate insider threats.

Schedules

Automate pentest scheduling for consistent security assessments in your environment.

Pentest Templates

Set up and manage inputs for your pentests, streamlining the testing process.

Endpoint Security Effectiveness

Gain visibility into your Endpoint Detection and Response (EDR) tool performance.

Campaigns

Organize your pentest results and see from the attacker's perspective.

Back to Portal home

3.3.2 Vulnerability Management Hub

Vulnerability Management Hub

NodeZero's **Vulnerability Management Hub** is a powerful set of tools designed to help security teams seamlessly **Fix** and **Verify** weaknesses uncovered during autonomous pentests. As part of the FFV (Find, Fix, Verify) 2.0 initiative, the Remediations Hub simplifies workflows, improves clarity, and supports automation across vulnerability lifecycle management.

OVERVIEW

The Vulnerability Management Hub introduces capabilities specifically built for **Fixers** and **Verifiers** to:

- Prioritize and take action on high-risk weaknesses.
- Track remediation and verification progress.
- Integrate with existing ticketing and workflow systems (e.g., ServiceNow, Jira).
- Provide audit-ready evidence of mitigation or remediation.
- Maintain an operational backlog that distinguishes between system observations and user decisions.

KEY FEATURES

Vulnerability Management Page

A centralized view that organizes all discovered weaknesses and their current fix/verify status. This serves as the main workspace for Fixers and Verifiers.

- System status versus user status: In separate, dedicated columns, view what NodeZero observed (the **STATUS** column) alongside what your team decided (the **MARKED AS** column).
- **Last Mitigated:** A column that displays the date a weakness was last confirmed as mitigated, if available. The date links directly to the pentest that confirmed the mitigation, so your team can quickly trace when remediation took effect and review the verification evidence. This is especially useful for tracking remediation timelines and understanding how quickly issues are being resolved after a fix or 1-Click Verify run.
- Filter and sort by severity, asset, status, or remediation type.
- See grouped weaknesses tied to common fixes (e.g., 1-Click Verify).
- Inline actions: add notes, dispatch tickets, mark as verified, or close items.



Vulnerability Management Hub dashboard, with system versus user status indicators

---

Weakness State Definitions

NodeZero tracks the lifecycle of vulnerabilities across pentests using standardized states. These help your team understand whether issues are actively present, have been resolved, or have re-emerged due to configuration drift.

System Status (Observation-Based)

NodeZero automatically assigns system statuses based on what it observes during a pentest.

- **Open:** A weakness that is present as of the most recent pentest. This includes newly discovered vulnerabilities and unresolved issues from previous tests.
- **Mitigated:** A weakness that was previously open but is no longer detected, while the associated asset is still present. This indicates the issue was resolved (e.g., patched or hardened) without removing the host.
- **No Longer Found:** An explicit status for weaknesses previously detected but no longer observed. This applies when the asset (or root asset) is found, but the weakness is not.

> **Tip**
>
> This means "not observed," not necessarily "proven fixed." It allows for a clean workflow without overstating confidence.

- **Regressed:** A weakness that was previously mitigated but has reappeared in a later pentest.

> **Important**
>
> For a weakness to be marked as "Regressed," it must have appeared in at least **three** pentests:
> 1. Initially discovered.
> 2. Marked as mitigated in a later test.
> 3. Detected again in a subsequent test on the same asset.

The determination is **not limited to the same pentest template**. What matters is that the weakness reappears in scope (e.g., same host and port) in a future test.

User Status (Workflow-Based)

User statuses allow customers to manage their backlog based on internal decisions.

- **Compensating Control** The user has mitigated the risk through alternative controls rather than fixing the underlying issue directly.
- **Verify Compensating Control** The user has declared a compensating control and is awaiting verification that it sufficiently mitigates the risk.
- **Fixed** The user has remediated the issue and believes the underlying weakness is no longer present.
- **Mute** The user has chosen to hide the issue from active views without resolving or accepting the risk.
- **Potential False Positive** The user believes the finding may not be valid and is investigating whether the issue actually exists.
- **Risk Accepted** The user has reviewed the issue and explicitly accepted the risk without taking remediation action.
- **To Do** The user has acknowledged the issue and plans to address it, but has not yet taken action.
- **Closed:** A user-applied status to remove items from the active queue. This is intended for cases where NodeZero cannot confidently assign "No Longer Found," such as:
- Assets decommissioned or permanently out of scope.
- Network or attack configuration changes that prevent re-testing.
- Programmatic closure decisions based on internal policy.

Why Status Separation Matters

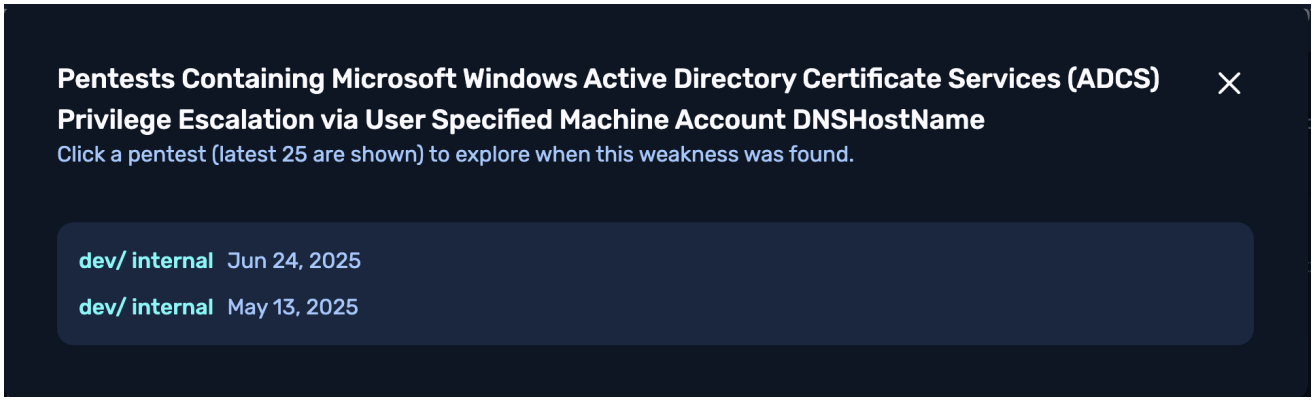
By maintaining **System Status** and **User Status** as separate columns, NodeZero prevents the conflation of attacker observations with workflow decisions.

1. **Operational Focus:** The VMH behaves like a real operational backlog, keeping focus on what still needs action.
2. **Observation Honesty:** We maintain transparency about what was actually seen versus what was manually moved through the workflow.
3. **Audit Integrity:** Even if a weakness is "Closed" or "No Longer Found," NodeZero preserves the full history and reporting access for compliance and traceability.

Asset Detail Modal

Clicking on an asset opens a modal view with:

- Associated weaknesses by risk level.
- Status of remediation/verification.
- Related notes or actions.



Asset Detail modal

Weakness Detail Modal

Dive deeper into a specific weakness with:

- Exploitation evidence and contextual attack path details.
- Clear remediation guidance (e.g., patch, config, mitigation).
- Verification status and fix history.

Impact Detail Modal

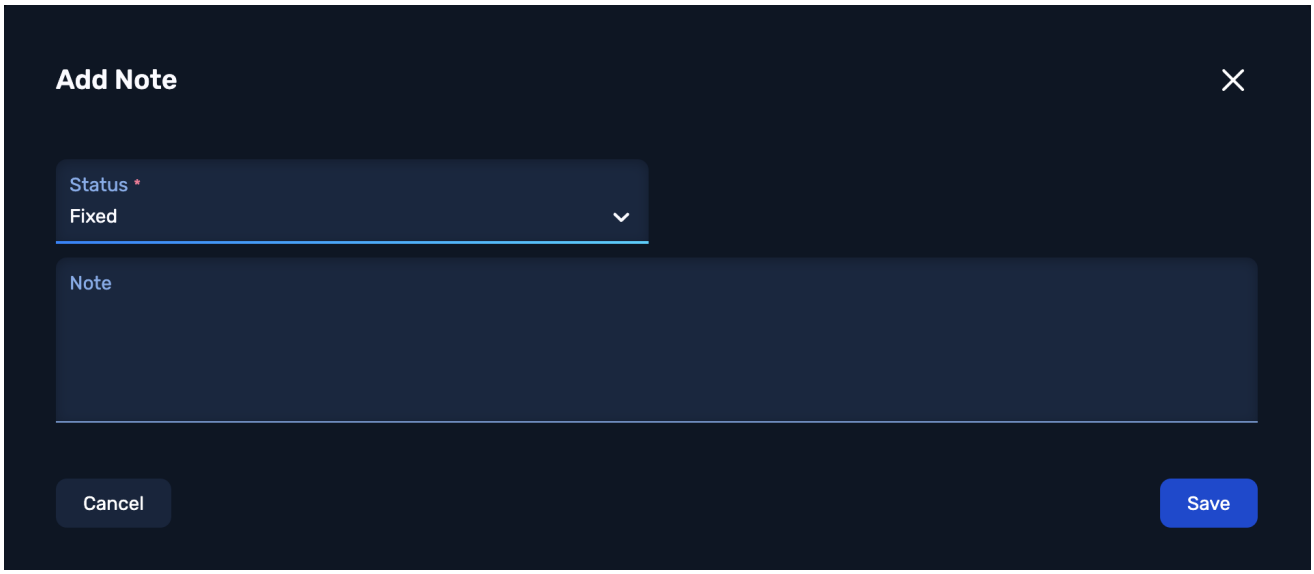
Understand the broader impact of unresolved weaknesses:

- What could happen if the issue isn't addressed.
- Which assets or business units are affected.

Add Notes

Leave internal context for team collaboration:

- Note types: remediation notes, verification status, assumptions, blockers.
- Notes persist across assets and weaknesses for traceability.



Add Note modal

Bulk 1-Click Verify

Group related weaknesses with a **single common fix**:

- Apply fix or verification to all grouped weaknesses at once.
- Helps speed up MTTR and improve clarity on shared vulnerabilities.

1-Click Verify Pentest modal

How to Use Ticketing in NodeZero

The Ticketing feature allows you to seamlessly create and manage issues in external platforms like ServiceNow or Jira directly from the NodeZero Vulnerability Management Hub. You can dispatch tickets from NodeZero to your integrated ticketing system, and each ticket will include details such as the weakness name, description, assets affected, fix actions, and any additional notes.

Prerequisites

Before using Ticketing, ensure the following are in place:

- Your organization has integrated Jira or ServiceNow with NodeZero.
- You have the correct permissions in both NodeZero and the external system.
- (Optional) Configure the webhook integration so that ticket status changes automatically sync back to NodeZero.

For setup instructions, see:

- Jira Setup Guide
- ServiceNow Setup Guide

Creating a ticket

You control exactly which weaknesses are turned into tickets. To create one:

1. In the VMH table, select a weakness that you want to track.
2. Open the actions menu (three dots ⋮).
3. Choose "Create Ticket."
4. A modal will appear, allowing you to edit the Ticket Name and add notes.
5. Click "Dispatch Ticket."

Create Ticket modal

Viewing and Managing Tickets

Once tickets are created, navigate to the **Ticketing** sub-tab inside VMH to see:

- Ticket ID and link to the external system.
- Current ticket status (e.g., Open, In Progress, Closed).
- The weakness associated with the ticket.

Ticketing page

Keeping Status Updated

- **With Webhooks:** Status changes in Jira/ServiceNow are automatically synced back into NodeZero.
- **Without Webhooks:** You'll need to open the actions menu (three dots ⋮) and click "Sync Ticket Status."

Common Use Cases

- Per-finding tickets: Create tickets only for high-priority weaknesses (no auto-creation).
- Tracking remediation progress: Monitor open/closed status from NodeZero without switching between tools.
- Audit trail: Maintain a single place where vulnerabilities and their associated tickets are visible.

Best Practices

- **Use ticketing selectively:** Focus on high-severity issues rather than every weakness.
- **Configure webhooks:** This ensures the smoothest synchronization between tools.
- **Audit your backlog:** Regularly review the "Closed" and "No Longer Found" items to ensure no security gaps have reappeared due to environment changes.

3.3.3 Threat Actor Intelligence

Threat Actor Intelligence

Threat Actor Intelligence connects validated attack paths discovered by NodeZero to the real-world adversaries most likely to exploit them.

Traditional vulnerability management prioritizes weaknesses using CVSS (Common Vulnerability Scoring System) scores or static threat feeds. However, attackers rarely choose targets based solely on severity. They exploit weaknesses that provide reliable paths to meaningful outcomes, such as credential theft, data exfiltration, or domain compromise.

Threat Actor Intelligence enriches NodeZero findings with adversary context, mapping exposures and attack paths to known threat-actor techniques and campaigns. This enables security teams to understand not only what is vulnerable, but who is likely to exploit the vulnerability and what the outcome could be.

By correlating exploit evidence from autonomous pentesting with real-world adversary behavior, Threat Actor Intelligence helps organizations prioritize remediation based on realistic attacker activity and business impact.

> **Add-on feature**
>
> This feature requires an Elite plan or RBVM add-on package. To enable it, please contact your account or Customer Success representative.

This page covers:

- Threat Actor Intelligence Defined
- Why It Matters
- How Threat Actor Intelligence Works
- Accessing Threat Actor Intelligence
- Using Threat Actor Intelligence for Prioritization
- Best Practices

THREAT ACTOR INTELLIGENCE DEFINED

Threat Actor Intelligence correlates validated attack paths with known adversary behaviors and techniques. Instead of presenting exposures as isolated vulnerabilities, Threat Actor Intelligence analyzes how weaknesses could be used in real attack campaigns.

This provides additional context for security teams, including:

- How exposures align with attacker techniques.
- Which threat actors are known to use similar methods.
- What outcomes those attack paths could achieve.

This approach helps organizations move beyond theoretical vulnerability prioritization toward attacker-informed risk reduction.

WHY IT MATTERS

Security teams often struggle with prioritizing remediation, because most tools provide incomplete context. Common challenges include:

- Thousands of vulnerabilities with little indication of real exploitability.
- Threat intelligence feeds that lack environment-specific context.
- Security reports that describe threats without demonstrating how they apply to a given organization’s infrastructure.

Threat Actor Intelligence addresses this gap by combining three critical elements:

- Validated exposures discovered through NodeZero pentesting.
- Adversary tactics and techniques observed in real-world campaigns.
- Attack paths showing how weaknesses can be chained together.

This enables teams to focus their remediation efforts on exposures that attackers are most likely to exploit.

HOW THREAT ACTOR INTELLIGENCE WORKS

Threat Actor Intelligence incorporates several layers of analysis to help you correlate NodeZero findings with known adversary techniques and campaigns.

Attack Path Validation

During the autonomous pentests that you run, NodeZero executes real attack chains. These attack paths demonstrate how an attacker could move from an initial foothold to sensitive assets. These validated attack paths form the foundation for other Threat Actor Intelligence features.

Adversary Technique Mapping

NodeZero‘s maps its attack steps to techniques from frameworks such as MITRE ATT&CK. This helps identify how discovered weaknesses align with known attacker tactics, techniques, and procedures.

Threat Actor Correlation

When relevant, Threat Actor Intelligence identifies the known threat actor behaviors and campaigns behind attack paths. This enables your security team to understand which adversaries are known to use similar techniques, and who might be targeting you.

Outcome Analysis

Threat Actor Intelligence evaluates the potential outcomes of a successful attack path, such as:

- Domain compromise.
- Credential harvesting.
- Data exfiltration.
- Lateral movement into critical systems.

This helps teams assess the potential business impacts of an exploit chain, and prioritize defenses.

ACCESSING THREAT ACTOR INTELLIGENCE

Threat Actor Intelligence is available directly within the NodeZero platform, and it enhances findings across the NodeZero Portal. Users can view adversary context alongside validated exposures and attack paths.

Where to Find Threat Actor Intelligence

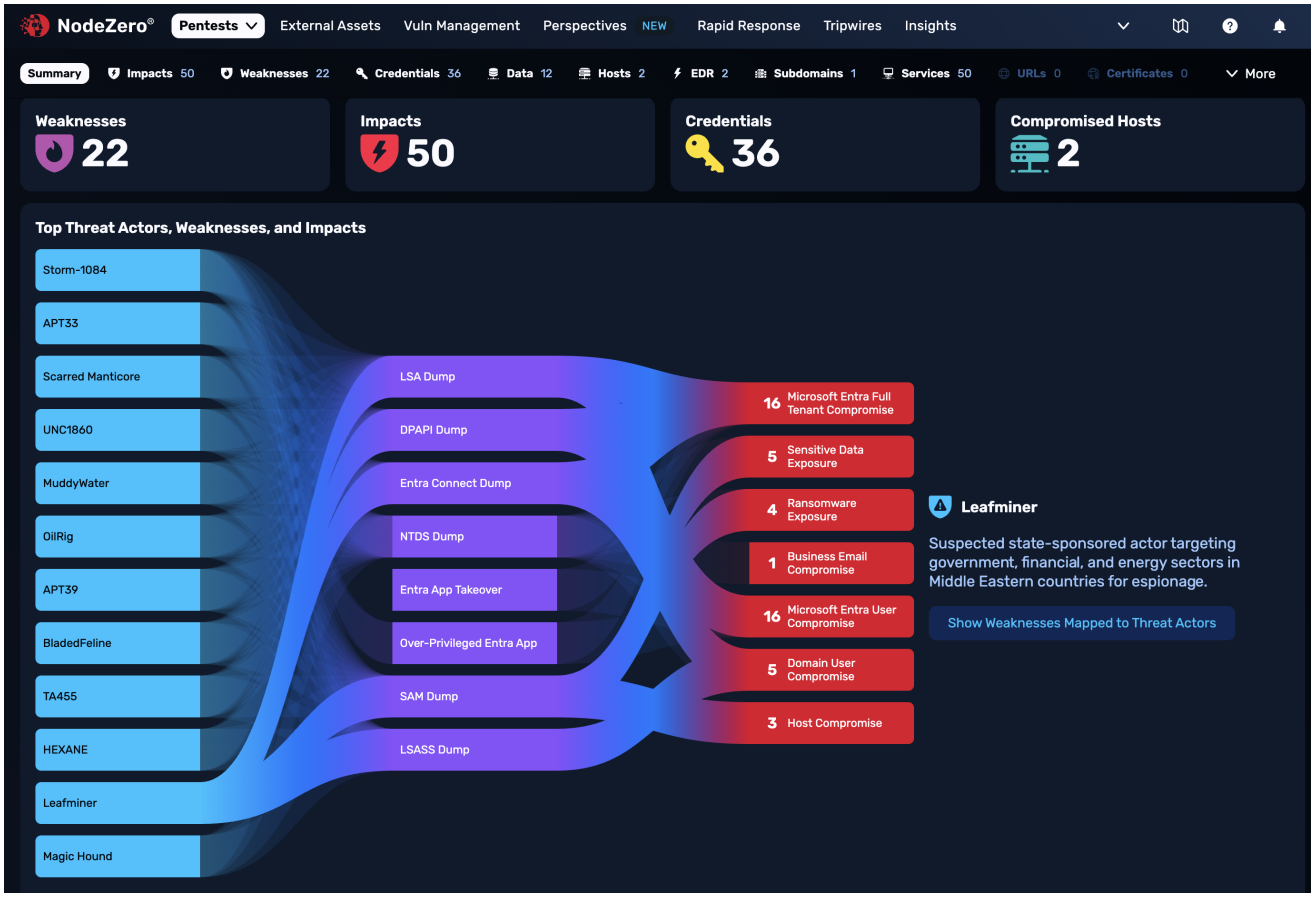
Threat Actor Intelligence information appears within several areas of the NodeZero Portal. Typical locations include:

- Sankey diagrams.
- Weakness context panels.
- Exposure analysis views.
- Risk intelligence summaries.
- Attack path findings.

Threat Actors Visualized

With Threat Actor Intelligence enabled, you can examine multiple threat actors' identities, exploits, and impacts in a single view:

1. Select **Pentests**.
2. Select a completed pentest.
3. On the pentet's **Summary** tab, scroll down to the sankey diagram.
4. As shown below, click a Threat Actor on the left to display summary details on the right.
5. In that right panel, optionally select **Show Weaknesses Mapped to Threat Actors** to proceed to the filtering view outlined below.



Revealing Threat Actors from Summary sankey diagram

Weaknesses by Threat Actor

To navigate directly to filtering weaknesses by individual Threat Actor, you can use the following path in the NodeZero Portal:

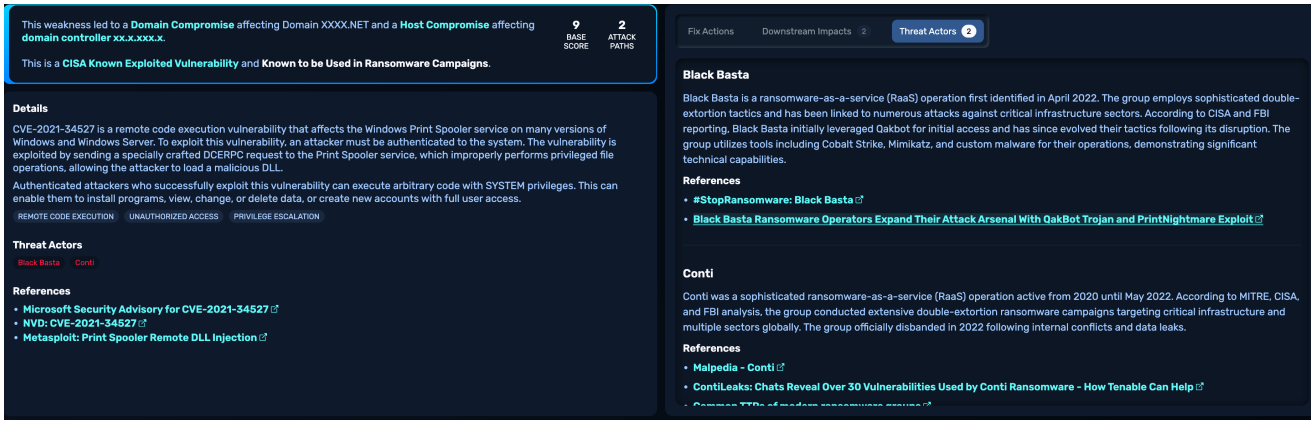
1. Select **Pentests**.
2. Select a completed pentest.
3. Select the **Weaknesses** tab within that pentest, if populated.
4. Look for the scrollable **By Threat Actor** panel at the right.
5. Here, click individual Threat Actors of interest to filter the bar graphs and table to isolate that actor.

Filtering by Threat Actor on a pentest's Weaknesses tab

Threat Actor Details

To see granular Threat Actors information within a weakness' exposure details, use the following path in the NodeZero Portal:

1. Select **Pentests**.
2. Select a completed pentest.
3. Select the **Weaknesses** tab within that pentest, if populated.
4. Select a weakness **Name** of interest to display its details.
5. Look for a **Threat Actors** tab on the right, to find detailed context about identified perpetrators.



Viewing Threat Actors context within a selected weakness

Threat Actor Context Panel

Within an exposure or attack path view, Threat Actor Intelligence provides additional context including:

- Relevant adversary techniques.
- Associated threat actor behaviors.
- Potential attacker objectives.

As shown in the following example, this information helps analysts quickly understand how discovered weaknesses relate to real-world attacker activity.

Threat Actors (12)

APT29 - State-sponsored by Russia, targets government and finance, motivated by espionage.

APT33 - State-sponsored by Iran, targets aviation, energy, and finance for espionage and disruption.

AvosLocker - RaaS group targeting multiple industries with sophisticated ransomware.

Black Basta - Independent ransomware group targeting multiple industries globally for financial gain.

Dragonfly - State-sponsored threat actor targeting energy and industrial sectors for espionage.

Ember Bear - State-sponsored Russian threat actor targeting government, defense, and energy sectors for espionage.

Ke3chang - State-sponsored Chinese threat actor targeting government, defense, and telecommunications for espionage.

Leafminer - State-sponsored threat actor from Iran, targeting Middle Eastern finance, government, and energy sectors for espionage.

menuPass - State-sponsored group from China targeting technology and manufacturing for espionage.

MuddyWater - Iranian state-sponsored group targeting energy, government, and finance sectors for espionage.

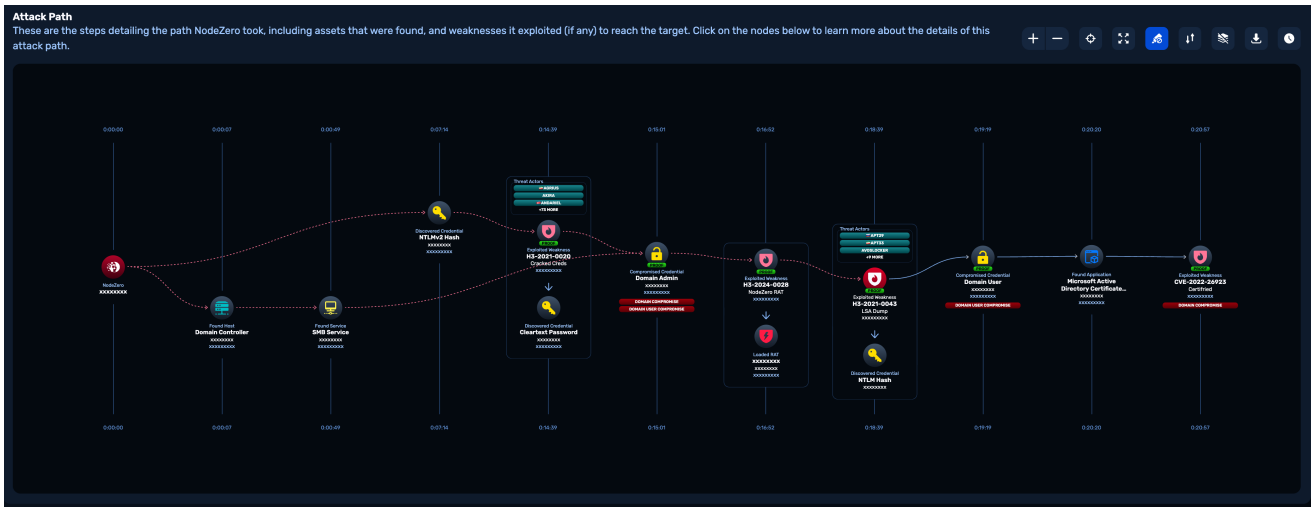
OilRig - Iranian state-sponsored threat actor targeting energy and finance sectors.

Threat Group-3390 - Threat Group-3390 is a Chinese state-sponsored APT group targeting government, defense, and tech sectors for espionage.

Viewing Threat Actors context within an attack path

Attack Path View

Threat Actor Intelligence also enhances attack path analysis by highlighting techniques used across the chain. This enables teams to see how attacker behavior evolves across different stages of an intrusion.



Attack path with techniques mapped to Threat Actors

USING THREAT ACTOR INTELLIGENCE FOR PRIORITIZATION

Threat Actor Intelligence helps security teams prioritize their remediation efforts based on realistic attacker behavior. Below are some common workflows.

Prioritizing Remediation

Security teams can prioritize exposures that:

- Enable meaningful attacker progress.
- Align with known attacker techniques.
- Lead to high-impact outcomes.

This reduces time spent addressing vulnerabilities that do not meaningfully increase risk.

Investigating Attack Paths

Threat Actor Intelligence helps analysts understand how weaknesses could be chained together to achieve attacker objectives. This supports deeper analysis of how attackers may move through an organization’s environment.

Communicating Risk to Leadership

Threat Actor Intelligence helps translate technical findings into business-relevant risk. Security leaders can more readily explain:

- Which exposures matter most.
- What outcomes attackers could achieve.
- How remediation reduces risk.

BEST PRACTICES

Organizations can maximize the value of Threat Actor Intelligence by following several practices:

- Run NodeZero pentests regularly, to identify new attack paths.
- Prioritize remediation of exposures associated with high-impact attacker outcomes.
- Re-test environments after remediation to confirm that exploit paths have been eliminated.
- Track changes in exposure over time, to measure improvements in security posture.

RELATED TOPICS

- Vulnerability Risk Intelligence
- NodeZero Assessments
- Attack Path Analysis
- Remediation and Re-Testing

3.3.4 Vulnerability Risk Intelligence

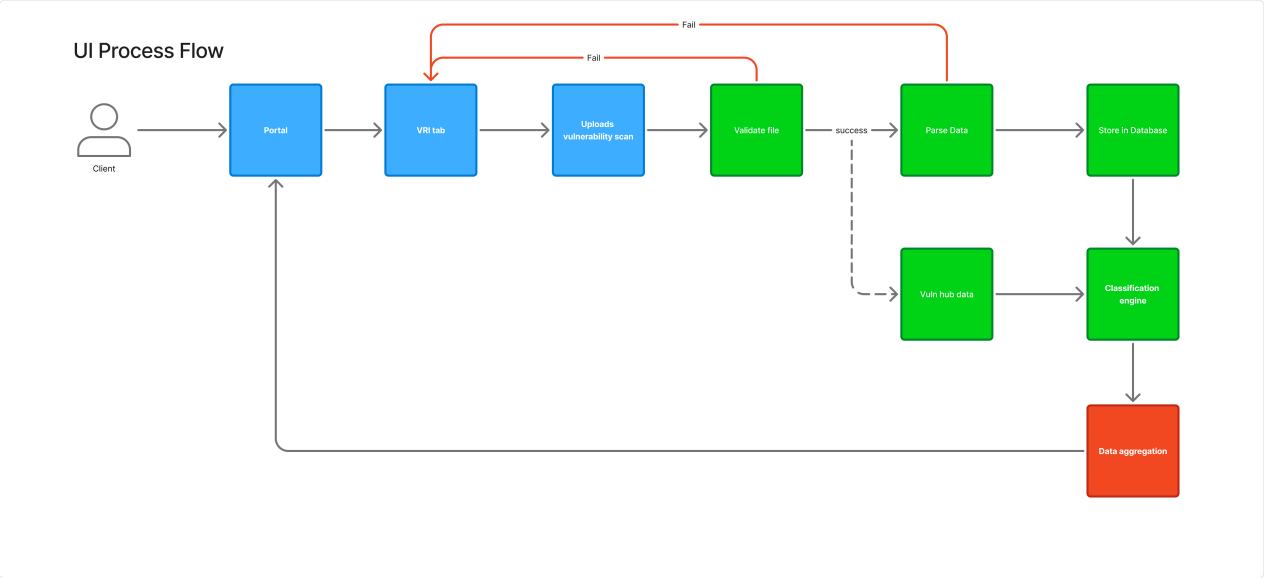
Vulnerability Risk Intelligence (BYO Scanner)

NodeZero's **Vulnerability Risk Intelligence (VRI)** feature enables customers to upload vulnerability scan exports from tools like Tenable (Nessus), Rapid7, and Qualys, and receive attacker-validated risk classification. NodeZero leverages real exploit evidence and attack correlation to prioritize findings based on what attackers can actually exploit.

OVERVIEW

NodeZero's VRI capability brings attacker-first validation to scanner data. Customers can upload vulnerability scan outputs (CSV or .nessus) via the UI or API, and NodeZero classifies each vulnerability-asset pair according to exploitability and context. NodeZero:

- Confirms exploitability
- Determines contextually exploitable attacks
- Highlights vulnerabilities tied to high-value assets or threat actor behavior
- Shows unique exploits and mitigated weaknesses found by NodeZero



HOW IT WORKS

1. **Upload** your scanner data to NodeZero via UI or API.
2. NodeZero uses its **Exploit Correlation Engine** to process each CVE-asset pair.
3. Vulnerabilities are **classified** into risk categories (see below).
4. Results are shown in the **Risk Intelligence tab** and can be exported via UI or API.

SUPPORTED SCANNER SOURCES

- **Tenable/Nessus CSV**
- **Tenable/Nessus XML**
- **Rapid7 CSV**
- **Qualys CSV**

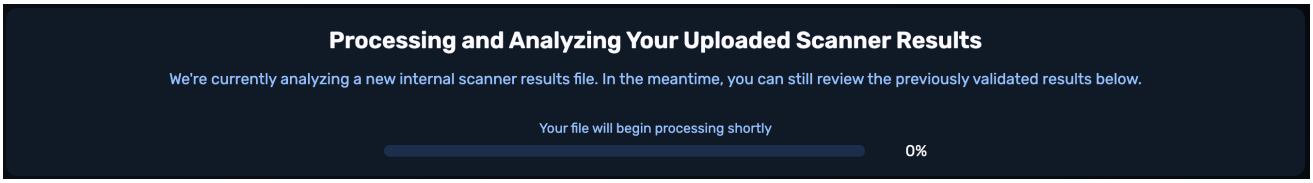
CLASSIFICATION CATEGORIES

Each vulnerability is classified into one of the following:

Category	Description
Confirmed Exploitable	Successfully exploited by NodeZero.
Contextually Exploitable	Reachable through RCE or chaining, but not directly landed.
High-Value Asset	Found on assets identified as mission-critical.
Threat Actor Detected	Tied to TTPs observed in real-world attacks.
Vulnerable but not Exploited	Not validated but still potentially exploitable.
Asset Not Enumerated	Asset couldn't be matched to NodeZero data.
Mitigated Weakness	Issue already mitigated based on prior results.

USING THE UI

1. Go to the **Risk Intelligence** tab in the Vulnerability Management Hub.
2. Upload your scanner file via drag-and-drop or file picker.
3. NodeZero processes your file within **1 hour** (SLA).
4. View classification results directly in the UI.



Upload Validation

- Max file size: **≤1GB**
- File type validation: CSV, XML, .nessus
- Duplicate check via checksum
- If the scope isn't in the file, user will be prompted to enter it manually.

USING THE API

1. Get a **pre-signed URL** from the API (requires authentication).
2. Upload your file using a POST request to the signed URL.
3. Fetch results via GET request.
4. API supports **CSV**, **XML**, and **JSON**
5. Token-auth required
6. Export formats: **JSON**

SCANNER FILE STATUSES

Status	Description
PENDING	The scanner file is awaiting to be uploaded or has been uploaded but risk intelligence has not started.
PROCESSING	The scanner file is undergoing risk intelligence.
FAILED	The scanner file failed risk intelligence. View the status message for more details.
COMPLETED	The scanner file completed intelligence processing.
EXPIRED	The scanner file is expired. Scanner files expire after 90 days.
DUPLICATE	The scanner file is a duplicate of another file. See status message for details.

SECURITY & COMPLIANCE

- Files uploaded via **HTTPS** to presigned S3 URLs
- Stored in **KMS-encrypted S3** buckets
- Retained temporarily (max 90 days), then deleted post-ingestion
- Normalized data retained permanently
- RBAC and full audit logging enabled
- SOC2 compliant, GDPR aligned

SCANNER FILE FIELD MAPPINGS

Tenable/Nessus CSV

Requirement	Field Type	Accepted Field Name(s)
Must Have	IP Address	`IP Address`, `Host`, `asset.display_ipv4_address`
Must Have	Plugin ID	`Plugin ID`, `definition.id`
Must Have	CVE	`CVE`, `CVE ID`, `definition.cve`
Should Have	Port	`Port`
Should Have	Protocol	`Protocol`
Should Have	FQDN	`FQDN`, `asset.display_fqdn`
Should Have	MAC Address	`MAC Address`, `asset.display_mac_address`
Should Have	OS	`OS`, `asset.operating_systems`
Should Have	NetBios	`NetBios`, `asset.netbios_name`

Tenable/Nessus XML

Requirement	Field Type	Accepted Field Name(s)
Must Have	IP Address	`host-ip`
Must Have	Plugin ID	`pluginID`
Must Have	CVE	`cve`
Should Have	Port	`port`
Should Have	Protocol	`protocol`
Should Have	FQDN	`host-fqdn`
Should Have	MAC Address	`mac-address`
Should Have	NetBios	`netbios-name`
Should Have	OS	`operating-system`, `os`

Rapid7 CSV

Requirement	Field Type	Accepted Field Name(s)
Must Have	IP Address	`Asset IP Address`, `IP Address`
Must Have	Vulnerability ID	`Vulnerability ID`
Must Have	CVE	`Vulnerability CVE IDs`, `CVE`
Should Have	Port	`Service Port`
Should Have	Protocol	`Service Protocol`
Should Have	FQDN	`FQDN`
Should Have	MAC Address	`Asset MAC Addresses`
Should Have	OS	`Asset OS Name`, `OS`

Qualys CSV

Requirement	Field Type	Accepted Field Name(s)
Must Have	IP Address	`IP`
Must Have	QID	`QID`
Must Have	CVE	`CVE ID`
Should Have	Port	`Port`
Should Have	Protocol	`Protocol`
Should Have	FQDN	`FQDN`
Should Have	MAC Address	`MAC Address`
Should Have	NetBios	`NetBIOS`
Should Have	OS	`OS`

Additional Field-Matching Details

- Field matching is case-sensitive. Submissions with incorrect casing may not be processed correctly.
- When multiple field names are listed for a field type, they are listed in priority order. If your scanner file contains more than one of the accepted field names, NodeZero will use the first match found in the priority order.

FAQS

Q: Can I sanitize scan files before upload (e.g., remove hostnames/IPs)?
A: No, that would prevent asset mapping and classification from working properly. NodeZero requires full host/vuln pairing.

Q: Is data encrypted?
A: Yes, in transit (HTTPS) and at rest (AES-256 with AWS KMS).

Q: How long is data stored?
A: Files are deleted after 90 days. Metadata and classification results are retained indefinitely.

Q: Do you support PDF uploads?
A: No. Only structured formats like CSV or .nessus are supported.

Q: Why do vulnerability counts not match?
A: When you upload a scanner file (e.g., from Tenable) to NodeZero's Vulnerability Risk Intelligence, you may notice a mismatch between the number of vulnerabilities listed in your original scan and the number displayed in the NodeZero platform.

This is expected behavior, due to how NodeZero processes and normalizes scanner data. Let's say your Tenable scan includes a single vulnerability entry with this list of CVEs:

```
CVE-2024-12345, CVE-2024-67890, CVE-2025-12345, CVE-2025-67890
```

NodeZero will treat each CVE as an individual weakness and create four separate records — one for each CVE. This allows for more precise correlation between scanner results and what NodeZero discovers or confirms during the autonomous pentest.

As a result, it's common for the total number of records in the NodeZero report to exceed the original row count in your scanner export.

Additional Notes

- This CVE-level granularity enables NodeZero to provide enhanced risk insights, including confirmed versus unconfirmed weaknesses, chaining potential, and downstream impact.
- Filters like **"Found by Scanner Only"** may return more results than expected because they include each CVE as a separate item, rather than in a grouped entry.

FEATURE AVAILABILITY

- Available for customers on the **Elite SKU**
- Requires RBAC permissions to use the Vulnerability Management module

---

Note

VRI currently only supports the ingestion of internal (e.g. network, workstation) scanner data.

SEE ALSO

- NodeZero API Documentation
- NodeZero VM Hub
- Data Privacy Policy

3.3.5 MCP Server

NodeZero MCP Server

The NodeZero MCP (Model Context Protocol) Server acts as a natural language bridge between your LLMs and the NodeZero API, enabling AI agents to directly integrate real-world pentesting capabilities into agentic workflows. By securely exposing NodeZero's operations to LLMs, the MCP Server empowers your AI to: understand offensive security context, prioritize risk based on exploit-driven data unique to your organization, and execute pentesting operation types (op types) as part of autonomous Find, Fix, Verify (FFV) cycles.

USING MCP SERVER

Use the MCP Server to integrate generative artificial intelligence in your NodeZero workflows:

- Use natural language with your LLM or AI agent to query via the NodeZero API for your organization's asset, vulnerability, and exposure context.
- AI drives remediation or incident response workflows using organization-specific, exploit-driven prioritization.
- Launch and manage pentests directly from your LLM without using the NodeZero Portal.
- Feed offensive security intelligence into agentic workflows for rapid verification and risk reduction.
- Simulate attacker behavior for exposure mapping, attack path visualization, or "what-if" security exercises while understanding risk in layman's terms.
- Enable LLM-powered security orchestration that automates portions of Find, Fix, Verify (FFV) driven through real exploitability context.
- Tailor system prompts to your specific use cases to improve response accuracy and reduce token usage.

NEXT STEPS

Review the differences, and then choose either a locally-hosted MCP server or the Horizon3.ai-hosted MCP Server for use with your IDE or other AI client and your LLM models.

Feature	Horizon3.ai-hosted MCP Server	Locally hosted MCP Server
Authentication	OAuth 2.1	Self generated Portal API Key (stored locally)
User Mode	Multi-user supported	Single only
Deployment	Simple	Requires configuration
Transport protocol default	Streamable HTTP	stdio recommended
Infrastructure	Maintained by H3	Self hosted & maintained
Customizability	Prompts and Agents by H3	Highly customizable

- Horizon3.ai-hosted MCP Server setup instructions Recommended
- Locally-hosted MCP Server setup instructions

Horizon3.ai-hosted MCP Server

GET STARTED WITH HORIZON3.AI-HOSTED MCP SERVER

Set up the MCP Server hosted by Horizon3.ai in your IDE or other AI client.

Prerequisites

- An IDE or AI client that supports streamable HTTP and OAuth 2.1 (e.g. VS Code, Cursor, Windsurf)
- A modern browser for completing the OAuth login

Setting Up Visual Studio Code

To set up the Horizon3-hosted MCP Server in Visual Studio Code:

1. Open the command palette in VS Code.
2. Click **Show and Run Commands**.
3. Run the command `MCP: Add Server`.
4. Select `HTTP (HTTP or Server-Sent Events)` as the type.
5. Enter the server URL:

- **US:** `https://mcp.horizon3ai.com/mcp`
- **EU:** `https://mcp.horizon3ai.eu/mcp`
- **FedRAMP High:** `https://mcp.gov-horizon3ai.com/mcp`

6. Provide a name for the server (for example, `horizon3-mcp-server`).

```
# Visual Studio Code creates an `mcp.json` file (or adds to it if it already exists) with the information you just entered
    {
    "servers": {
        "horizon3-mcp-server": {
            "url": "https://mcp.horizon3ai.com/mcp",
            "type": "http"
        }
    },
    "inputs": []
    }
```

7. After saving your configurations, VS Code will prompt you to authenticate to `proxy.horizon3ai.com`

1. After clicking `Allow`, log in to your Horizon3 Portal Account following the redirect. Once you successfully log in, close the browser and you're ready to get started!

Client Compatibility

The Horizon3 MCP Server works with MCP-compliant clients that support both OAuth 2.1 authentication and **HTTP streamable transport**. This includes:

- **VS Code** with MCP extensions - Fully compatible
- **Claude Code** - Fully compatible
- **Claude Desktop** - Currently has some known compatibility issues
- **Gemini CLI** - Full compatible
- **Windsurf** - Fully compatible

Set Up Other IDEs or Clients

1. In your client of choice, enter the server URL:

- **US:** `https://mcp.horizon3ai.com/mcp`
- **EU:** `https://mcp.horizon3ai.eu/mcp`
- **FedRAMP High:** `https://mcp.gov-horizon3ai.com/mcp`

2. Use streamable HTTP as the transport mode
3. Begin the OAuth login process when prompted and complete using your NodeZero credentials after being redirected

Manual Set Up

To set up the Horizon3-hosted MCP Server for other IDEs or clients manually, create an `mcp.json` file in your workspace or home directory with an entry for the MCP Server:

```
{
   "servers": {
       "horizon3-mcp-server": {
           "url": "https://mcp.horizon3ai.com/mcp",
           "type": "http"
       }
   },
   "inputs": []
}
```

Troubleshoot Common Issues

Issue	What You'll See	Solution
General connection problems	Authentication fails or MCP server won't connect	Restart your MCP client (VS Code, Claude Code, etc.) - this fixes most issues
Network/firewall blocking	Can't reach authentication or MCP endpoints	Ensure you can access `portal.horizon3ai.com`, `oauth-proxy.horizon3ai.com`, and `mcp.horizon3ai.com`
Redirect URI not whitelisted	Error about redirect URI during OAuth flow	Contact support to whitelist your client's specific callback URL if it's not localhost/127.0.0.1
Browser doesn't redirect back	Stuck on "redirecting" or success page	Close the browser tab and restart your MCP client connection
Session expired	MCP server stops responding after ~1 hour	Reconnect to get a fresh authentication token

AUTHENTICATION & AUTHORIZATION DETAILS

Authentication & Authorization Details

This page explains how the H3 MCP Server keeps your data secure and how to connect your applications.

The Horizon3.ai-hosted MCP Server, uses **OAuth 2.1** to ensure secure access to your data. Secure access means:

- **Secure login** - Uses your existing Portal account credentials
- **No password sharing** - Applications never see your login details
- **Industry standard security** - Uses the same OAuth protocol as major platforms
- **Session-based access** - Once authenticated, you can use the server until your session expires

How to Connect In VS Code

When setting up the Horizon3.ai MCP Server in VS Code:

1. **Add the server URL** - Configure VS Code with the Horizon3.ai MCP Server production URL as explained in Horizon3.ai-Hosted MCP Server Setup
2. **Start the connection** - Click to start using the MCP server
3. **Authorize the connection** - VS Code will ask if you want to connect to the OAuth server
4. **Login via browser** - Your browser opens and takes you to the Portal login page
5. **Complete authentication** - Log in with your Portal credentials (or skip if already logged in)
6. **Return to VS Code** - The browser redirects back to VS Code automatically
7. **Start using the server** - You're now connected and can access MCP server features

What happens during login

- If you're already logged into Portal, the browser redirects back to VS Code immediately
- If you need to log in, you'll use your regular Portal login preference
- The authentication happens in your browser, so VS Code never sees your credentials
- Once complete, VS Code receives secure access credentials to use the MCP server

Technical Details

The H3 MCP Server uses OAuth 2.1 with PKCE (Proof Key for Code Exchange) and JWT (JSON Web Tokens) for secure authentication. This ensures:

- **No insecure client secrets** - VS Code doesn't need to store sensitive credentials
- **Protection against attacks** - PKCE prevents authorization code interception
- **Secure token format** - JWT tokens provide cryptographically signed access credentials
- **Standards compliance** - Uses current security best practices
- **Cross-platform compatibility** - Works consistently across different operating systems

Redirect URI requirements

The Horizon3.ai MCP Server accepts callback URLs that use `localhost` or `127.0.0.1` with any port (for example, `http://127.0.0.1:33418` or `http://localhost:3000`). If your MCP client uses a different callback URL, you may see an error about the redirect URI not being whitelisted - contact support to have your specific URI added to the whitelist.

Security Features How your data stays protected

- **Browser-based authentication** - Login happens in your secure browser environment
- **No stored passwords** - Your credentials are never saved in VS Code or other applications
- **Encrypted connections** - All communication between applications and the server is encrypted
- **Session management** - Access automatically expires after a period of inactivity

Token-Based security

Instead of passwords, the system uses secure access tokens:

- **Automatic management** - VS Code handles tokens in the background
- **Time-limited** - Tokens expire after 1 hour for security
- **Unique per session** - Each connection gets its own secure token
- **Revocable** - Access can be revoked if needed

Manage Your Connection Session duration

- Sessions remain active while you're using VS Code
- Inactive sessions expire automatically for security
- You'll need to reconnect if your session expires

Disconnecting

- Close VS Code to end your MCP server session
- Your authentication will expire automatically
- No manual logout required

Security Best Practices For your protection

- **Use trusted networks** - Avoid public WiFi for initial setup when possible
- **Keep Portal credentials secure** - Use strong passwords and enable two-factor authentication if available
- **Close VS Code when done** - Especially on shared computers
- **Monitor activity** - Be aware of when and where you're connecting to the MCP server

Get Help

If you encounter authentication issues:

1. **Restart your client** - Close and reopen your MCP client (VS Code, Claude Code, etc.)
2. **Check network access** - Ensure you can reach `portal.horizon3ai.com`, `oauth-proxy.horizon3ai.com`, and `mcp.horizon3ai.com`
3. **Verify callback URL** - Make sure your client uses `localhost` or `127.0.0.1` for OAuth redirects
4. **Clear browser cache** - Sometimes helps with redirect issues
5. **Contact support** - If problems persist, reach out with specific error messages

The authentication process is designed to be simple and secure. Once set up, you shouldn't need to think about it - your client will handle the connection automatically.

Locally-hosted MCP Server

GET STARTED WITH LOCALLY-HOSTED MCP SERVER

To get started with a locally-hosted MCP Server choose a setup method:

Deploy MCP Server with IDE — Recommended

Step-by-step instructions to set up the NodeZero MCP Server

Deploy MCP Server with SSE, HTTP or Manually

Additional instructions for alternative deployment methods

Deployment help:

- Troubleshooting Locally-Hosted MCP Server Setup
- Security Best Practices

DEPLOY WITH IDE (RECOMMENDED)

IDE Deployment

The NodeZero MCP Server allows you to connect your IDE or AI assistant to Horizon3.ai APIs using your own infrastructure and LLM. Use this guide to quickly deploy a **locally hosted MCP Server** and start using NodeZero with your own tools.

Current Limitations

- Stdio mode is the **Recommended** deployment path.
- All MCP server modes (stdio, SSE, HTTP) are local and single-user
- The server runs only on the system where you launch the Docker container.
- Multi-user mode is **not supported**.
- **VS Code / GitHub Copilot must use "Agent" mode** for MCP servers.
- Do not use "Ask" or "Edit" modes — they will not leverage MCP tools.

> **Possible Security Risks**
>
> Using SSE/Streamable HTTP mode with this locally deployed MCP might present a **security risk**, because this mode does not support runtime isolation and your JWT token will not be isolated to a session ID.

Create an API Key for the MCP Server

The NodeZero Runner uses the `h3-cli` to communicate with the Horizon3.ai API, requiring an API key with specific permissions.

- **Navigate to API key settings:**
- In the Portal, click the **Profile Menu** (upper right corner), then click **Settings**
- Click **My Settings** to display the **API Keys** section.

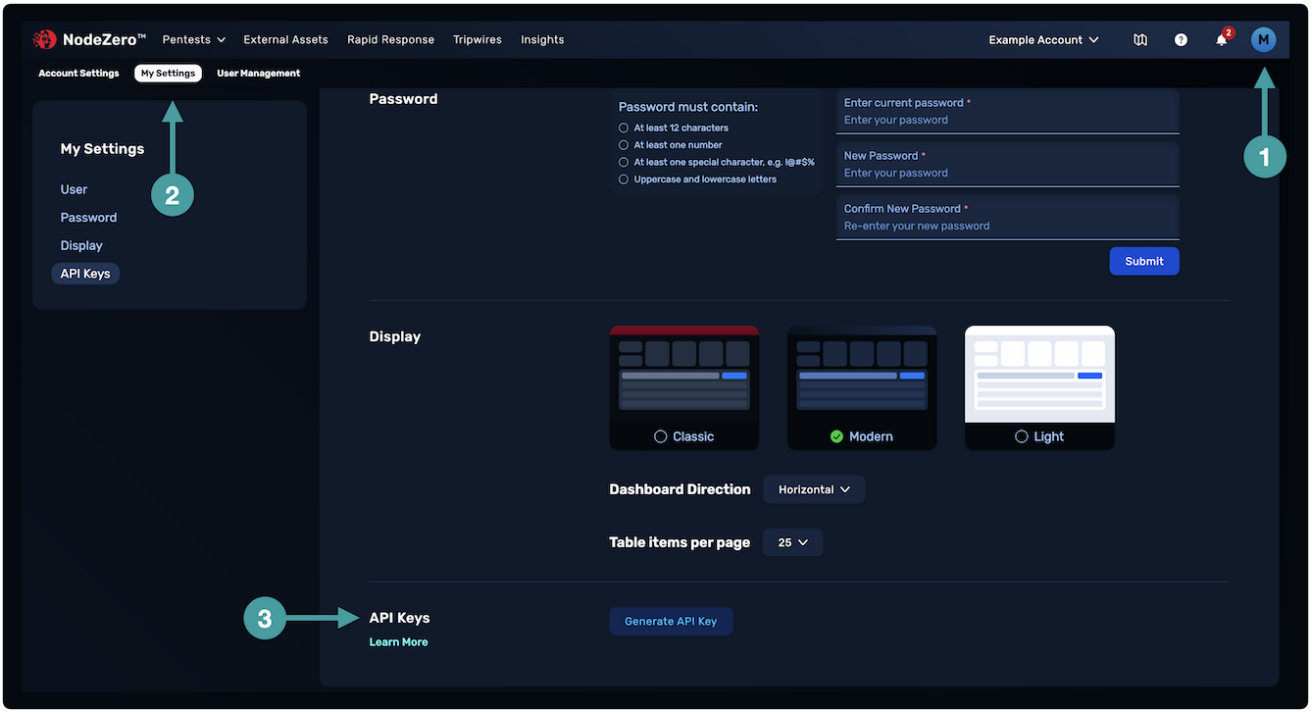


Figure 1 - My Settings page with the API Keys section visible.

- **Generate a key:**
- Create a new API key and set its permission level to **User**.
- Save the API key securely. You will need it in a following step.

Install Docker

> **Success**
>
> If you are using an already configured NodeZero Host or OVA, Docker should already be installed.

Follow these steps to install and configure Docker to enable NodeZero functionality on your host. Check out the Docker Documentation for troubleshooting.

Begin by updating your system and installing Docker, run the following commands:

```
sudo apt update
sudo apt install -y docker.io
```

Start and Enable Docker

Ensure Docker starts automatically, run the following commands:

```
sudo systemctl start docker
sudo systemctl enable docker
```

Confirm that Docker Was Successfully Installed

Check the Docker version by running:

```
docker --version
```

Run a test container to verify functionality by running:

```
sudo docker run hello-world
```

Download the NodeZero MCP Server Container

Download the container from either `Docker Hub` or the `AWS Public Registry`.

Docker Hub **AWS Public Registry**

```
docker pull horizon3ai/h3-mcp-server:latest
```

```
docker pull public.ecr.aws/horizon3ai/h3-mcp-server
```

Setting up NodeZero MCP Server with VS Code and Copilot in Agent Mode

Prior to starting, ensure that VS Code is configured to support stdio MCP servers - see VS Code MCP Server with Copilot Setup

> **Compatible IDEs**
>
> Other MCP-enabled IDEs or MCP clients should be compatible. However, please reference official documentation from your preferred method.

Instructions for VS Code:

1. Open VS Code with GitHub Copilot enabled.
2. Open a **Copilot chat** and switch to **Agent** mode.
3. Click Configure Tools → Add MCP Server > Command (stdio).
4. When prompted for Enter Command, type `docker` and press Enter.
5. Provide a name for the server (e.g., `NodeZeroMCP`) and press Enter. Use this name in the JSON input below replacing `<your-mcp-server-name>`
6. VS Code will open a `mcp.json` or `settings.json` tab. If not, press `Cmd+Shift+P` (Mac) / `Ctrl+Shift+P` (Windows/Linux), type Open Settings JSON, and select Preferences: Open User Settings (JSON).
7. Update your configuration as follows. Use the `inputs` feature to securely prompt for the API key (generated from step 1) at runtime, keeping it out of the file and reducing risk of accidental exposure.

US Region **EU Region** **Example with other MCP Servers**

```
{
  "servers":{
    "<your-mcp-server-name>": {
      "type": "stdio",
      "command": "docker",
      "args": [
        "run",
        "--pull", "always",
        "-i",
        "--rm",
        "-e", "H3_API_KEY",
        "horizon3ai/h3-mcp-server:latest"
      ],
      "env": {
        "H3_API_KEY": "${input:h3_api_key}"
      },
      "inputs": [
        {
          "type": "promptString",
          "id": "h3_api_key",
          "description": "H3 API Key",
          "password": true
        }
      ]
    }
  }
}
```

```
{
  "servers":{
    "<your-mcp-server-name>": {
      "type": "stdio",
      "command": "docker",
      "args": [
        "run",
        "--pull", "always",
        "-i",
        "--rm",
        "-e", "H3_API_KEY",
        "-e", "H3_GQL_URL=https://api.horizon3ai.eu/v1/graphql",
        "-e", "H3_AUTH_URL=https://api.horizon3ai.eu/v1/auth",
        "horizon3ai/h3-mcp-server:latest"
      ],
      "env": {
        "H3_API_KEY": "${input:h3_api_key}"
      },
      "inputs": [
        {
          "type": "promptString",
          "id": "h3_api_key",
          "description": "H3 API Key",
          "password": true
        }
      ]
    }
  }
}
```

```
{
  "servers": {
    "filesystem": {
      "type": "stdio",
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem", "/path/to/allowed/files"]
    }, // (1)!
    "NodeZeroMCP": {
      "type": "stdio",
      "command": "docker",
      "args": [
        "run",
        "--pull", "always",
        "-i",
        "--rm",
        "-e", "H3_API_KEY",
        "horizon3ai/h3-mcp-server:latest"                // (2)!
      ],
      "env": {
        "H3_API_KEY": "${input:h3_api_key}"
      },
      "inputs": [
        {
          "type": "promptString",
          "id": "h3_api_key",
          "description": "H3 API Key",
          "password": true
        }
      ]
    } // (3)!
  }
}
```

1. **Add a comma here** - When you have an existing MCP server, you must add a comma after its closing brace to separate it from the next server entry.

2. **NodeZero server configuration** - This is where you paste the complete NodeZero MCP server configuration. Make sure it's properly indented and nested within the "servers" object.

After configuring VS Code with NodeZero MCP server it will:

- Securely prompt for your API key
- Pull the latest container
- Connect to NodeZero MCP server via stdio in Agent Mode.

DEPLOY WITH SSE, HTTP OR MANUALLY

SSE and HTTP Deployments

Security Risk

Use SSE or Streamable HTTP **only if your IDE or GenAI client cannot use stdio**. Using SSE or Streamable HTTP mode with this locally deployed MCP may present a **security risk** as this mode does not support runtime isolation and your JWT token will not be isolated to a session ID.

SSE and Streamable HTTP Deployment configurations

We do not recommend using SSE or Streamable HTTP with a locally hosted MCP Server due to inherent security limitations:

- No runtime isolation: The MCP process shares its environment with your host.
- JWT tokens are not session-scoped: Tokens are valid globally, not tied to a session ID.
- Increased attack surface: Any exposed HTTP endpoint can potentially be abused.

If you **MUST** use this mode (e.g., your IDE or GenAI client cannot use stdio). It is recommended to limit your exposure by:

- Limit to 1 user, 1 API key per NodeZero MCP Server instance.
- Do not expose the endpoint to untrusted networks.
- Consider running in a sandboxed environment or isolated container.

Sample configuration JSON for enabling on with an IDE

SSE Deployment **Streamable HTTP Deployment**

```
"<your-mcp-server-name>": {
    "type": "sse",
    "command": "docker",
    "args": [
        "run", "-i", "--rm",
        "-e", "H3_API_KEY",
        "horizon3ai/h3-mcp-server:latest",
        "sse", "8000"
    ],
    "env": {
        "H3_API_KEY": "${input:h3_api_key}"
    },
    "inputs": [
        {
        "type": "promptString",
        "id": "h3_api_key",
        "description": "H3 API Key",
        "password": true
        }
    ]
}
```

```
"<your-mcp-server-name>": {
    "type": "http",
    "command": "docker",
    "args": [
        "run", "-i", "--rm",
        "-e", "H3_API_KEY",
        "horizon3ai/h3-mcp-server:latest",
        "streamable-http", "8000"
    ],
    "env": {
        "H3_API_KEY": "${input:h3_api_key}"
    },
    "inputs": [
        {
        "type": "promptString",
        "id": "h3_api_key",
        "description": "H3 API Key",
        "password": true
        }
    ]
}
```

Manual Deployment

Set up the NodeZero MCP Server container.

Run the container manually with a descriptive name `<your-container-name>`:

```
docker run -d --name <your-container-name> \
  -e H3_API_KEY={your-key-here} \
  horizon3ai/h3-mcp-server:latest
```

Stop it with:

```
docker stop <your-container-name>
```

Stdio Mode (Attach-Only via IDE)

```
"<your-mcp-server-name>": {
  "type": "stdio",
  "command": "docker",
  "args": ["run", "-i", "--rm", "horizon3ai/h3-mcp-server:latest"]
}
```

SSE Mode (Attach-Only via IDE)

```
"<your-mcp-server-name>": {
  "type": "sse",
  "url": "http://localhost:8000/sse"
}
```

Streamable HTTP Mode (Attach-Only via IDE)

```
"<your-mcp-server-name>": {
  "type": "http",
  "url": "http://localhost:8000/mcp"
}
```

TROUBLESHOOT & ADDITIONAL TIPS

Troubleshooting and Additional Deployment Tips

Diagnosing Docker Issues

Check Docker Service state:

```
sudo systemctl status docker
```

- Inactive/failed → service issue.
- Active but CLI fails → daemon issue.

Service Commands:

```
sudo systemctl start docker      # Starts the service and daemon
sudo systemctl stop docker       # Stops the service and daemon
sudo systemctl status docker     # Shows service + daemon status
sudo systemctl enable docker     # Auto-start daemon at boot
```

Restart Docker Daemon:

```
sudo systemctl restart docker
pgrep -x dockerd || sudo dockerd
```

If `dockerd` is run manually, it stops when terminal closes. Enable it for auto-start.

Run MCP Server in Background

```
docker run -d --name <your-container-name> \
  -e H3_API_KEY={your-key-here} \
  horizon3ai/h3-mcp-server:latest
```

Stop later:

```
docker stop <your-container-name>
```

Clean Up Containers and Keys

```
docker rm <your-container-name>
unset H3_API_KEY
```

Use Single-User Mode

Each container should use one API key.

SECURITY BEST PRACTICES

Security Best Practices

For additional information on how we built this locally hosted MCP server with security by design, check out our NodeZero MCP Server blog.

- **Use single-user mode**: A single instance of the Horizon3.ai MCP Server is designed to use a single Horizon3.ai API key.
- **Pass keys securely**: The only way to pass the API key to the server is via the `-e H3_API_KEY={your-key-here}` option on the `docker run` command.
- **Restrict network**: Run locally or behind VPN/firewall.
- **Stop/remove containers** when not in use.
- **Rotate keys regularly** and test before use in production.
- **Create GraphQL examples**: Prepare sample queries for fetching test data or triggering NodeZero assessments.

Tools

MCP SERVER TOOLS

MCP Server provides the following tools to AI assistants:

1. `fetch_h3_graphql_docs` : Get GraphQL schema documentation for constructing queries
2. `run_h3_graphql_request` : Execute GraphQL queries against the Horizon3.ai API

3.3.6 Attack Configuration

Attack Configuration

When configuring a pentest with NodeZero, users are given the option to enable or disable a set of Attack Configuration Options. These options are controllable because they effect the performance of the pentest, or because they have the potential to disrupt the target environment.

This reference page lists the attack configuration flags available in NodeZero, along with descriptions of how these flags effect NodeZero's behavior.

When all options are disabled, the operation is still a pentest. Click for more info.

The following activities are performed:

- Asset Discovery
- Identifying potential vulnerabilities
- Exploiting most vulnerabilities/misconfigurations (that have been vetted to not have an operational impact on the target)
- Limited credential discovery and credential pivoting

The following activities are not performed:

- Windows Active Directory attacks
- Man-in-the-middle attacks
- Hash cracking
- Password Spray
- Azure AD pivoting
- Default Cred checking
- OS credential dumping
- Any brute force enumeration
- Any exploits specifically disabled in the advanced config (but most exploits are still executed as described above)

BRUTE FORCE

Properties related to modules that carry out brute force attacks.

Name	Description	Risk
DNS	Enables brute forcing of internal DNS records. Only applies if an operation has been scheduled with the Intelligent Scope option. This may place noticeable load on DNS servers in the network.	low
S3	Enables brute-force discovery of S3 buckets using wordlists and top level company domain names. This can add significant time if the pentest has been configured to run against many top-level domains.	none
Subdomains	Enables brute-force discovery of company subdomains using a large wordlist of common subdomain names. This can aid in the discovery of more external assets but significantly extend the time it takes for discovery to complete.	none

CREDENTIAL VERIFICATION

Properties related to modules that use credentials discovered by NodeZero to access services in the environment.

Name	Description	Risk
MS Entra (Azure AD) Credential Pivoting	Enables using domain user credentials discovered in an internal pentest against Azure Active Directory services. Requires user-entered Domains from the OSINT step.	none
Azure AD Password Spray	Enables password spraying Azure cloud users with common passwords. By default, a given user will be tried only three times every 60 minutes. There is a small chance of locking out accounts.	moderate
Credential Reuse	Checks for access to services and shares using local user (non-domain) authentication.	none
Domain User	Checks for Windows domain user access by authenticating with credentials against the SMB service running on the Windows Domain Controller.	none
Internal Password Spray	Enables password-spraying domain users with common passwords. We recommend enabling the Brute Force: Domain User flag along with this option. By default, a given user will be tried only twice every 60 minutes. If NodeZero detect three or more users as locked out from a given domain, it stops spraying passwords on that domain.	moderate

DATA

Properties related to data discovery.

Name	Description	Risk
Domain Admin Scanning of SMB Shares	Enables scanning of SMB shares using domain administrator credentials that were injected into the pentest or discovered during the course of the pentest. Enabling this flag provides a more complete picture of data risk but can add significant time to the pentest.	none
Extended Domain User Scanning of SMB Shares	Enables scanning of all SMB shares accessible to domain users whose credentials were injected into the pentest or discovered during the course of the pentest.	none
Verify Permissions on SMB Shares	Verify read, write, list, and delete permissions on an SMB share by writing a test file and deleting it afterwards. Cleanup of the test file may fail in exceptional circumstances.	none

DEFAULT CREDENTIALS

Properties related to modules that check for default credentials using a dictionary attack with known default credentials.

Name	Description	Risk
FTP	Enables checking default credentials against FTP services found by NodeZero.	low
Microsoft SQL Server	Enables checking default credentials against Microsoft SQL Server databases found by NodeZero. There is a small chance of locking out the `sa` account.	moderate
MongoDB	Enables checking default credentials against MongoDB databases found by NodeZero.	low
MySQL	Enables checking default credentials against MySQL databases found by NodeZero.	low
PostgreSQL	Enables checking default credentials against PostgreSQL databases found by NodeZero.	low
SNMP	Enables checking default SNMPv1 community strings against SNMP services found by NodeZero.	low
SSH	Enables checking default credentials against SSH services found by NodeZero. Against older ESXi servers vulnerable to CVE-2019-5528, this module may trigger a partial denial of service condition in the `hostd` process.	moderate
Telnet	Enables checking default credentials against telnet services found by NodeZero.	low
Web	Enables checking default credentials against HTTP or HTTPS web servers found by NodeZero.	low

ENVIRONMENT IMPACT

Properties related to modules that change the environment. All modules attempt to clean up after themselves but there is a small chance cleanup may fail.

Name	Description	Risk
ADCS ESC4 Attack - Misconfigured Templates Access Controls	Exploit vulnerable Active Directory Certificate Templates that allow an unprivileged user to overwrite Certificate Template security features -- enabling Subject Alternative Name (SAN). Restoration of original template configuration may fail in exceptional cases.	none
Anonymous Docker Engine Write Check	Checks for write privileges against a Docker Engine instance that allows anonymous (unauthenticated) access. The check attempts to create a Docker container or pull a Docker image and deletes the container or image afterwards.	none
Anonymous Printer Access	Check for anonymous access to printers over port 9100. This check may cause certain printer models to print out pages.	moderate
Anonymous ZooKeeper Write Check	Checks for write privileges against a ZooKeeper instance that allows anonymous (unauthenticated) access. The check writes to a ZooKeeper node and deletes it afterwards.	none
Atlassian Crowd and Crowd Data Center Remote Code Execution (CVE-2019-11580)	Checks for exploitability of CVE-2019-11580 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none
CVE-2022-26923 (Certifried) Privilege Escalation - Creation of Machine Account	Attempt to exploit vulnerable Active Directory Certificate Services Privilege Escalation by creating a machine account and manipulating its attributes. Deletion of the machine account may fail in exceptional circumstances.	none
Elasticsearch Write Check	Checks for write privileges against an Elasticsearch cluster. The check attempts to create an index and deletes it afterwards.	none
FTP Write Check	Checks for write privileges against an FTP server. The check creates a remote directory and deletes it afterwards.	none
Insecure JMX (H3-2020-0022)	Tests exploitability of the insecure JMX weakness (H3-2020-0022). The test checks for remote code execution by installing a payload on the vulnerable JMX service, runs a small set of commands using the payload, and uninstalls the payload at the end. There is a small chance that cleanup of the payload may fail.	none
ManageEngine ServiceDesk Plus PreAuth RCE (CVE-2021-44077)	Checks for exploitability of CVE-2021-44077 by uploading a malicious payload through that API, and execute the payload through another API. This upload is likely to be caught by AV software on the host. If successful, this exploit will leave behind a file msiexec.exe in the ManageEngine\ServiceDesk\site24x7 folder.	none
Subdomain Takeover	Proactively takeover and hold onto subdomains that are vulnerable to subdomain takeover (H3-2021-0002) to prevent bad actors from compromising them first.	none
VMWare vCenter Server Access Control Vulnerability (CVE-2020-3952)	Checks for exploitability of CVE-2020-3952 by adding an administrative user and removing it afterwards.	none
VMWare vCenter Server Plugin Remote Code Execution Vulnerability (CVE-2021-21972)	Checks for exploitability of CVE-2021-21972 by installing a webshell, executing a command within the webshell, and removing it afterwards. For vCenter servers running on Linux,	none

Name	Description	Risk
	it is possible that randomly-named webshells will be left behind on the vulnerable vCenter server if the exploit fails.	
VMWare vRealize Operations Manager SSRF Vulnerability (CVE-2021-21975)	Checks for exploitability of CVE-2021-21975 and CVE-2021-21983 by installing a randomly named webshell, executing a command within the webshell, and removing it afterwards. Cleanup of the webshell may fail in exceptional cases.	none
Zoho ManageEngine ADSelfService Plus API Auth Bypass (CVE-2021-40539)	Checks for exploitability of CVE-2021-40539 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none

EXPLOITATION

Attempt exploitation of a vulnerability to confirm that it can be exploited by NodeZero.

Name	Description	Risk
Bluekeep (CVE-2019-0708)	Tests exploitability of the Bluekeep vulnerability (CVE-2019-0708). There is a moderate-level risk this exploit may crash the target host, and it is not recommended for use against production systems.	high
Cisco Smart Install Vulnerability (CVE-2018-0171)	Tests exploitability of the Cisco Smart Install vulnerability (CVE-2018-0171). The test attempts to pull router config from the vulnerable router via the TFTP protocol. Against a few older models of Cisco routers, running this exploit may cause the router to reload or go down.	moderate
EternalBlue (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerability EternalBlue. This is a kernel buffer overflow exploit and carries a moderate risk of crashing the target. It is not recommended for use against production systems. This exploit is only attempted if NodeZero is able to reliably determine the target operating system and NodeZero is not able to first exploit EternalChampion/EternalSynergy/EternalRomance.	moderate
EternalChampion/ EternalSynergy/ EternalRomance (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerabilities EternalChampion, EternalSynergy, and EternalRomance.	low
Exploding Can (CVE-2017-7269)	Tests exploitability of the IIS 6.0 WebDAV vulnerability CVE-2017-7269, aka Exploding Can.	low
HP iLO Web API Remote Code Execution (CVE-2017-12542)	Tests exploitability of the HP iLO Web API Remote Code Execution vulnerability (CVE-2017-12542). The test attempts to retrieve users and their credentials by exploiting a heap-based buffer overflow.	low
Heartbleed (CVE-2014-0160)	Tests exploitability of the Heartbleed vulnerability (CVE-2014-0160), if discovered by NodeZero. This test dumps memory from the vulnerable server.	low
Server Service Vulnerability (MS08-067)	Tests exploitability of the Windows SMB remote code execution vulnerability CVE-2008-4250, aka MS08-067. There is a high likelihood that this exploit will crash the SMB service on the target after successful exploitation.	high

HASH CRACKING

Properties related to cracking hashes found in the environment.

Name	Description	Risk
Automatic Hash Cracking	Automatically attempt to crack hashes found in the environment.	none

MAN IN THE MIDDLE ATTACKS

Properties related to modules that conduct man-in-the-middle (MITM) attacks.

Name	Description	Risk
Expanded LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This will sniff all available traffic regardless of scope.	none
Limited LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This is limited to the scope provided during the configuration of the pentest. If selected, this option overrides the 'Expanded LLMNR and NetBIOS poisoning' option.	none
Net-NTLM Authentication Coercion	Enables Net-NTLM Authentication coercion techniques. This allows attackers to capture Net-NTLM (NTLMv2) hashes by coercing machines to authenticate to an attacker controller server.	none
Net-NTLM Hash Relaying	Enables SMB relay attacks. This allows attackers to gain unauthorized access to machines by capturing Net-NTLM (NTLMv2) hashes over the network and relaying them to target SMB servers.	none

POST-EXPLOITATION

Properties related to actions taken after compromising a host

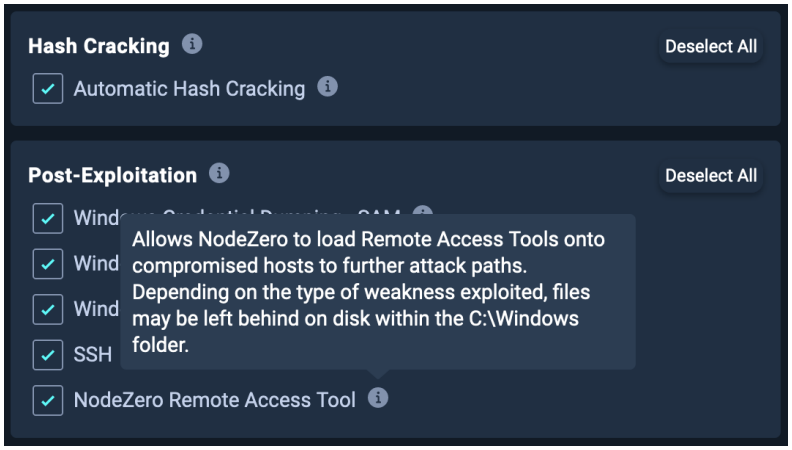
Name	Description	Risk
SSH	Enables post-exploit actions such as system enumeration and privilege escalation on hosts for which SSH access was gained. In exceptional circumstances, files may be left on disk in the /tmp folder.	none
Windows Credential Dumping - LSA Secrets	Enables dumping of credentials from the Local Security Authority (LSA) after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	none
Windows Credential Dumping - LSASS	Enables dumping of credentials stored in the Local Security Authority Subsystem Service (LSASS) process, after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	low
Windows Credential Dumping - SAM	Enables dumping of credentials from the Security Account Manager (SAM) database after gaining administrative access. In exceptional circumstances, cleanup may fail, leaving files on disk.	none

Remote Access Tool

Paid Feature

Limited Access Feature Advanced post-exploitation and lateral movement are only available to paid clients

NodeZero can leverage detected weaknesses and vulnerabilities to deploy Remote Access Tools (RATs). These tools provide additional access that NodeZero uses to further explore attack paths during operations. This functionality is enabled by default but can be disabled under the `Post-Exploitation` section in the advanced configuration settings by deselecting `NodeZero Remote Access Tool`.



NodeZero coordinates these RATs to conduct more extensive post-exploitation on hosts, including common MITRE ATT&CK techniques such as system information gathering and credential dumping.

WHAT IS A REMOTE ACCESS TOOL?

Remote Access Tools are applications that attackers use to remotely administer an endpoint. These tools fall into different categories, including legitimate administrative tools like AnyDesk, as well as specially designed tools meant to evade antivirus and EDR products, such as Cobalt Strike, Meterpreter, and Sliver.

NodeZero utilizes a specially developed internal tool that offers many of the same capabilities as the aforementioned adversary emulation tools.

HOW WILL I KNOW IF NODEZERO DEPLOYED A RAT DURING AN OPERATION?

During an operation, NodeZero will generate an event in the Real-Time View that includes the host to which it attempted to deploy the RAT, along with any associated artifacts that might be present, such as a filename.

After an operation, the host's "Activity Logs" will contain the same type of information.

ARE THERE RISKS ASSOCIATED WITH ENABLING THIS FEATURE?

No, the tooling that NodeZero employs is a User Mode RAT, which means it will not introduce system instability when used. Any post-exploitation activities conducted, such as credential dumping, use safe tactics to avoid causing any system disruptions.

DOES THE RAT LEAVE ANY ARTIFACTS AFTER AN OPERATION?

The RAT makes significant efforts to clean up itself and any artifacts during and after an operation. When the operation ends, all deployed RATs are uninstalled from all systems. However, in rare circumstances, some artifacts may be left on disk. If any artifacts are left, they can most commonly be found in:

- `C:\Windows\Temp\` or `/tmp/` - Some weaknesses exploitation and credential dumping activities may momentarily create a file with a random name in this directory.
- `C:\Windows\System32\spool\drivers\x64\` - A file may be created here when exploiting common print service issues.

3.3.7 BloodHound

BloodHound

BloodHound is an open source toolset to collect and analyze relational data within an Active Directory and/or Azure environment. BloodHound uses graph theory to reveal hidden and often unintended relationships within these environments to allow Attackers and Defenders to identify highly complex attack paths that would otherwise be impossible to quickly identify.

BloodHound has become an industry standard tool for both Red Teams and Blue Teams to attack and defend Active Directory environments. For details on how to use the BloodHound tool, see BloodHound's Official Documentation.

> **Compatibility Note**
>
> As of May 2024, NodeZero now supports BloodHound CE v5.8+.

INSTALLING AND SETTING UP BLOODHOUND

Please see BloodHound's Installation documentation:

- Install BloodHound CE with Docker Compose

HOW DOES NODEZERO USE BLOODHOUND?

After NodeZero discovers and verifies a domain user credential, it will utilize a BloodHound data collector to gather information on the Active Directory or Entra environment. NodeZero stores this data in a neo4j 4.4.x graph database in our ephemeral architecture during the life of the pentest, and will utilize it to identify complex attack paths that may lead to compromising the domain or Entra ID tenant. After the pentest finishes, the BloodHound data is backed up and stored for a limited time. H3 customers who wish to utilize NodeZero's BloodHound collections to inform their own Red/Blue/Purple team operations can request the data from a pentest for a **limited time**.

OBTAINING NODEZERO'S BLOODHOUND DATA

The ability to download NodeZero's collected BloodHound data is a paid feature, unavailable on trials. If you would like to request access to this feature, please contact H3 Customer Success.

USING A NODEZERO PENTEST'S BLOODHOUND DATA

NodeZero provides BloodHound data in the form of a neo4j backup dump file. Users can use this file to directly load the data into the neo4j database they connect the BloodHound GUI to.

Using `neo4j-admin` to Import NodeZero's BloodHound Dump to Neo4j

Neo4j provides an administrative command-line tool called `neo4j-admin` to manage/administer its Database Management System (DBMS). This tool is typically located in the neo4j `bin` directory. `neo4j-admin`'s `load` command loads the archive file that NodeZero produces. The command can be run from an online or an offline neo4j DBMS. Typically, the `neo4j-admin load` command should be run as the `neo4j` user to ensure appropriate file permissions.

```
$neo4j-admin load --database=neo4j --from=<DUMP_FILE_PATH>
```

Reference: neo4j 4.4 - Restore a database dump

BloodHound CE provides an environment variable called `NEO4J_DATA_MOUNT` that can be used to mount a neo4j data directory. Running the following command will extract the neo4j dump archive to a data directory that can be passed to `NEO4J_DATA_MOUNT` in the current directory.

```
docker run -v <PATH_TO_DUMP_FILE>:/dump.dump -v $PWD/data/:/data --rm  neo4j/neo4j-admin:4.4 neo4j-admin load  --database=neo4j --from=/dump.dump
```

Now set the NEO4J_DATA_MOUNT environment variable and start bloodhound CE.

```
export NEO4J_DATA_MOUNT=$PWD/data
curl -L https://ghst.ly/getbhce | docker compose -f - up
```

3.3.8 Injecting Credentials

Injecting Credentials

NodeZero can run pentests from a compromised user perspective. This type of perspective shows the impact an attacker would have if leveraging a specific set of assumed compromised credentials.

Users can run an authenticated pentest by injecting credentials into the pentest via the Real-Time View, as described below.

NodeZero uses injected credentials in ways that emulate how an attacker may use credentials they compromise. This feature allows users to execute "what if" scenarios to see what impacts may result from compromised credentials.

WHY INJECT CREDENTIALS?

Injecting credentials can reveal the risks posed by scenarios like the following.

What if employee X was phished?

No matter how advanced our network's technological defenses, humans have been and will continue to be a popular attack vector through which attackers can gain initial access. Whether through phishing attacks or other forms of social engineering, we should expect that user credentials may fall into an attacker's hands.

If your organization performs phishing exercises you may identify a set of credentials that are prone to being phished. By injecting their credentials into a pentest NodeZero can generate a complete picture of the potential impacts of a successful phishing attack.

What if employee X goes rogue?

We all like to believe we can trust our employees and co-workers, but at Horizon3 we encourage our users to ~~Trust but~~ Verify. It is important to implement and verify access policies that use the concept of least privilege: users and service accounts should only have access to the resources to which they need access.

By injecting a credential for a user or service account into a pentest, NodeZero can generate a complete picture of what resources that account has access to.

HOW TO INJECT CREDENTIALS

Users can inject credentials through the Real-Time View immediately after launching a pentest.

Demo Pentest
Status
Preparing
Type
Internal Pentest
Created By
John Doe
Scheduled
08/22/2024, 02:38:42 AM UTC
Duration
-
Maximum Runtime
30d
Op State
provisioning
Altering configurations of hosts or services in the target network during the pentest can cause unexpected results.
Preparing Pentest
Your pentest is setting up the resources it needs.
Copy and run this one-time command on your Docker Host to install and run NodeZero.
Copy
One-time curl script expires in ~11 Hours and ~20 Minutes
Show Me How
Findings
0 Hosts
0 Weaknesses
0 Potential Weaknesses
0 Credentials
0 Potential Credentials
Credentials
Inject Credentials
This test is running from a different perspective. As you previously injected a credential for this scope, for the best comparisons between pentests, we recommend you inject the same credential(s) to ensure consistent results and trends by running from the same perspective. Read More
No credentials have been injected.
NodeZero Host
Last Contact
-
System
-
Node
-
Release
-
Version
-
Machine
-

Click the `Inject Credentials` button to open the Inject Credentials modal. In the modal, choose a credential type from the `Add Credential` drop-down to add a new credential. The supported credential types are shown in the table below.

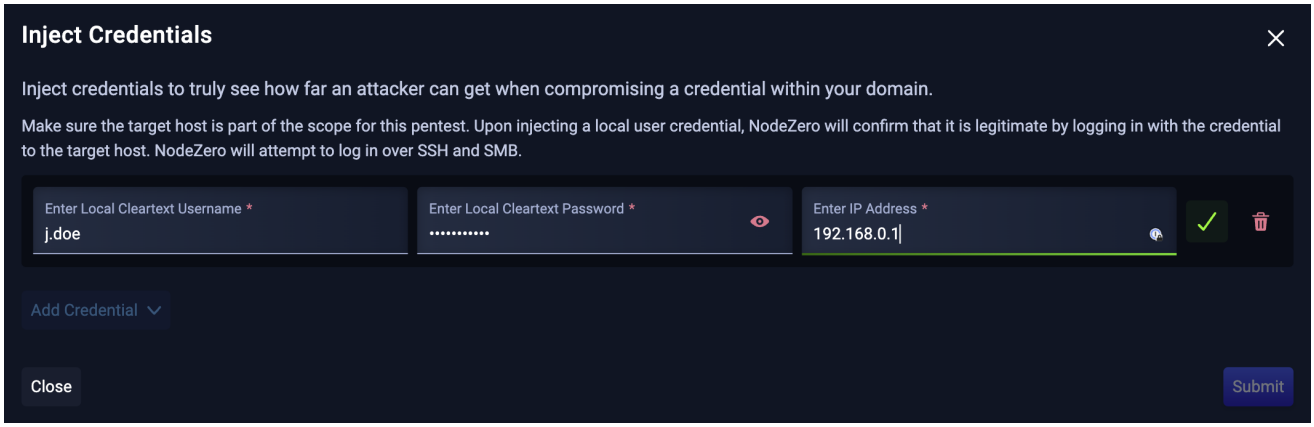
Type	Description	Example
Domain User: Cleartext	Cleartext credentials for an Active Directory domain user. If there is not a domain controller in scope, NodeZero will not attempt to use this credential. An attacker may compromise this type of credential through various means including phishing, social engineering, key logging, or password guessing.	Username: `john.doe` Password: `MyPassword123`
Domain User: NTLM Hash	The NTLM Hash for an Active Directory domain user. If there is not a domain controller in scope, NodeZero will not attempt to use this credential. An attacker may compromise this type of credential if they were able to dump the SAM or NTDS database on a domain controller.	Username: `jdoe` Hash: `31d6cfe0d16ae931b73c59d7e0c089c0`
Local User: Cleartext	Cleartext credentials for a local Windows or Linux user. These credentials include the IP address of the local machine and will be used to attempt login over SSH and SMB. An attacker may compromise this type of credential through various means including phishing, social engineering, key logging, or password guessing.	Username: `jdoe2` Password: `MyPassword123` IP Address: `10.0.0.1`
Local User: NTLM Hash	The NTLM Hash for a local Windows user. These credentials include the IP address of the local machine and will be used to attempt logins over SMB. An attacker may compromise this type of credential if they are able to dump the SAM database on a local Windows machine.	Username: `Administrator` Hash: `31d6cfe0d16ae931b73c59d7e0c089c0` IP Address: `192.168.0.1`
AWS User: Access Keys	An AWS access key and secret access key. By injecting an AWS credential, all cloud resources belonging to the associated AWS ID will be considered in scope. An attacker may compromise this type of credential by finding it on a compromised machine or file share, as they are commonly stored in files in the user's directory.	Access Key ID: `AKIASP2TPHJSVM75TWVN` Secret Access Key: `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`
Azure	The username and password of an Azure user. NodeZero will authenticate to Azure using these credentials. The pentest will include all cloud resources and services that this resource can access. An attacker may compromise this type of credential through various means including phishing, social engineering, key logging, or	Enter Entra ID username in UPN format: `john.doe@example.com` Enter Entra ID Password: `MyPassword123`

Type	Description	Example
	password guessing. Note: Microsoft has deprecated the use of non-MFA enabled accounts. When Microsoft fully prohibits their use, Azure non-MFA credentials will no longer be a valid credential type for injection. For details, see Microsoft's Azure Mandatory Multifactor Authentication: Phase 2 blog post.	
Azure MFA	NodeZero will authenticate to Azure using an Azure MFA device code with the Azure Tenant ID provided. You will receive an email alert when it is time to enter your MFA code from the Real-Time View page. The pentest will include all cloud resources and services that this resource can access. For more detailed instructions, see Injecting an Azure MFA Credential. Need help finding your Tenant ID? See Microsoft's How to find your Microsoft Entra tenant ID topic for step-by-step instructions.	Tenant ID: `aaaabbbb-0000-cccc-1111-dddd2222eeee`
Azure Service Principal Certificate	NodeZero will authenticate using a service principal certificate (in `.pfx` format) that you provide. You also need to provide the Tenant ID and Application ID. If the `.pfx` cert is password-protected, you need to provide its password. The pentest will include all cloud resources and services that this resource can access. An attacker may compromise this type of credential by finding it on a compromised machine. Need help finding your Service Principal's Application ID in the Entra portal? See Microsoft's Application and Service Principal Objects in Microsoft Entra ID topic. Need help finding your Tenant ID? See Microsoft's How to Find Your Microsoft Entra Tenant ID topic for step-by-step instructions.	Tenant ID: `aaaabbbb-0000-cccc-1111-dddd2222eeee` Application ID: `00001111-aaaa-2222-bbbb-3333cccc4444` .pfx file: `myserviceprincipalcert.pfx` Password (if the .pfx certificate is password-protected): `MyPassword123`

Tip

While you cannot inject an AWS Role in the Real-Time View, there is another way to accomplish this. See: Injecting an AWS Role.

You also have the option to inject credentials on a Runner assigned to an internal pentest.



After entering the credential details, click the green checkmark. Add more credentials as desired, then click `Submit` to send the credentials to NodeZero.

You can view the status of credentials in the Real-Time View, using the icons next to each credential. Credentials states are shown below:

Icon	State	Description
	Pending	The credential has been submitted but has not yet been received by NodeZero. This state may occur when a pentest first begins until the ephemeral infrastructure is fully deployed.
	Received	The credential has been received by NodeZero. If NodeZero has attempted to use the credential but authentication was unsuccessful, it will remain in the Received state.
	Confirmed	The credential has been used by NodeZero to successfully authenticate within the target environment.

Tip

You may continue to inject credentials in the Real-Time View until the pentest completes and enters the Processing state. Injecting credentials may extend the duration of a pentest, but the best way to minimize the pentest operation runtime is to inject credentials early in the pentest.

FAQ

Is it safe to inject credentials in NodeZero?

Yes. Horizon3 takes the security of injected credentials seriously.
Once injected, credentials are securely transferred to NodeZero's ephemeral environment, which is dedicated to a single pentest and is destroyed once the pentest is completed. Sensitive parts of credentials (e.g. plaintext passwords, hashes, private keys, etc.) are never stored in persistent databases.

How many credentials can I inject?

NodeZero supports injecting up to twenty total credentials per pentest.

Are injected credentials used when re-running a pentest?

No. Since injected credentials are destroyed at the end of a pentest, NodeZero cannot use them when re-running a pentest.
However, in the Real-Time View, you can see descriptions of the credentials that were used in the previous pentest. We encourage users to re-enter these credentials to get a comparable pentest experience.

Does injecting credentials increase the duration of a pentest?

There are many factors that affect pentest duration, the most significant of which are number of live hosts, services, and web applications. When credentials are injected, NodeZero will attempt to authenticate with the credentials, and if successful will perform various post-exploitation tasks such as enumerating shares and dumping credentials on compromised hosts, which can lead to further discoveries that may extend the duration of the pentest.

The best way to minimize the pentest runtime is to inject credentials early in the pentest.

Injecting an AWS Role

AWS roles can be used to assign permissions to various AWS services. Injecting an AWS role can help understand the impact of one of these roles being compromised.

> **Example**
>
> What would happen if someone hacked into our web application running on that EC2 instance?

Users can inject an AWS role into a pentest by running NodeZero on an EC2 instance and assigning that EC2 instance an IAM instance profile.

When the pentest starts, NodeZero will query the IMDS and treat the EC2 instance role as an injected credential. By injecting an AWS credential, all cloud resources belonging to the associated AWS ID will be considered in scope.

Since an EC2 instance can only have one IAM instance profile assigned, and because an IAM instance profile can only contain one role, it is only possible to inject one AWS role per pentest.

CREATING AN IAM INSTANCE PROFILE

An instance profile is what attaches an AWS role to an EC2 instance.

If you create an AWS Role in the AWS Management Console for an EC2 instance (assign the EC2 service as the trusted entity) an IAM instance profile with the same name will also be created. This means that often you do not need to manually create an IAM instance profile.

If you created the AWS Role with the AWS CLI or if the role was not created for an EC2 instance you will need to manually create an IAM instance profile.

To manually create an IAM instance profile you will need to use the AWS CLI. The commands below show how you would create an IAM instance for a role named `ExampleRole`.

```
aws iam create-instance-profile --instance-profile-name ExampleRole
aws iam add-role-to-instance-profile --instance-profile-name ExampleRole --role-name ExampleRole
```

> **Tip**
>
> The role assigned to the instance profile must have the EC2 service specified in the trust relationship policy.

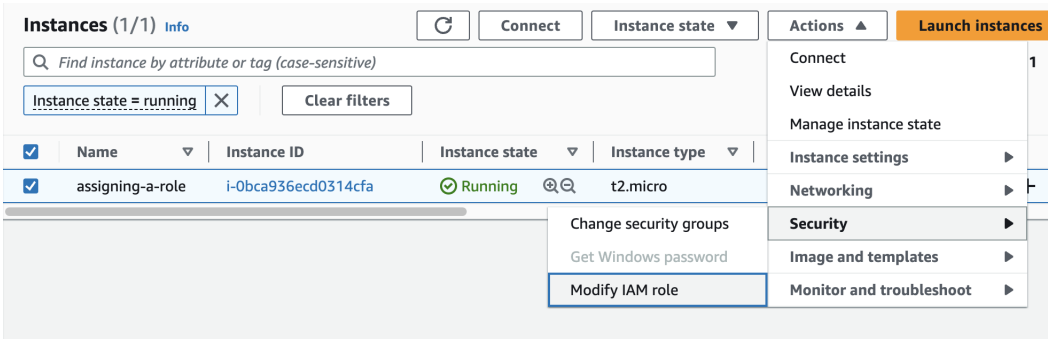
ASSIGNING AN IAM INSTANCE PROFILE TO AN EC2 INSTANCE

Method 1 - AWS Management Console - Assign Instance Profile when Launching a New Instance

Launch a new EC2 instance using the `Launch an instance` flow, find the `Advanced details` section and select an IAM instance profile from the drop-down.

Method 2a - Assign Instance Profile to an Existing EC2 Instance - AWS Management Console

From the `EC2 Dashboard` or from the `Instance Details` page select an EC2 instance, click the `Actions` drop-down menu, click the `Security` sub-menu, and finally select `Modify IAM Role`.



From the `Modify IAM role` page, select the IAM role you want to inject from the drop-down and click `Update IAM role.`

Method 2b - Assign Instance Profile to an Existing EC2 Instance - AWS CLI

AWS CLI Command:

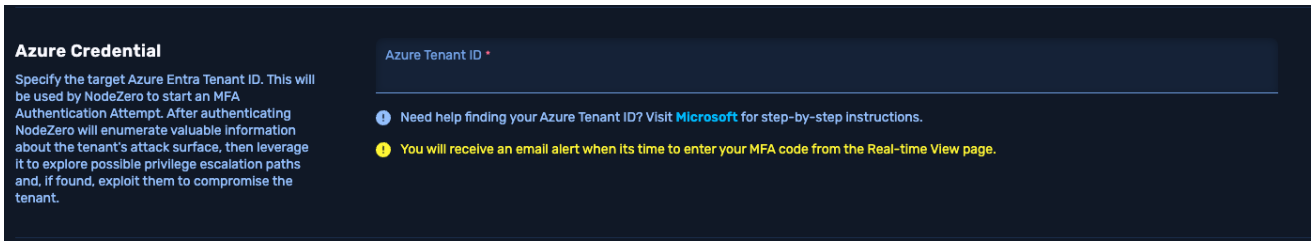
```
aws ec2 associate-iam-instance-profile --instance-id i-123456789abcde123 --iam-instance-profile Name=admin-role
```

Injecting an Azure Entra ID MFA Credential

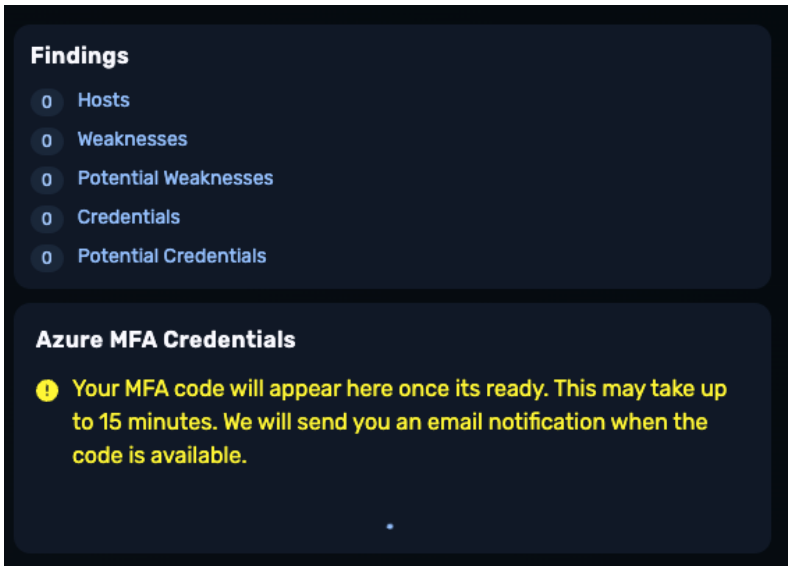
In October 2024, Microsoft began enforcing mandatory Multifactor Authentication (MFA) for some Azure services and resources. In order to maintain support for Azure credential injection, NodeZero now supports OAuth 2.0 Device Code Flow to authenticate and authorize the use of an Entra ID credential for pentests.

INJECTING AN AZURE MFA CREDENTIAL FOR THE AZURE ENTRA ID PENTEST

When configuring an Azure Entra ID Pentest, users will now be prompted to provide their Azure tenant ID instead of a cleartext username and password. NodeZero will utilize the tenant ID to begin the Device Code Flow once the pentest starts.



After the pentest is configured and begins provisioning, the user will automatically be re-directed to the Real Time View (RTV) and will see an "Azure MFA Credentials" box indicating they will be notified once an MFA code is available for them to finish authentication.



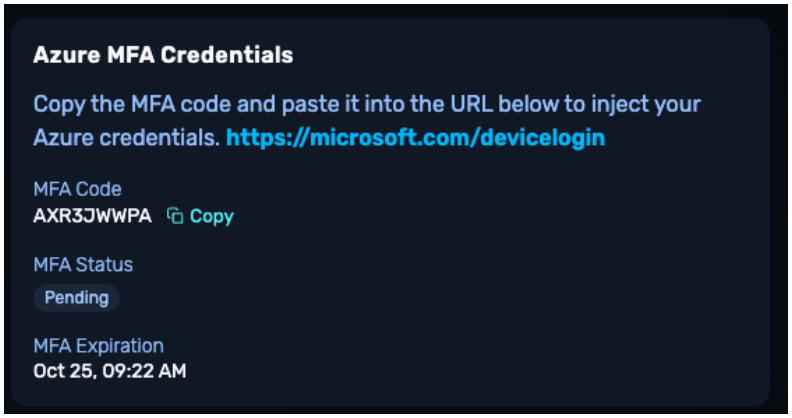
Once the pentest begins, NodeZero will start the Device Code Flow and will display a MFA code that the user will use to complete the authentication/authorization. Users will also receive an email notification that the MFA code is available in the pentest's RTV.

INJECTING AN AZURE MFA CREDENTIAL FROM THE REAL TIME VIEW (RTV)

User's can inject an Azure MFA Credential utilizing the "Inject Credentials" button on a pentest's RTV page. After selecting "Azure MFA" from the dropdown menu, user's will be prompted for their Azure Tenant ID. Once submitted, it may take approximately 2 minutes during a live/running pentest for NodeZero to provide the required MFA code from Microsoft.

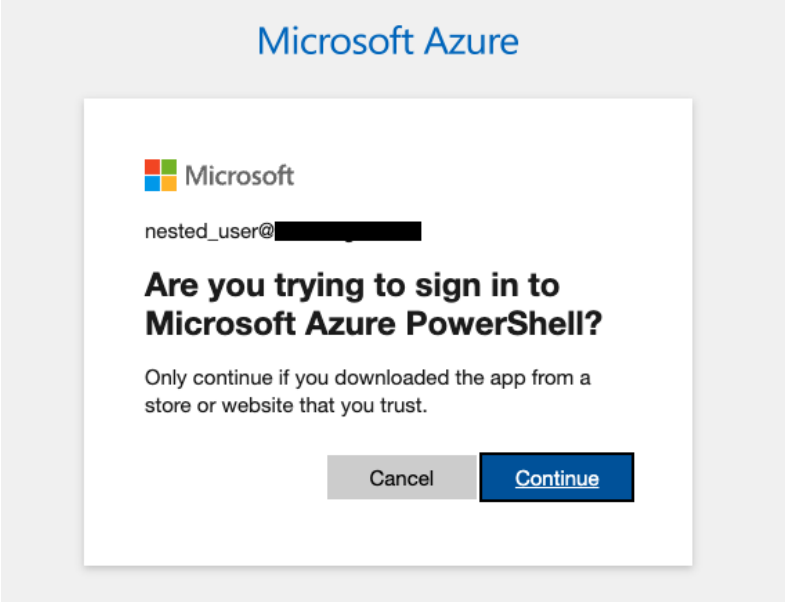
COMPLETING AUTHENTICATION/AUTHORIZATION

Once NodeZero receives the MFA code from Microsoft, users have 15 minutes to return to the pentest's RTV in the NodeZero Portal and complete Authentication/Authorization. If a user fails to return to the NodeZero Portal to complete authentication/authorization, NodeZero will automatically re-request a new MFA code two additional times before the user must manually initiate the Device Code Flow again.



The user must copy the MFA code and navigate to the authentication URL provided: https://microsoft.com/devicelogin.

Once the user interactively authenticates to Microsoft, they will be prompted to authorize NodeZero access to the tenant.



Note

NodeZero utilizes the client id for "Microsoft Azure Powershell" when requesting authorization for the Device Code Flow. A lot of common attacker capabilities utilize this client ID, as Microsoft Azure Powershell, being a first party Microsoft utility, has default delegated permissions to several valuable resources such as the MS Graph API. NodeZero does **NOT** use this client id for password spray or user enumeration activities.

Once Authorization is granted, NodeZero will receive tokens for the selected identity to access Azure resources.

Azure MFA Credentials
Copy the MFA code and paste it into the URL below to inject your Azure credentials. https://microsoft.com/devicelogin
MFA Code
AXR3JWWPA Copy
MFA Status
Success
Credentials
Inject Credentials
This test is running from a different perspective. As you previously injected a credential for this scope, for the best comparisons between pentests, we recommend you inject the same credential(s) to ensure consistent results and trends by running from the same perspective. Read More
administrator
Cleartext
e7a4dd4f-5269-46ec-b904-2b30e0c87497
Azure MFA

3.3.9 Schedules

Automate Scheduling of Pentests

A common use case is running pentests automatically on a recurring basis, for example once a week or once a month. This enables continuous autonomous pentesting without any user intervention - no need to sign-in to the Portal to create the pentest, no need to copy+paste the NodeZero launch script to deploy it.

The instructions below walk through how to create a pentest schedule in the Horizon3.ai Portal.

> **Looking for scheduling via CLI instead of Portal?**
>
> Refer to our `h3-cli` guide on Scheduling from CLI.

HOW IT WORKS

Creating a pentest schedule involves three steps:

1. Select a pentest template
2. Specify the schedule
3. Verify the pentest schedule is working as expected (optional)

Click the **Create Schedule** button in the Portal to get started.

1. Select a pentest template

A pentest template specifies the scope and attack configuration for a pentest. Pentest templates can be created within the **Run Pentest** wizard in the Portal. Note that you can create a template in the wizard without having to run the pentest.

NodeZero Runner required for internal pentests

Creating a pentest schedule for internal pentests requires first setting up a NodeZero Runner. A NodeZero Runner handles the automated deployment of NodeZero on your NodeZero Host within your internal network.

Once a NodeZero Runner has been set up, you can save it to your pentest template, or you can select it when you create the pentest schedule.

2. Specify the schedule

Once you have created and selected your pentest template, the next step is to specify when you want the pentest to run. You have two options for specifying the schedule:

1. Create a simple weekly or monthly schedule using the convenient scheduling form in the **Create Schedule** wizard.
2. Specify a CRON expression for greater precision and control over the schedule.

CRON expressions are an industry-standard way for specifying a recurring schedule. CRON expressions were originally developed for use with the cron job-scheduling service on Unix systems, but they are now widely supported by a number of modern job-scheduling services.

See the section on CRON expressions below for examples and more information.

Schedule precision: pentest schedules support a **minimum precision of 1 hour**. For example you can schedule a pentest to run at 5:00pm, but you cannot schedule it to run at 5:30pm. Therefore the `minute` field in your CRON expression (the first field) must always be 0.

Schedule time zone: scheduled times use UTC time zone. This applies to both the scheduling form and CRON expressions.

3. Test and verify the pentest schedule (optional)

Now that you have created a pentest schedule, you can verify everything is working as expected by clicking the **Trigger Now** button. This will trigger the schedule to run immediately, rather than waiting for its normally scheduled time.

The **Trigger Now** feature gives you an opportunity to test the entire flow and address any issues right away instead of discovering them later when the schedule is normally triggered.

If you don't want to actually run the pentest right now, you can cancel it from the Portal once it gets started.

ENABLING AND DISABLING A PENTEST SCHEDULE

You can enable or disable a pentest schedule at any time by toggling its status in the Portal.

TROUBLESHOOTING

If a pentest schedule fails to execute at its normally scheduled time, you will receive an email notification alerting you about the error. The error message is also surfaced on the pentest schedule in the Portal.

For further assistance, contact H3 support via the chat icon in the Portal.

CRON EXPRESSIONS

CRON expressions are an industry-standard way for specifying a recurring schedule. A CRON expression is structured as a series of 5 elements:

```
{minute} {hour} {day-of-month} {month} {day-of-week}
```

There are a number of articles that document CRON expressions and outline how to build them.

Below are several examples of CRON expressions for various time intervals - weekly, monthly, quarterly, etc. You can use these examples as a guide for building your own CRON expressions.

You can verify your CRON expression by plugging it into an online utility like cronhub or crontab.guru.

- `0 8 1 * *`: runs on the first of every month, at 8am UTC.
- `0 8 * * THU`: runs every Thursday (weekly) at 8am UTC.
- `0 8 * * THU#2`: runs on the 2nd Thursday (`THU#2`) of every month.
- `0 8 1 */3 *`: runs on the 1st day of every 3rd month (quarterly).

Running a one-off pentest at a specific date and time

It is not possible to specify a CRON expression that runs only once. Therefore it is not possible to create a pentest schedule that runs only once. However you can approximate this behavior by setting the CRON expression to a specific date and time, then disabling the schedule after the pentest completes.

For example, to schedule the pentest to run at 8am UTC on June 1st, use the following CRON expression:

```
0 8 1 JUN *
```

Note this will run the schedule every June 1st, annually. So if you want to run the pentest only once, disable the schedule after the pentest completes.

References:

1. h3-cli on GitHub
2. Horizon3.ai Portal

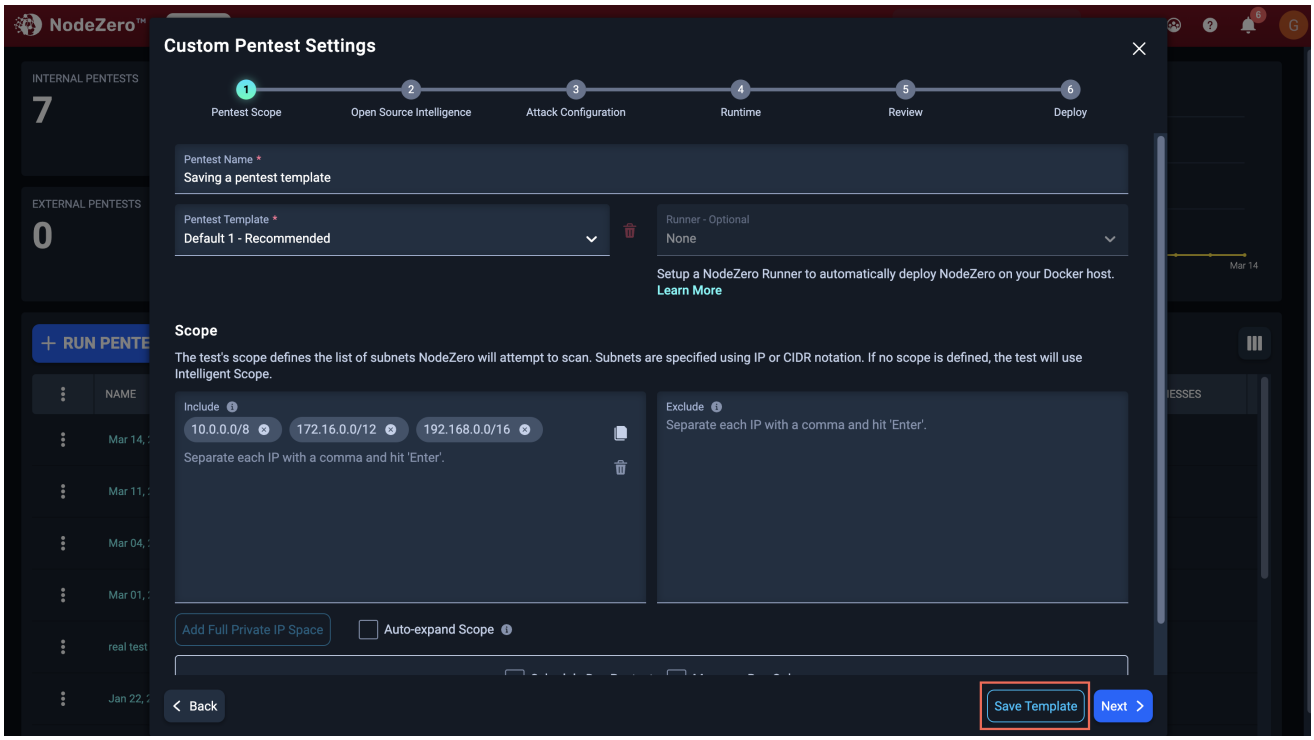
3.3.10 Pentest Templates

Templates

Templates allow you to save a pentest configuration and then use that saved configuration when running a one-off pentest or when scheduling repeated pentest runs. You can save or load templates from within the Run Pentest wizard, or using the template management page in the NodeZero Portal.

TEMPLATES IN THE RUN PENTEST WIZARD

When you define a one-off pentest using the Run Pentest wizard, you can click the `Save Template` button at the bottom of the wizard to store your test configuration in a template for future use:



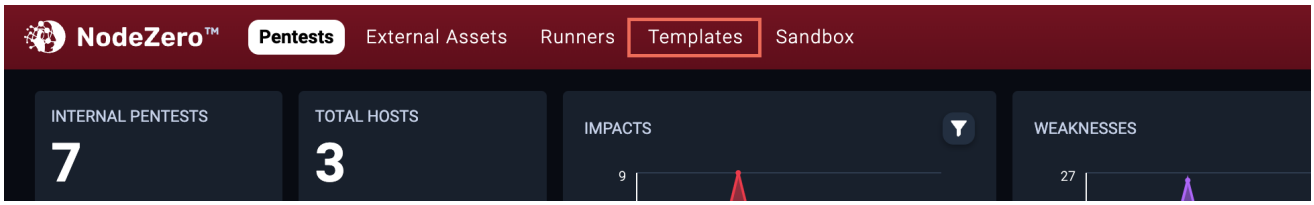
If you want to use a previously-saved template when defining a test in the wizard, use the `Pentest Template` dropdown on the first page of the Run Pentest wizard:

You can also delete a template by clicking the trash icon near the `Pentest Template` dropdown.

TEMPLATE MANAGEMENT PAGE

Although you can access or create templates from within the Run Pentest wizard, you may want to see all of your templates and manage them in one place. For this, you can use the template management page. This page lists all of your templates and lets you duplicate or delete them. You can edit templates you saved earlier, and create new ones.

To access the template management page, click the `Templates` link in the navigation area at the top of any NodeZero Portal page:

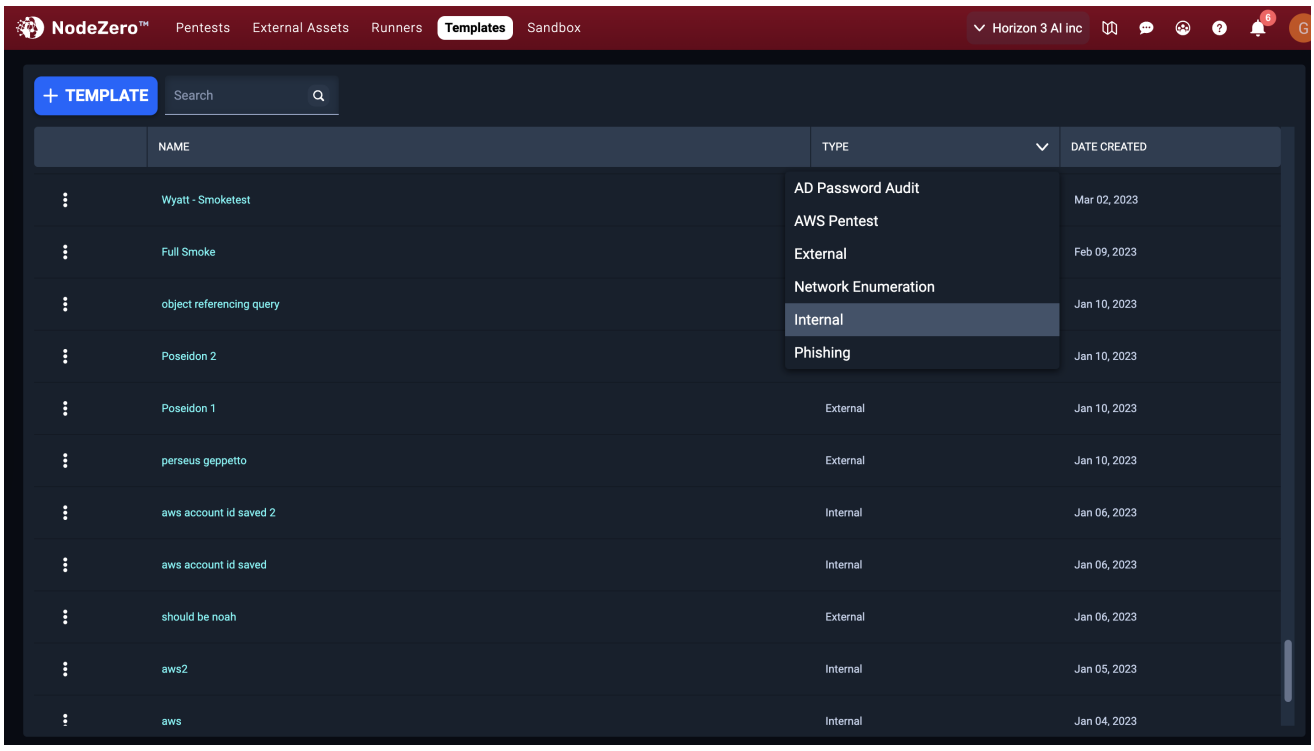


The template list

The template list shows you all of your templates in one place:

You can click on each column header to adjust the sort order for the table. You can also search template titles by entering some text in the search box at the top of the table.

By default, the template table shows templates belonging to any test type. You can filter this list to only show a specific test type by clicking the filter arrow in the `Type` column:



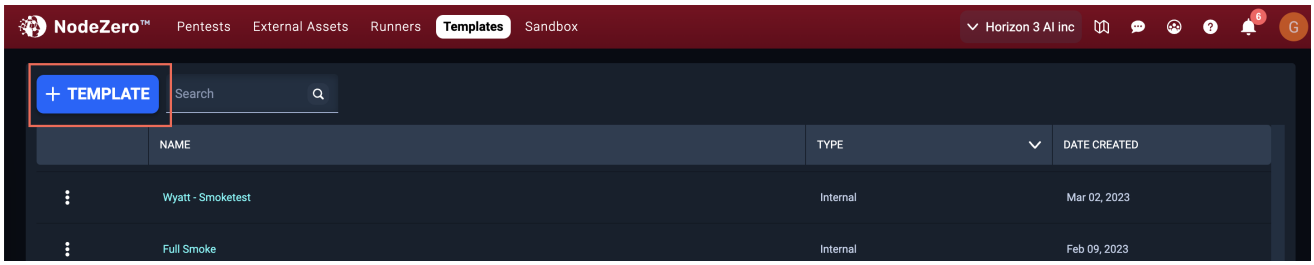
Within the table, you can duplicate or delete any template row by opening the action menu for that row and selecting the corresponding action:

Template names are unique

Template names must be unique within your organization. If you attempt to create or duplicate a template with a name that is already used by an existing template, you'll receive a warning. You can choose to overwrite the original template, or cancel.

Creating templates

You can create a new template from the template management page by clicking on the `+ Template` button near the top:

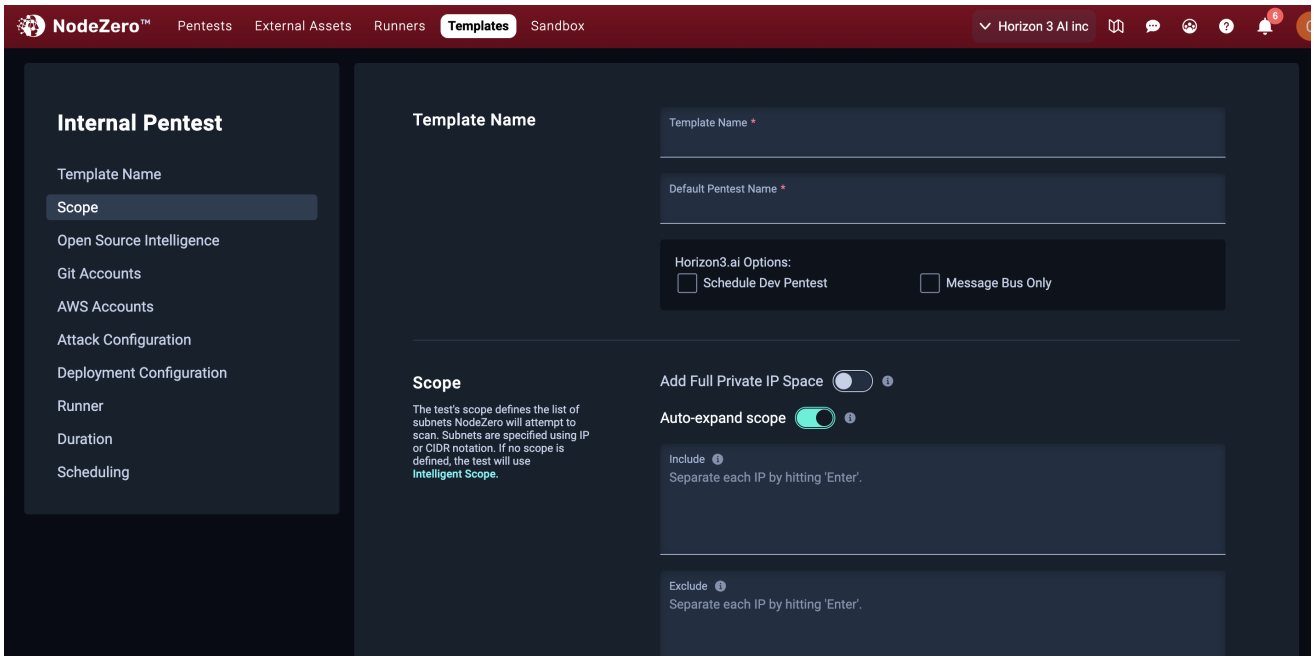


This will open the template type selection dialog, where you can select which type of template you want to create:

After selecting a type, you will see a form where you can input all of the configuration for this template. This configuration will be used when starting pentests based on this template. For more about running pentests from templates, see Running pentests using templates.

You can use the navigation links on the left to jump to different parts of the form. Every template must have a name and a default pentest name (which is used as a default name when running pentests from this template).

Depending on the test type you selected for this template, the form will have different fields. For example, here's the form for an Internal template:



Fill out the form as needed for your testing requirements, then click `Create` at the bottom of the form.

If there were any issues with the form values, you'll see validation errors on the form. Otherwise, you will be taken back to the template list. Your new template will be ready to use from the pentest wizard.

> **Template names are unique**
>
> Template names must be unique within your organization. If you attempt to create or duplicate a template with a name that is already used by an existing template, you'll receive a warning. You can choose to overwrite the original template, or go back and change the name.

For more information about the actual test configuration parameters and what they do, see the test documentation.

Editing templates

You can edit the configuration for an existing template. To do this, find the template you want to edit in the list and click on it. This will take you to the edit page for that template, which is similar to the page for creating templates. The main difference is that the template's type and name cannot be changed after they were first saved.

Make any required changes to this template and then click `Update` at the bottom of the page. You'll be returned to the template list once this has completed.

RUNNING PENTESTS USING TEMPLATES

The purpose of having a template is so that you can easily start pentests using a template's predefined configuration. There are two ways to do this: manually, or on a schedule.

To manually start a pentest using a template, use the Run Pentest wizard, as described earlier in this document.

You can also run pentests regularly by setting up a schedule. For more information, see this page.

3.3.11 Advanced Data Pilfering

Advanced Data Pilfering (ADP)

Compatibility

Internal Pentesting • Elite package or RBVM add-on.

OVERVIEW

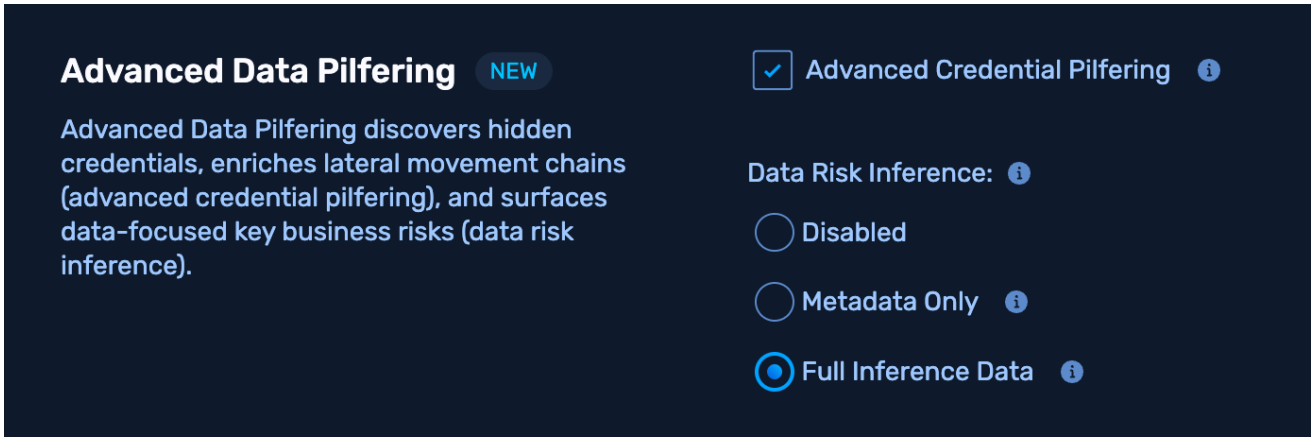
Advanced Data Pilfering (ADP) extends NodeZero's autonomous attack engine to identify and contextualize sensitive business data encountered during internal operations-including phishing and insider threat scenarios. It combines:

- **Advanced Credential Pilfering (ACP)**
- **GenAI-powered Data Risk Inference**

Findings are auto-tagged with **Business Risk categories**, prioritized with **context scoring**, and displayed across the UI, reports, and attack graphs.

HOW TO USE

- **Step 1:** While creating an internal or phishing pentest, or an insider threat test, scroll to the Advanced Data Pilfering section:



- **Step 2:** Choose whether to enable Advanced Credential Pilfering, which uses advanced semantic reasoning to detect and extract credentials from various data sources. These credentials are then used to move laterally and escalate privileges during the attack.

> **Note**
>
> When this option is enabled, scanning extended domain users is turned off by default.

- **Step 3:** Choose a Data Risk Inference option to automatically identify business risks – such as exposed personal data, financial reports, or confidential strategy documents, and show the potential impact if they were compromised:
- **Disabled:** Turns off Data Risk Inference entirely.
- **Metadata Only:** Sends only file metadata (names, sizes, hashes). This is the fastest option, but may result in fewer findings.
- **Full Inference Data:** Sends small snippets of text from files that appear business-sensitive, enabling deeper and more accurate risk inference.
- **Step 4:** Run the test.
- **Step 5:** Once the test completes, you can view the results in any of these existing NodeZero workflows:
- **Impacts Tab:** New `Business Risk` column.
- **Weaknesses Tab:** Same column linking to weak points.
- **Data Tab:** Filter by Business Risk tags.
- **Sankey View:** Shows flow of sensitive data.
- **Attack Graph:** Includes a Business Risk node.
- **Reports:** Executive Summary highlights exposures.
- **APIs / MCP:** Export tags and GenAI Action Logs.

WHY IT MATTERS

The Problem

Security teams routinely uncover data during pentests-files, credentials, blobs-but struggle to determine **what's actually risky**. Traditional tools (e.g., DLP, DSPM):

- Generate noise with regex-based detection.
- Lack attacker context.
- Fail to link exposures to business consequences.

The Value of ADP

ADP closes this gap by:

- **Tagging files with business risk context** (e.g., Material Financials, Trade Secrets).
- **Prioritizing high-impact findings** using GenAI analysis.
- **Making every inference explainable** through Action Logs.

"We don't just show you **what** was found - we show you **why it matters**."

KEY DIFFERENTIATORS

Feature	Traditional Tools (DLP/DSPM)	ADP (NodeZero)
Detection Context	Regex / Keywords	Attacker-contextual (real operations)
Prioritization	Manual, noise-prone	GenAI-driven, context-aware
Explainability	Limited	Fully logged, transparent
Data Usage	Data stored/indexed	Runtime only, not used for training
Business Risk Mapping	Broad classification	Precise, executive-relevant tagging

---

HOW IT WORKS

ADP is **enabled by default** during internal pentests. No configuration or tuning needed.

Runtime Flow

1. **Advanced Credential Pilfering (ACP)** → Extracts credentials from reachable systems, files, shares.
2. **Data Risk Inference (GenAI)** → Evaluates content using LLM to determine business significance.
3. **Tagging** → Applies **Business Risk categories** like:
   - Operational Disruption
   - Critical System Shutdown
   - Revenue Interruption
   - Software Delivery Disruption
   - Supply Chain Breakdown
   - Leak of Sensitive Communications
   - Executive Fraud & Impersonation
   - Regulatory Breach Penalties
   - Theft of IP & R&D
   - Unauthorized Physical Access
   - Brand Impersonation & Takeover
4. **Context Scoring** → Adds `+2` score boost to business-risk findings to elevate them in dashboards and reports.

EXAMPLE USE CASES

- **Hidden VPN Credentials** in contracts → Tagged as Credentials + Legal Risk
- **Unreleased Earnings Forecasts** → Tagged as Material Financials
- **API Keys in Git Repo** → Tagged as Source Code Exposure
- **Defense Industrial Designs** → Tagged as Trade Secrets

BUSINESS RISK CATEGORIES

ADP currently supports **12 categories** of risk:

Auto-Tag	Business Risk	Real-World Example
Sensitive Communications	Reputational Damage	Sony Pictures 2014
Health Data (PHI)	Regulatory Penalties	Anthem 2015
Personal Data (SSNs, etc.)	Regulatory Penalties	Equifax 2017
Employee Data / HR	Regulatory + Reputational Risk	Sony Pictures HR
Source Code	Software Delivery Disruption	SolarWinds 2020
IP or R&D	Theft of Intellectual Property	PLA Indictment 2014
Manufacturing Data	Theft of Production Techniques	U.S. Steel / PLA
Alarm / Security Systems	Unauthorized Physical Access	Verkada 2021
Customer Data	Regulatory Penalties	Target 2013
Financial Data	Regulatory Penalties	Equifax 2017
Database Schemas / Config Files	Operational Disruption	CloudNordic 2023

GENAI ARCHITECTURE

Model Hosting

- All LLM calls are containerized within Horizon3's AWS infra.

Data Flow

Component	Data Sent to Model
Model Hosting	AWS Bedrock (Llama 4 Maverick)
Credential Extraction	Small text snippets
Repo Risk Detection	Metadata only or up to 10 file snippets
Database Schema Analysis	Table/Column names only (no data rows)

All data is filtered, minimal, and not used for training.

Prompt Management

- Custom prompts created by Horizon3.ai.
- Model outputs are **advisory only** – no autonomous actions.

How GenAI Is Integrated and Secured

Advanced Data Pilfering (ADP) leverage AWS Bedrock, a fully managed foundation model platform provided by Amazon. Specifically, NodeZero uses the Llama 4 Maverick model to perform semantic reasoning over structured metadata and sensitive content (depending on the feature).

The architecture is designed for security, explainability, and data isolation:

- A dedicated container runs inside the NodeZero Kubernetes cluster, sitting adjacent to the Core service.
- This container is responsible for communicating with AWS Bedrock and sending data for inference.
- The system analyzes short file content snippets that appear business-sensitive.

Importantly:

- No data is stored or used for training, by either Horizon3.ai or AWS.
- AWS Bedrock provides strong data isolation guarantees.
- The amount and type of data passed to the model is strictly controlled via configuration—ensuring minimal, targeted input for each use case.

For more on AWS foundation models, see: https://aws.amazon.com/what-is/foundation-models/

KEY CAPABILITIES

What ADP Is Designed For

- Highlighting **business risk** from attacker-discoverable data.
- Providing **explainable GenAI tagging** with full transparency.
- Prioritizing findings based on real attack paths—not theoretical access.
- Requiring **no manual tuning or rule writing**—just run the test.
- Making risk obvious across the UI, reports, and API.

What ADP Does Not Do

- ADP is **not a traditional DLP or DSPM tool**. It doesn't crawl all data.
- It doesn't perform full share scanning or file indexing.
- ADP does not replace compliance archiving or governance workflows.
- It focuses strictly on **attacker-accessible data** during live operations.

BUSINESS RISK TAXONOMY

ADP maps discovered data to a defined set of Business Risk categories using GenAI reasoning. These mappings help security teams communicate clearly with executives, auditors, and remediation teams.

Data Auto-Tag	Business Risk Name	Impact Rationale
Strategic Business Communications	Leak of Sensitive Communications	Leaked strategic communications create immediate reputational damage and stakeholder confidence erosion through exposure of internal debates, personnel issues, and sensitive business discussions. The 2014 Sony Pictures breach is a key example.
Health Data	Regulatory Breach Penalties	Breaches involving PHI trigger HIPAA, HITECH, and GDPR Article 9 violations. The 2015 Anthem breach (78.8M records) led to a $16M HIPAA settlement and regulatory oversight.
Personal Data	Regulatory Breach Penalties	Exposure of SSNs, addresses, or phone numbers triggers GDPR, CCPA, and state notification laws. The 2017 Equifax breach affected 147M people and led to a $575M settlement.
Employee Data / HR	Regulatory Breach Penalties	Exposure of employee data (SSNs, payroll, benefits) results in dual regulatory exposure (GDPR, HIPAA). The Anthem breach again is illustrative.
Source Code	Software Delivery Disruption	Compromised source code or CI/CD pipelines allow attackers to inject malicious code or disrupt delivery. Example: 2020 SolarWinds supply chain attack.
Intellectual Property	Theft of IP & R&D	Theft enables replication of proprietary innovations. 2014 Chinese PLA indictments showed IP theft from Westinghouse and U.S. Steel.
Manufacturing / Production	Theft of IP & R&D	Theft of proprietary formulas or processes enables global competition to replicate and undercut pricing. Again seen in the 2014 PLA indictments.
Employee Data / HR	Leak of Sensitive Communications	HR data (reviews, salaries, layoffs) leaks cause reputational and legal exposure. The Sony breach revealed exec salaries and performance reviews.
Physical Security / Surveillance / Alarm Systems	Unauthorized Physical Access	Compromised surveillance or alarm systems grant unauthorized access. 2021 Verkada breach gave access to 150,000 live camera feeds.
Customer Data	Regulatory Breach Penalties	Regulated customer data triggers GDPR, PCI-DSS, CCPA, leading to fines and audits. Example: 2013 Target breach ($18M in settlements).
Financial Data	Regulatory Breach Penalties	Financial data breaches invoke PCI-DSS, SOX, GLBA rules. The Equifax breach again illustrates regulatory consequences.
Digital Infrastructure	Operational Disruption	Attackers can misconfigure or corrupt infrastructure (configs, routes). The 2023 CloudNordic incident caused massive service outages due to ransomware and config issues.

FAQS

Q: Which operations support ADP? A: Internal only – phishing, insider, internal lateral movement.

Q: Is it included in all SKUs? A: No – ADP is part of the **Elite** SKU only.

Q: Is it on by default? A: Yes – both ACP and Data Risk Inference are default ON.

Q: Is customer data used for training? A: No – data is used only at runtime. Inferences are logged and explainable.

Q: Does this replace DLP/DSPM? A: No – ADP finds **attacker-reachable** risk, not compliance-level coverage.

3.3.12 High-Value Targeting

High-Value Targeting (HVT)

Release Date: October 10, 2025

Product: NodeZero (Internal Pentesting - Elite SKU)

OVERVIEW

High-Value Targeting (HVT) is a GenAI-powered capability in NodeZero that identifies the most critical assets, identities, and attack paths within your network—those that, if compromised, would pose the greatest business risk. These may include executive accounts, domain controllers, ERP systems, or exposed credentials tied to sensitive infrastructure.

HVT transforms raw findings into business risk narratives. By mapping exposures to categories like financial fraud, operational disruption, or reputational damage, it helps teams quickly understand why something matters - not just what was compromised.

This capability eliminates guesswork, allowing teams to: • Prioritize remediation around what's most impactful • Clearly communicate findings to leadership in plain business terms • Move beyond traditional vuln scoring or manual graph analysis tools

HVT is built to answer:

"What would an attacker go after first—and why does it matter?"

HOW TO USE

- **Step 1:** While creating an internal or phishing pentest, or an insider threat test, scroll to the High-Value Targeting section:

- **Step 2:** Choose whether to enable high-value targeting, and if so, what level of targeting.
- **Disabled:** Choose this option if you do not want to target high-value identities or assets for testing. If disabled, normal test will be run where all hosts and users are targeted in random order without context.
- **Comprehensive Targeting:** Choose this option if you want to target high-value identities and assets for testing during an internal NodeZero pentest,
- **High Value Targeting Only:** Choose this option if you only want to target high-value identities and assets for testing and exclude less valuable findings.
- **Step 3:** Run the test.
- **Step 4:** Once the test completes, you can view the results in any of these existing NodeZero workflows:
- Pentest summary page: You'll see a new card that shows the total number of business risks, and the Sankey chart will show a new section with all the impacts associated with the business risks. You'll also see a new bar chart below the Sankey chart that will show the hosts and credentials associated with business risks found.
- Impacts, Weaknesses, Credentials, Data, and Hosts sub tabs will show high-value target indicators that show why an asset matters.
- Attack Graphs will show high-value assets and identities tagged and visually prioritized.
- PDF Reports will include HVTs and explain business risks.

WHY IT MATTERS

The Problem

Security teams face an overwhelming volume of technical findings, but often lack clarity on which ones actually affect the business. Traditional tools answer "What can be exploited?" but not "What matters most to our organization?"

This leads to:

- Inefficient remediation prioritization
- Technical-to-business disconnect
- Missed opportunities for faster wins

The Value of HVT

HVT closes this gap by:

- **Pinpointing crown jewels**—executive accounts, production systems, critical paths.
- **Mapping technical findings to business risks** using GenAI.
- **Prioritizing the top 5% of assets** that carry 95% of business risk.
- **Providing explainable context** for remediation and executive reporting.
- **Requiring zero configuration or expertise**—built directly into NodeZero.

"We don't just show you what was found - we show you why it matters."

KEY BENEFITS

Capability	Benefit
GenAI-Powered Identification	Detects executive accounts, critical systems, DCs, etc.
Business Risk Mapping	Tags findings with labels like Payment Diversion, Ransomware
Context Scoring	Prioritizes findings in UI and reports
Explainable Action Logs	Shows why a target was flagged
Turnkey Integration	No setup required—runs by default in internal tests

---

HOW IT WORKS

1. Compromise & Data Collection

NodeZero collects real-time data on credentials, paths, hosts, and identities during a test.

2. GenAI Inference

GenAI engine identifies:

- Executive or admin identities
- Critical hosts (e.g., ERP, mail, domain controllers)
- Chains findings into attacker-style paths
- Tags each with mapped business risk

3. Action Logs

Every high-value tag includes a transparent explanation of:

- How it was discovered
- Why it matters
- What risk it maps to

4. UI Integration

Results appear across:

- **Attack Graph**
- **Impacts, Weaknesses, Credentials, Data, and Hosts tabs**
- **Reports & Executive Summaries**

5. Customer Controls

- Default: Runs in **Comprehensive** mode
- Optional: **HVT-Only** or disable per test

EXAMPLE USE CASES

Scenario	Business Risk Identified
Compromised CFO account	Payment Diversion / Financial Fraud
ERP Server compromise	Critical Business Process Shutdown
Domain Controller access	Enterprise-wide Authentication Failure
Compromised dev environment	Software Delivery Disruption
Vendor credential reuse	Supply Chain Breakdown

BUSINESS RISK CATEGORIES

Credential Tags (Examples):

- Executive Identity → Executive Fraud & Impersonation
- Identity Manager → Ultimate Privilege Escalation
- Developer → Software Supply Chain Risk
- Third-Party → Supply Chain Disruption

Host Tags (Examples):

- Domain Controller → Authentication Collapse
- Medical Systems → Patient Care Disruption
- Mail Server → Reputational Damage (Leaked Comms)
- Virtualization Infra → Multi-System Shutdown

GENAI ARCHITECTURE

Model Hosting

- All LLM calls are containerized within Horizon3's AWS infra.

Data Flow

Component	Description
Model Hosting	AWS Bedrock (Llama 4 Maverick)
Data Sent	Minimal metadata only (usernames, hosts, graph context)
Prompting	Curated prompts analyze attacker context and relationships
Output Behavior	Advisory only - never executes actions or modifies assets

All data is filtered, minimal, and not used for training.

Prompt Management

- Custom prompts created by Horizon3.ai.
- Model outputs are **advisory only** - no autonomous actions.

How GenAI Is Integrated and Secured

High-Value Targeting (HVT) leverage AWS Bedrock, a fully managed foundation model platform provided by Amazon. Specifically, NodeZero uses the Llama 4 Maverick model to perform semantic reasoning over structured metadata and sensitive content (depending on the feature).

The architecture is designed for security, explainability, and data isolation:

- A dedicated container runs inside the NodeZero Kubernetes cluster, sitting adjacent to the Core service.
- This container is responsible for communicating with AWS Bedrock and sending data for inference. • The data sent includes usernames, hostnames, and BloodHound-derived relationship metadata to prioritize targets based on attacker-accessible graph context.

Importantly:

- No data is stored or used for training, by either Horizon3.ai or AWS.
- AWS Bedrock provides strong data isolation guarantees.
- The amount and type of data passed to the model is strictly controlled via configuration—ensuring minimal, targeted input for each use case.

For more on AWS foundation models, see: https://aws.amazon.com/what-is/foundation-models/

REPORTING & EXPLAINABILITY

- Action Logs: Exportable via UI, PDF, or API
- Executive Summary: Business risks phrased in plain language
- Context scoring boosts HVT results in dashboards and reports
- Risk nodes added directly to the attack graph

BUSINESS RISK MAPPINGS

NodeZero uses GenAI reasoning to map compromised credentials and hosts to specific **Business Risk Categories**, providing a shared language for security teams and leadership.

Credential-Based Business Risks

Credential Type	Business Risk	Description
Identity Manager	Executive Fraud & Impersonation	Enables attackers to forge tokens, create backdoor accounts, and impersonate users with full legitimacy.
Developer	Software Delivery Disruption	Allows injection of malicious code into CI/CD pipelines, affecting downstream customers.
Third Party / Vendor	Supply Chain Breakdown	Compromising vendor credentials disrupts dependencies like IT services and logistics.
Business Executive	Executive Fraud & Impersonation	Enables fraudulent wire transfers using compromised executive accounts.
Domain Administrator	Operational Disruption	Grants control over Active Directory, allowing attackers to lock out all users.
IT Department	Operational Disruption	Disabling IT infrastructure halts business operations.

Host-Based Business Risks

Host Type	Business Risk	Description
Backup Infrastructure	Operational Disruption	Attackers can destroy recovery paths, amplifying downtime from ransomware or outages.
Medical Systems	Operational Disruption	Disrupts patient care and triggers regulatory consequences.
Domain Controller	Critical System Shutdown	Enterprise-wide authentication failure from DC compromise.
Operational Technology	Critical System Shutdown	Shuts down physical operations like energy and manufacturing.
Virtualization Infra	Critical System Shutdown	Disables all workloads dependent on hypervisors or management servers.
Financial Systems	Revenue Interruption	Interrupts payroll, billing, and collections.
Development Environments	Software Delivery Disruption	Corrupts builds or introduces backdoors into product code.
Operations & Logistics	Supply Chain Breakdown	Cascades disruptions across inventory, suppliers, and partners.
Mail Server	Leak of Sensitive Communications	Exposes executive, legal, and HR email threads with reputational impact.

FAQS

Q: Which operations support HVT? A: Internal pentests, phishing, and insider threat. Not external-only or cloud tests (yet).

Q: Is it enabled by default? A: Yes - runs automatically in Comprehensive mode.

Q: Does it affect runtime or stability? A: No - metadata only is processed. Models are isolated.

Q: Is my data used to train the model? A: No. Your data is used at runtime only and never leaves your AWS environment.

Q: Does it integrate with ticketing? A: Not yet - planned in future releases.

3.3.13 Endpoint Security Effectiveness

Endpoint Security Effectiveness

Endpoint Security Effectiveness gives you visibility into how your Endpoint Detection and Response (EDR) tools perform during a NodeZero Internal pentest.

OVERVIEW

This feature shows:

- Which hosts have an EDR deployed, and which do not.
- Which NodeZero actions were blocked by the EDR, and which were permitted.
- How different EDR vendors performed across your environment.

This data helps you confirm where EDR protections are working, and where gaps, misconfigurations, or silent failures exist

HOW IT WORKS

During a pentest, NodeZero attempts attacker behaviors on target hosts, and records whether the EDR on the host blocked or permitted each action. The results are then published on the pentest results page's **EDR** tab. NodeZero is able to detect over 40 EDR vendors including SentinelOne, Crowdstrike, Microsoft Defender. Please contact us if you'd like to see the full list of supported vendors.

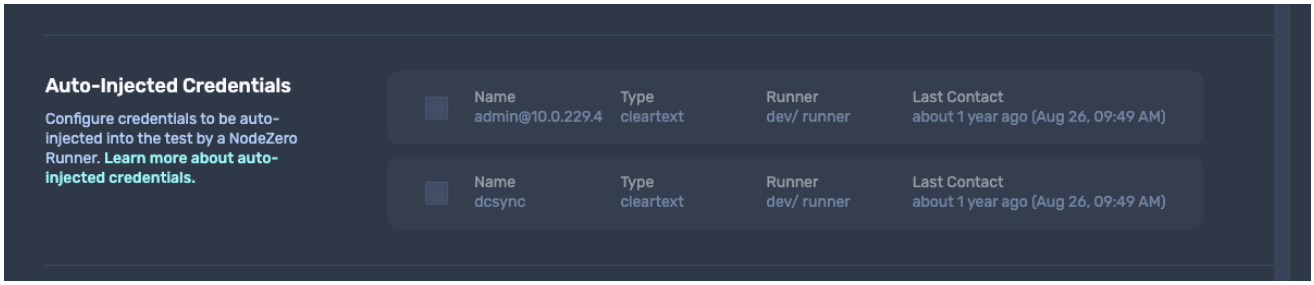
To ensure that NodeZero is able to access the host and analyze EDR performance, the following criteria have to be met:

- The host is running Windows (Linux is not currently supported).
- NodeZero has admin-level credentials to the host (this could be a local admin, a domain user with admin privileges, or a domain admin) with remote access to WMI (port 135). In some cases, NodeZero is still able to analyze the EDR on a host by exploiting a vulnerability on the host, or exploiting an admin-level credential.
- If a domain credential is being used, the domain controller needs to be in scope of the pentest. This can be modified via the template for the pentest.

Inject Credentials

To get the best results out of NodeZero's EDR testing, we strongly recommend you inject admin-level credentials as a part of the pentest definition. You can configure this via the pentest's template page, as shown below. This will give NodeZero the proper access it needs to run EDR tests on the hosts that are in scope.

When you inject credentials, ensure that within your test's **Attack Configuration**, you enable the **Credential Verification > Domain User** option.

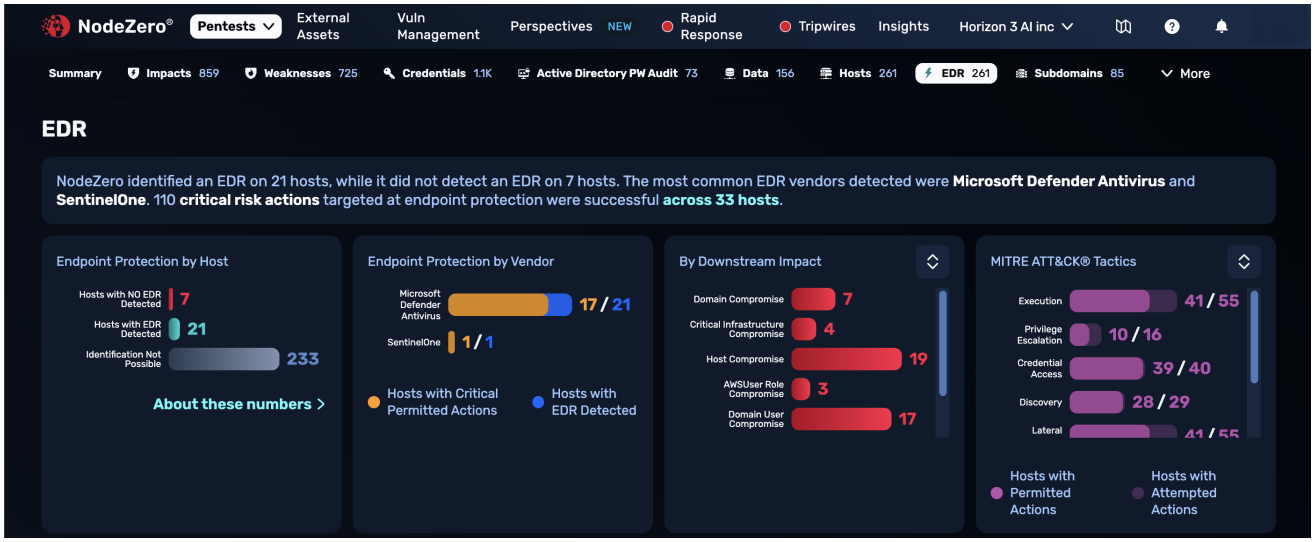


ANALYZING EDR RESULTS

Within the Internal Pentest results page, the **EDR** tab gives you access to an overview of EDR findings, along with analysis that you can use to tune your EDR configuration.

The **EDR** tab contains the following information:

- Coverage overview: Number of hosts on which EDR was detected, was not detected, or identification was not possible. Click **About these numbers** for breakdowns and details.
- EDR vendor breakdown: Which EDR vendors were discovered, with data on the percentage of actions blocked or permitted by one or more EDRs on each host.
- Downstream impacts: Potential consequences if actions were permitted (e.g., credential theft, domain compromise).
- MITRE mapping: Each permitted action is mapped to a MITRE ATT&CK tactic



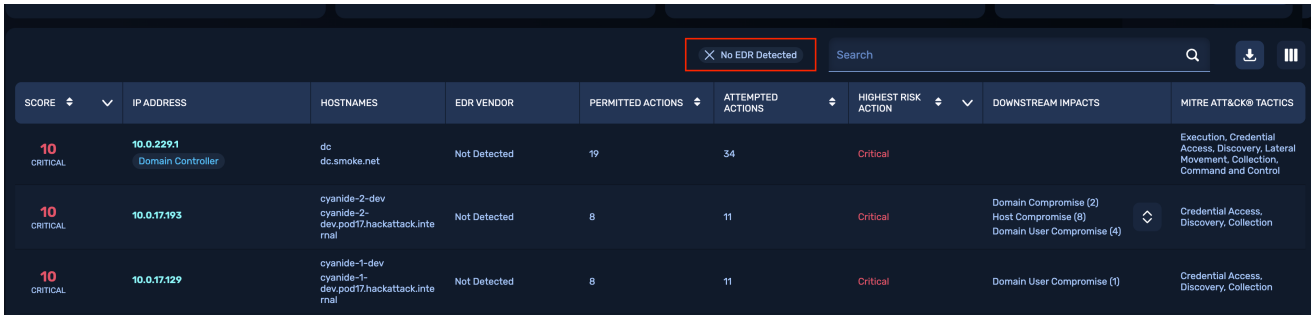
Hosts Table

The Hosts table contains the following information:

- Details for each host: hostname, IP, criticality, detected vendor, number of permitted events, and downstream impacts.
- Filters let you drill down by vendor or high-value asset

"No EDR" Filter

To identify hosts that need EDR coverage, navigate to the EDR summary dashboard shown above and select the **Hosts with No EDR Detected** bar. This will filter the results table as indicated in the example below.



Host-Specific Details

To get more information about which specific NodeZero actions were denied or permitted, you can click into a specific host and see details on specific actions.

For each host, you can see:

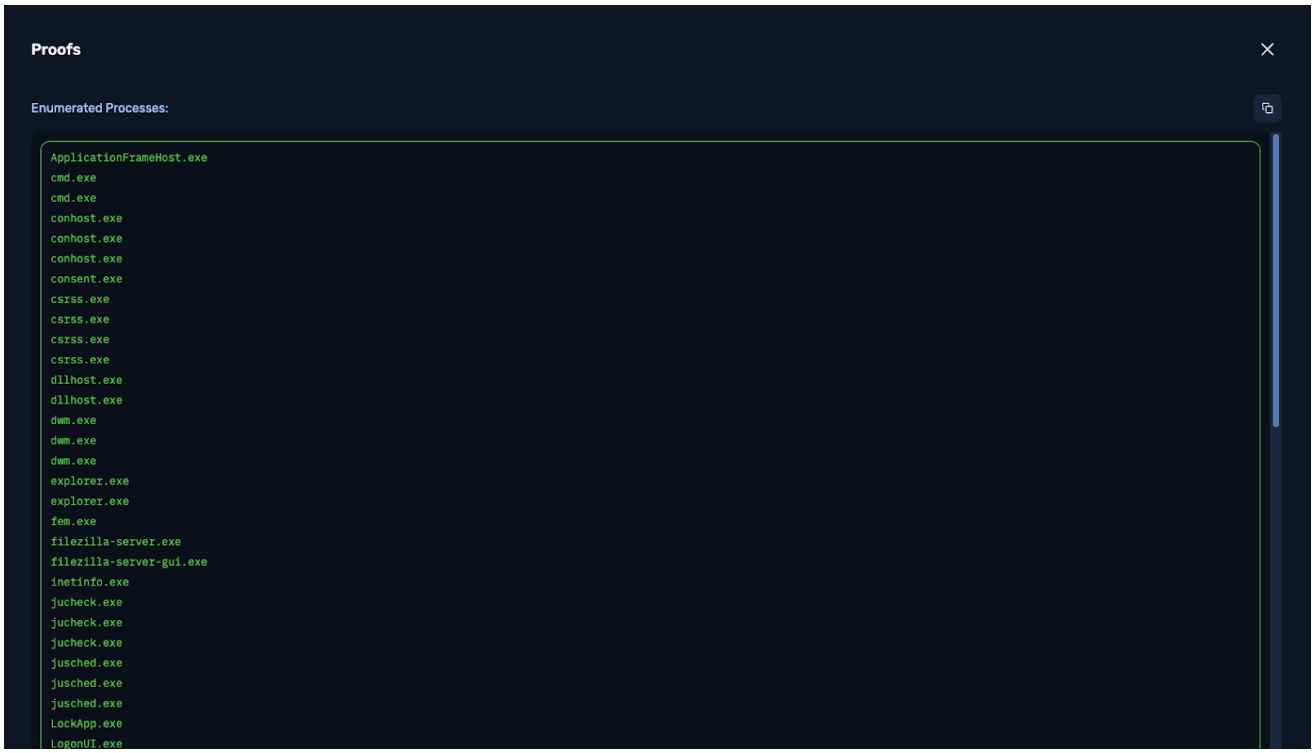
- Timeline of actions attempted by NodeZero.
- Outcome: blocked vs permitted.
- Severity rating: Critical, High, Medium, Low.
- Weakness mapping: Permitted actions linked to CVEs or known weaknesses.
- Proofs and timestamps: Command output and exact execution time for correlation with EDR logs

Events Table

The EDR Host events table contains rows for each event that occurred on this host. See the following screenshot for example:

Proof Details

Clicking on the Proof for a given event row shows the proof for that event:



Commands Details

Clicking on the Commands link for a given event shows a list of commands associated with this event:

Commands
Start Time
07/31/2025, 2:38 PM UTC
Run Time
less than a minute
Targets
10.0.229.11:135
End Time
07/31/2025, 2:39 PM UTC
Module
WMI Execute
MITRE ATT&CK® Tactics
Execution, Lateral Movement, Lateral Movement
Full Command
Copy
/opt/venv_cme/bin/python3 /opt/venv_cme/bin/wmiquery.py EXAMPLE.NET/administrator:U**********1 -file queries.txt

3.3.14 Campaigns

Campaigns and Perspectives

OVERVIEW

NodeZero campaigns and perspectives help your organization track how your security posture changes over time, from the attacker's viewpoint. A perspective is automatically created every time you create and save a pentesting template. Campaigns enable you to organize perspectives into groups that align with your organization's security initiatives, compliance goals, and team structures.

With these two capabilities together, you can easily see how your results are changing over time, and get recommendations for additional pentest coverage.

Example scenarios:

- Define campaigns for different network segments or security zones, to track the security posture of different segments over time.
- Use a campaign corresponding to different blue teams, to keep track of fixer progress.
- Use campaigns to keep an eye on compliance-related test coverage, to keep your compliance up-to-date.

> **Runner hints**
>
> Because these features are designed to assess repeated tests over time, their UI will nudge you to assign tests to Runners.

CAMPAIGN VIEW

You access campaigns (and perspectives) by selecting the **Perspectives** tab at the top of the NodeZero Portal. The main view for a campaign looks like this:

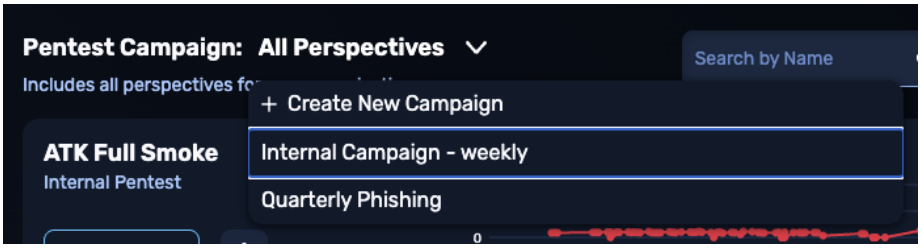


This Campaign page contains the following elements:

- Campaign name and description. The campaign name also serves as a menu, enabling you to select a different campaign.
- Controls for searching, filtering and sorting the campaigns.
- A button to switch between compact and full mode.
- A button for adding a perspective to this campaign. This can be an existing perspective, or a new perspective from Horizon3’s recommendations based on best practices.

Switching Campaigns

Clicking the campaign's name at the upper left opens a drop-down where you can select a different campaign, or create a new one.



All Perspectives – the Default Campaign

Your organization automatically includes a campaign called **All Perspectives** that includes every perspective that has been defined. This is an easy way to get an overview of all your pentest activity at once. This default campaign cannot be deleted, and these perspectives cannot be removed from the default campaign.

You can create new campaigns in addition to the default one. In these custom campaigns, you must specifically add the perspectives that belong to that campaign. You can remove these as desired.

Adding a New Campaign

Use the campaign drop-down and select **+ Create New Campaign**. In the resulting modal, specify the name for your new campaign, add a description to convey its purpose, then select all perspectives that should belong to this campaign. You can select all, some, or none of your existing perspectives. You can always change this later.

The maximum length for the campaign name is 128 characters. The maximum length for the description is 256 characters.

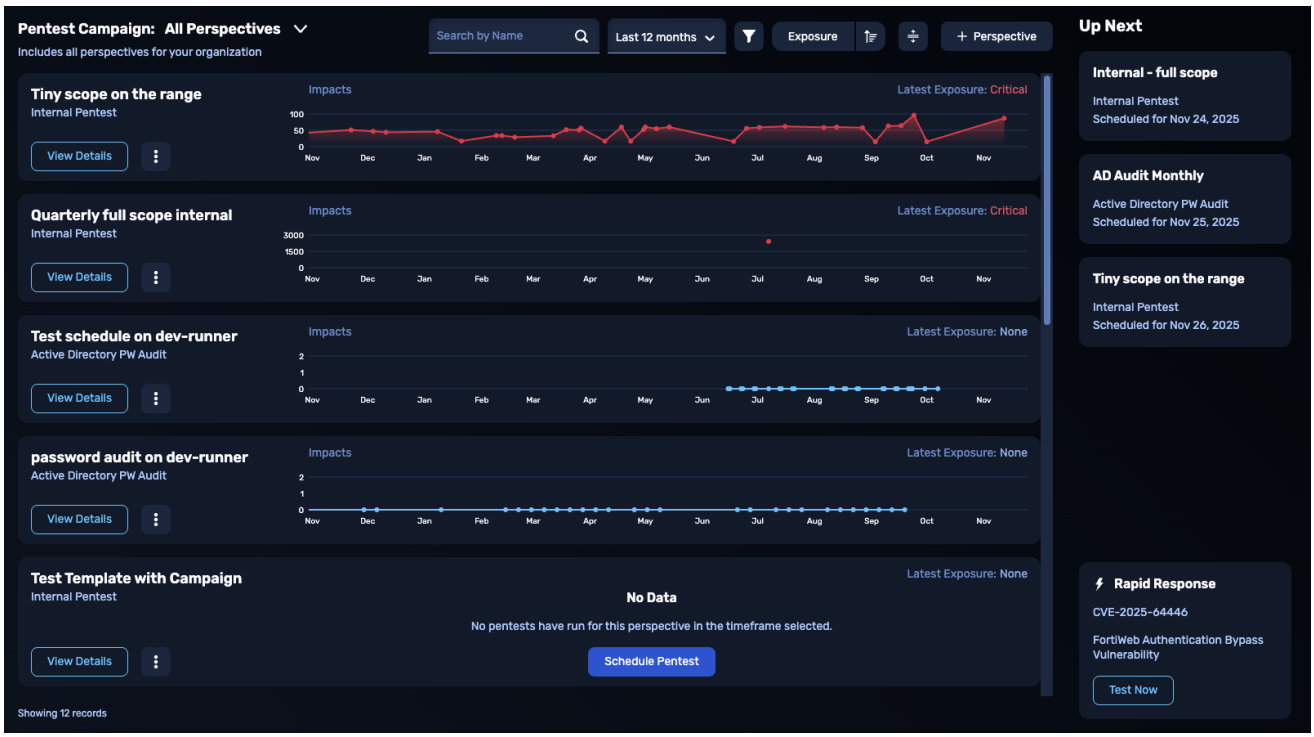
Click the **Create New Campaign** button to save your campaign's configuration.

Action Bar

At the top of the Campaign page are controls that filter or sort the perspectives list. You can search for perspectives by name, and filter them by retrospective time range, pentest type, exposure level, and other fields. A toggle enables you to change the sort order of the perspectives list.

Compact view

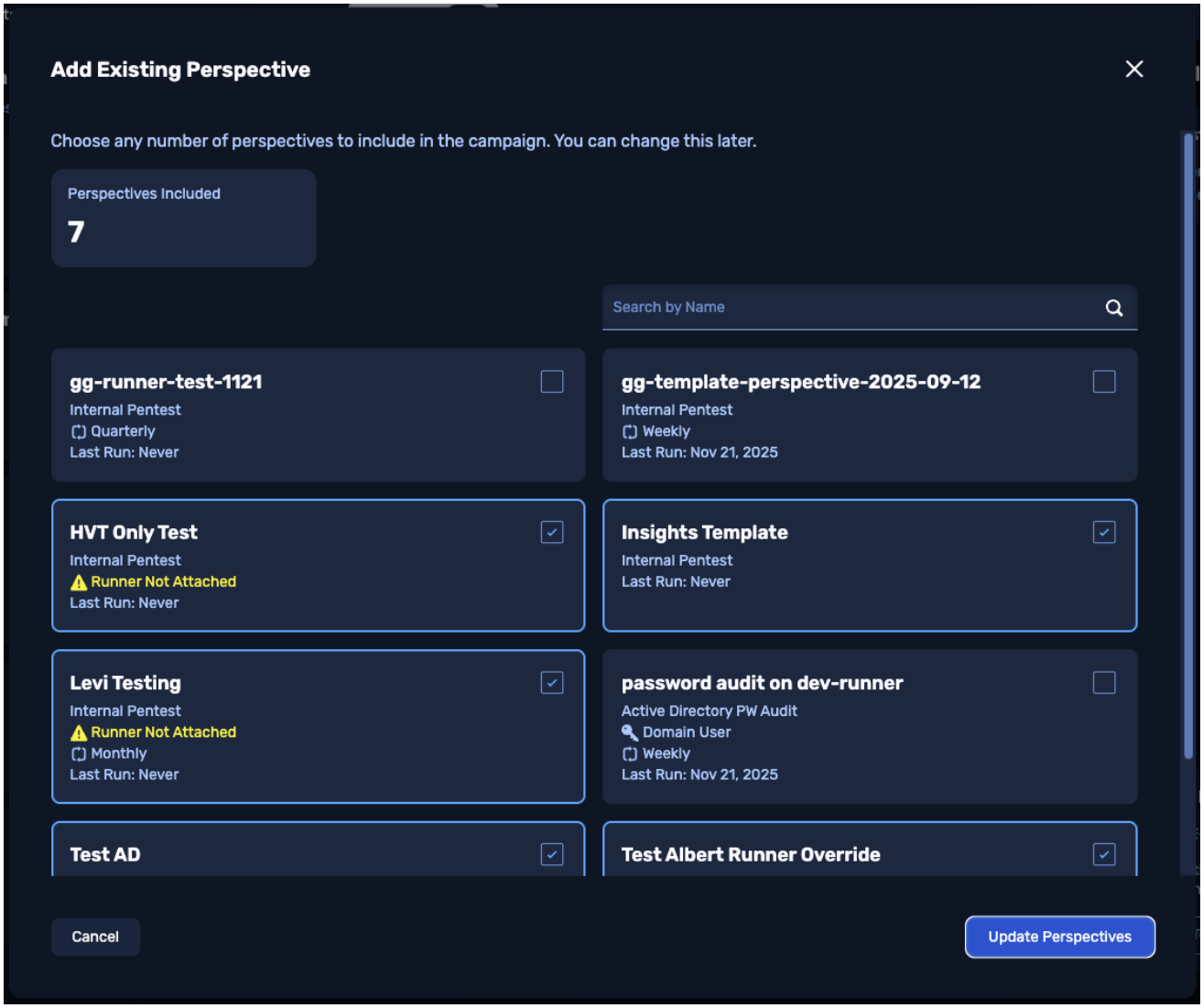
Clicking the Compact View button will condense the perspective graphs vertically, allowing you to fit more perspectives on your screen.



Add Perspective

Clicking `+ Perspective` enables you to add a perspective to this campaign. You can either add an existing perspective, or see a list of recommended perspectives based on best practices suggested by Horizon3 AI. Here, you can also remove perspectives from custom campaigns.

If you want to add existing perspectives, select the first option, and you will be able to select one or more perspectives to include in this campaign, similar to when you created the campaign:

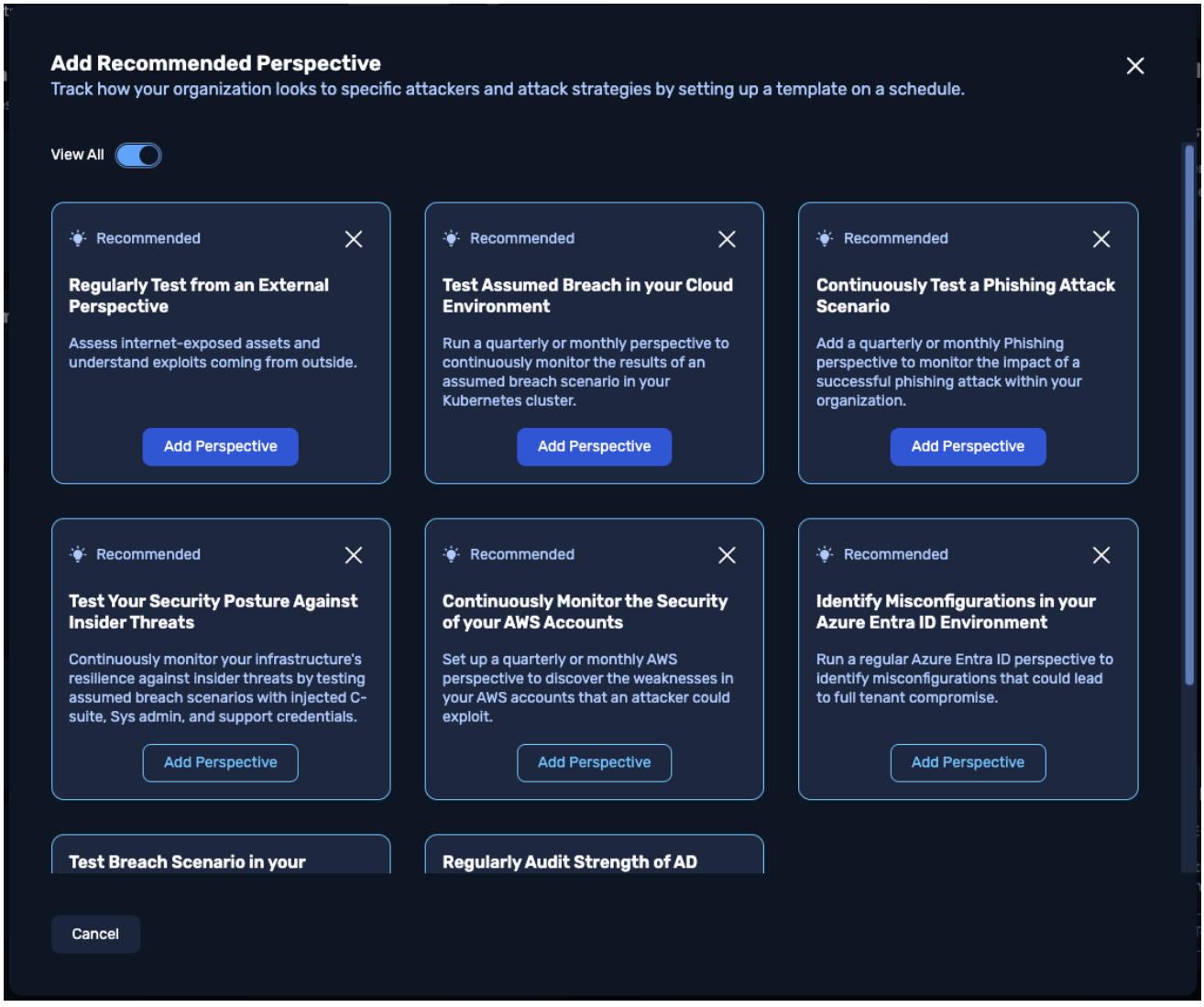


> **All Perspectives**
>
> If you're looking at the **All Perspectives** default campaign, there are no options to add or remove existing perspectives, because this campaign always contains all existing perspectives.

Recommended Perspectives

You can add a perspective to your campaign by following suggestions based on Horizon3.ai's best practices. This modal shows a prioritized list of the types of tests we recommend that you add for more coverage. Once you have the basics covered, we'll suggest more-specific tests to cover cloud deployments, phishing threats, and more.

By default, this modal shows only the top three recommendations. Click the link at the bottom to see more recommendations.



If you want to ignore a recommendation, click its `X` close box to dismiss it. Selecting a recommendation will take you to the template form to create a new pentest configuration for this perspective.

Perspective List

The main element on the Campaign page is the perspectives list. Here you'll find a row for each perspective in this campaign, showing its current exposure level, impacts count over time, and other details.

Each perspective displays a **View Details** button to open an expanded view (described below in Perspective View), as well as an Actions (⋮) menu to rename or remove this perspective. (Perspectives cannot be removed from the default **All Perspectives** campaign.)

Perspectives with no attached Runner will display a warning, plus a **Fix it** button that links you to the perspective's template configuration, where you can assign a Runner.

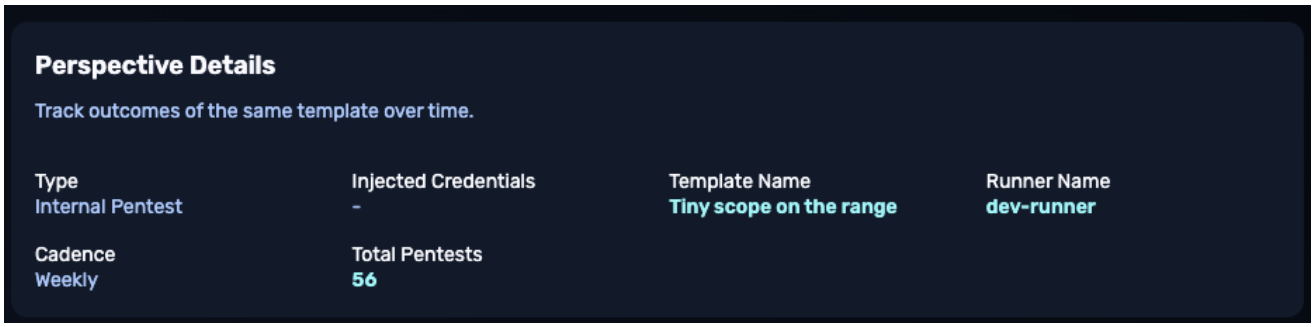


PERSPECTIVE VIEW

The perspective view gives you detailed information about the pentests, weaknesses, impacts, and threats associated with the pentests in this series. It tracks results longitudinally so that you can see what's getting better or worse.

Perspective Details

The Perspective Details card, atop the perspective view, shows you basic information like the test type, template and Runner names, pentest count, and schedule cadence. Click on linked values to get more information about each one. Click on the **Total Pentests** count to see a modal with details about all pentest runs for this perspective.



Exposure Score

This card shows the overall exposure score for this perspective, based on its most recent data (the last time the pentest ran). To learn more about how this is calculated, see Exposure Score Levels.

Threats and Weaknesses Diagram

This sankey diagram represents potential attack vectors, enriched by real-world Threat Actor Intelligence if this feature is enabled for your organization. This diagram shows a flow from threat actor group to exploited vulnerability, to impact, to business risk. Click on the different nodes in the diagram to learn more. This is based on the most recent pentest run in this perspective.



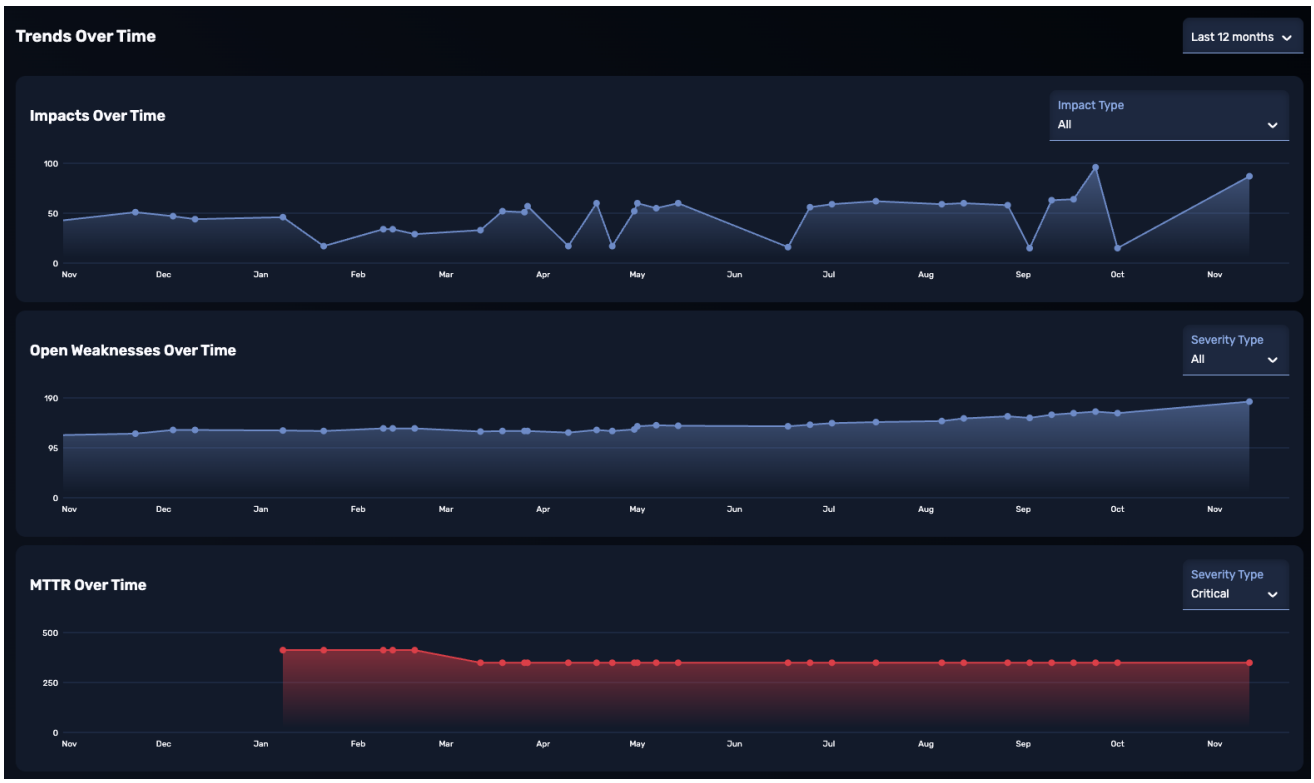
Weakness Charts

Next are four bar charts showing breakdowns of the weaknesses associated with this perspective, as of the most recent data:

- First, a breakdown of weakness by status, such as **Open** or **Regressed**.
- Second, a breakdown of user-supplied annotation status.
- Third is a breakdown by weakness severity.
- Last, we have a breakdown of weakness by impact type, enabling you to see which impact categories are most common in this perspective. This scrollable list will include categories like **Domain Compromise** and **Host Compromise**.

Trends over Time

At the bottom of the Perspective page are graphs showing activity in this perspective over time. There are graphs for impacts, weaknesses, and MTTR (mean time to remediation). These graphs provide drop-downs to filter to specific categories and severity levels.



AGGREGATING CAMPAIGNS INTO INSIGHTS

For a view of all your campaigns, NodeZero's Insights feature provides a combined analytics and management dashboard that shows your organization's overall exposure, remediation velocity, and risk trends over time. This facilitates streamlined, executive-level reporting of organization-wide initiatives and compliance.

3.4 Settings

3.4.1 Portal Settings

The Portal Settings section provides comprehensive control over your NodeZero account, enabling you to tailor security and operational settings to align with your organization's policies and workflows. From managing user roles and permissions to configuring integrations and fine-tuning system preferences, this section helps optimize both security and performance across your NodeZero environment.

User Management

Add, edit, remove users, and manage their permissions within NodeZero.

Single Sign On (SSO)

Enable and set up Single Sign On (SSO) for seamless user authentication.

Identification Provider (IdP) Guides

Guides for configuring an IdP for SSO integration to enhance security.

Email Notifications

Configure when and why NodeZero sends email notifications for critical events.

Setting a Proxy

Configure proxy if your organization's external traffic requires a proxy.

Client Management

Manage sub-clients, assign roles, and control access within parent accounts.

Co-Branding

Personalize reports for clients with your branding.

Integrations

Configure 3rd-party integrations with NodeZero.

Portal home

3.4.2 User Management

User Management

> **Admin feature**
>
> Only Org Admins can access User Management within the NodeZero Portal.

Each new user must be invited to a company account. If a user accesses the NodeZero Portal via social login or private SSO **without** an invite, this creates a personal, read-only account. By default, this personal account does not have access to any company accounts.

A user with Portal Org Admin access can send an invite to enable the user to switch from their personal account to their company account. The company account will become the default for all of that user's future logins.

NODEZERO USER ACCESS AND CONTROL

The NodeZero Portal supports the following login types:

Email Address and Password

This login type requires MFA (multi-factor authentication). NodeZero supports six-digit TOTP (time-based one-time password) codes from authenticator apps such as Okta Verify, Google Authenticator, 1Password, or Microsoft Authenticator.

NodeZero does not support MFA via SMS text messages or automated voice phone calls. Customers who normally rely on those methods should use single sign-on for NodeZero authentication.

Social Logins

The NodeZero Portal supports:

- Google
- Microsoft
- Entra ID (formerly called Azure Active Directory)
- Office365 (Entra ID/Azure AD is the backend)
- Free Microsoft accounts

> **Federal SSO requirements**
>
> NodeZero Federal does not support the email/password or social login methods. This product requires customer-managed single sign-on via your own identity provider, as described below.

SSO (Single Sign-On)

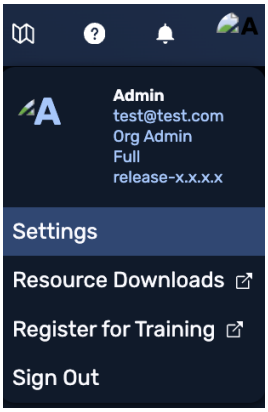
The NodeZero Portal supports OpenID Connect (OIDC)–compatible identity providers such as Okta, Keycloak, Ping, Google, or Azure AD/Entra ID.

In these flows, MFA is entirely handled by your identity provider. Any MFA method that your identity provider enforces (such as app-based TOTP or push, SMS/voice, hardware keys, or smart cards) will be honored before users reach NodeZero.

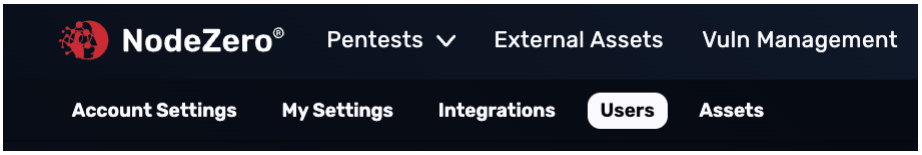
USER MANAGEMENT SETTINGS

Users with Portal Org Admin access are able to manage users and SSO provider settings, via the User Management menu.

To access the User Management menu, click the user profile button at the NodeZero Portal's top right, then select `Settings`.



Next, click `User Management`.



ADD USERS

Click the **+ User** (Add User) button to add users to your company account. All fields are required. Pay careful attention to the `Role` field to ensure that you're selecting the appropriate level of Portal access.



PORTAL ACCESS ROLES

The available NodeZero Portal access roles are:

- **Org Admin** – Full access to the company account.
- **User** – Access to run, schedule, and view pentests.
- **Readonly** – Access to only view previous pentest results.

Where Portal Org Admins already have a Basic username/password account, they will retain their username/password access to the NodeZero Portal even if **SSO Only** is enabled for the company account to which they belong. If you want to enforce Org Admins' login via SSO only, not username/password, please contact your Customer Success representative.

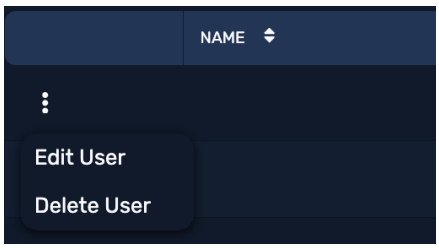
USER NOTIFICATIONS

When a user is added without SSO, they will receive two emails to the email address provided in the **Add User** form: a welcome email, and an email with temporary credentials to complete their account registration.

If **SSO Only** is enabled for a company account, the user will receive a single welcome email. This will direct them to follow whatever NodeZero Portal login processes their company has in place, via their identity provider.

EDIT OR DELETE USERS

You can edit or delete existing users by clicking the Actions menu (⋮) to the left of the user's name.



Next, select the appropriate action: `Edit User` or `Delete User`.

If you are using the Tripwires or Rapid Response feature, you can enable or disable users' access from the **Edit User** selection.

3.4.3 Single Sign-On (SSO)

Single Sign-On (SSO)

This page walks you through identifying and integrating supported identity providers with NodeZero.

SUPPORTED IDENTITY PROVIDERS

The NodeZero Portal supports OpenID Connect (OIDC)–compatible identity providers (IdP). This includes, but is not limited to: Okta, Keycloak, Ping, Google, and Azure AD. Any IdP that is OIDC-compliant should be compatible.

Once SSO is enabled, users have two ways to log in. They can log in at the Portal by selecting "Continue with Private SSO" or they can can log in via their IdP's application dashboard (Okta User Home, PingOne Application Portal, Microsoft My Applications, etc.).

General Requirements **US Portal Specifics** **EU Portal Specifics**

- OIDC-compatible identity provider
- Scopes: oidc (default), email, profile
- Grant type: authorization code
- Authorization type: POST

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse

ENABLE SINGLE SIGN-ON

The following sections provide the steps for setting up the SSO Provider for a Company Account. Note that some sections need to be followed by the NodeZero Portal's Org Admin, and others by the Identity Team Admin.

Build IdP App

To be completed by someone with permissions for Identity Team Admin

Check out our Identity Provider Guides section for some helpful direction on how to build and configure your app.

For IdPs not covered in our guides, we'll need the below outputs at a minimum. Once the app is built, provide the following information to the Portal Org Admin who will be creating the SSO Provider in the NodeZero Portal:

OIDC Discovery

When configuring the Issuer URL, do not include `/.well-known/openid-configuration`. This will be added automatically to your issuer for discovery.

Generic OpenID Provider **Okta** **Azure**

- Client ID
- Client Secret
- Issuer URL - Your IdP documentation should point you to the correct path. Typically `https://<yourIdPdomain>/<tenantID>`.

- Client ID
- Client Secret
- Issuer URL (`https://your_company_domain.okta.com`)

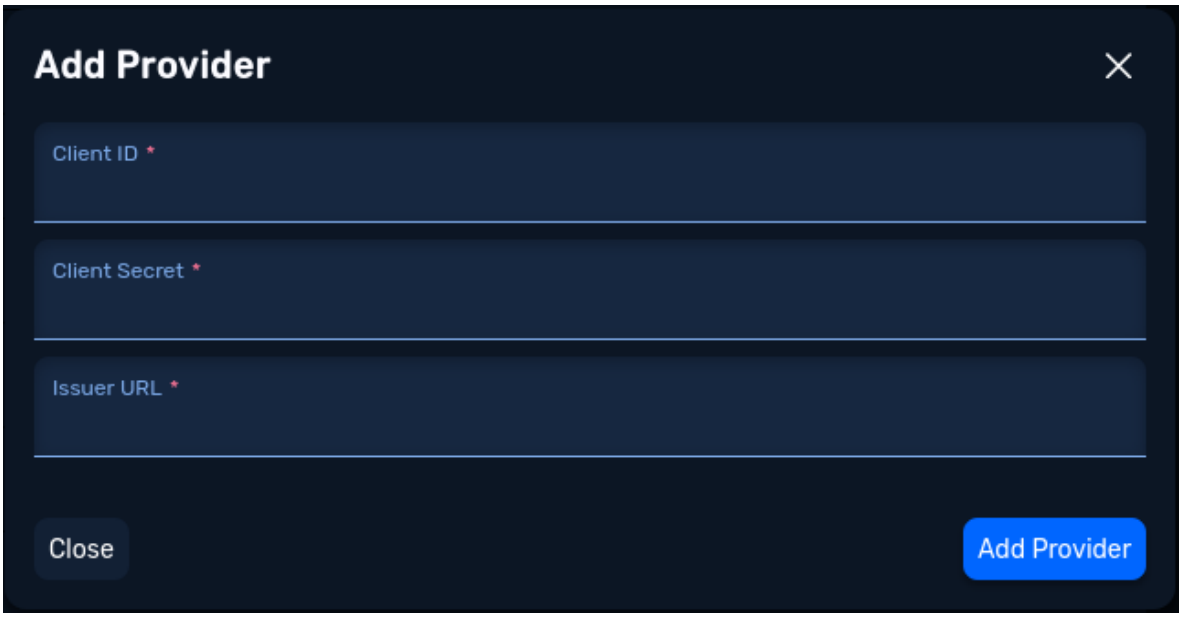
- Client ID
- Client Secret
- Issuer URL (`https://login.microsoftonline.com/<uuid>/v2.0`).
- Make sure to replace `<uuid>` with the UUID of the Azure app you created

Create SSO Provider in Portal

To be completed by someone with permissions for Portal Org Admin

1. Navigate to the Settings page. (Click the user profile button at the top right of the NodeZero Portal, then click `Settings`.)
2. On the resulting `Account Settings` page, scroll down to `Single Sign-On`.
3. Click the `+ Add Provider` button.

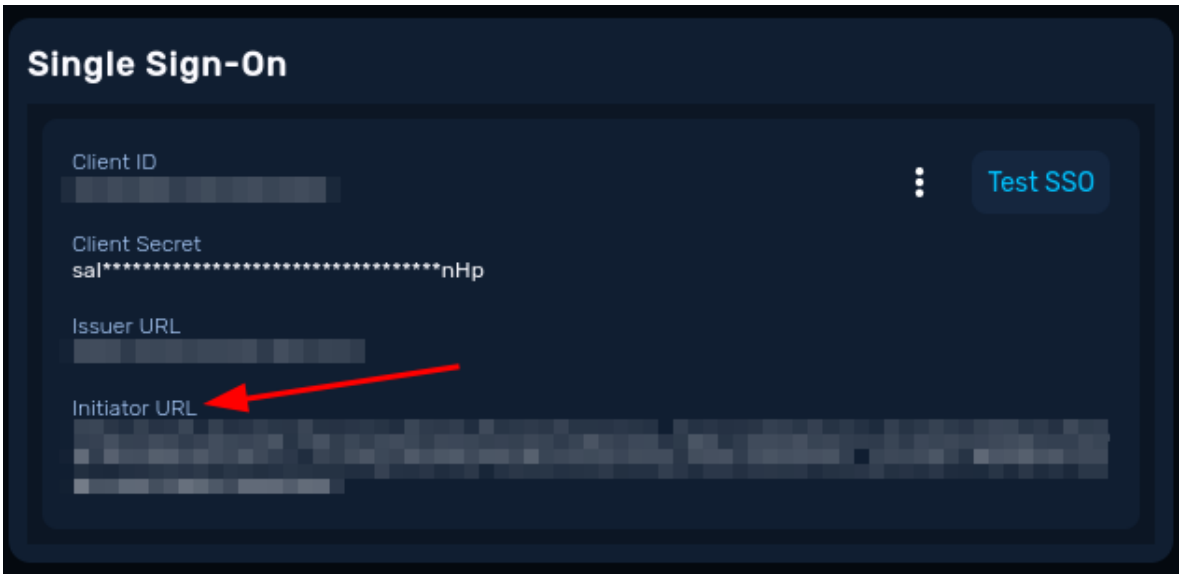
Populate the Add Provider form with the details provided by your Identity Team Admin from the Build IdP App section.



An **Initiator URL** value will be provided in the NodeZero Portal when the SSO Provider has finished updating. Test your SSO configuration and then provide this value to your Identity Team Admin.

Latency

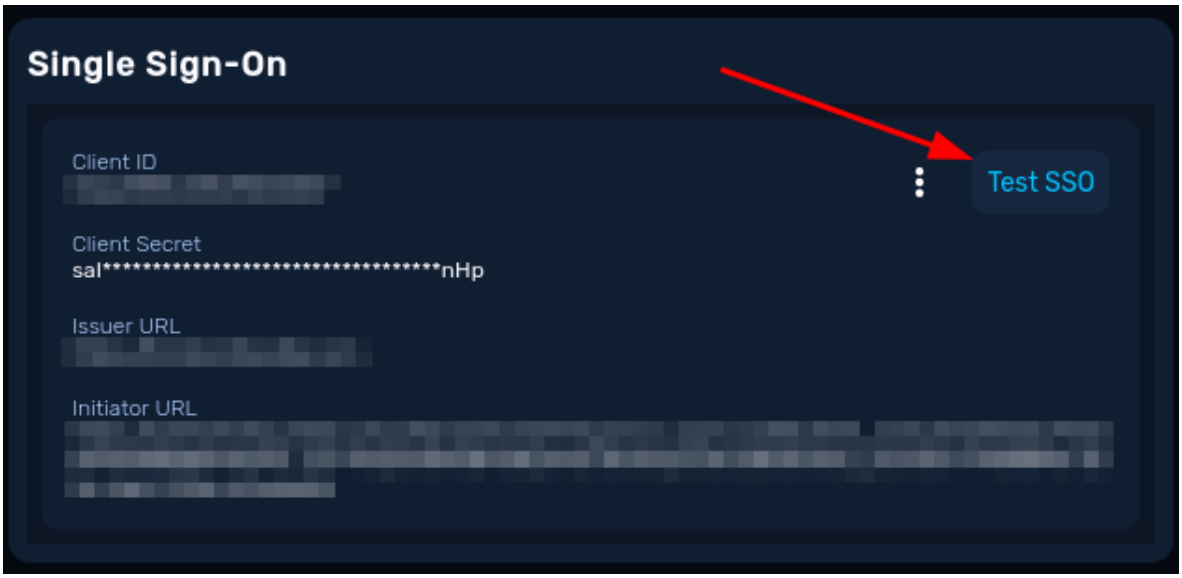
It can take 30–60 seconds for the SSO Provider to return the **Initiator URL**.



Test SSO Configuration

To be completed by someone with permissions for Portal Org Admin

The `Test SSO` button allows Org Admins to test the SSO configuration prior to sending the **Initiator URL** value to their Identity Team Admin.



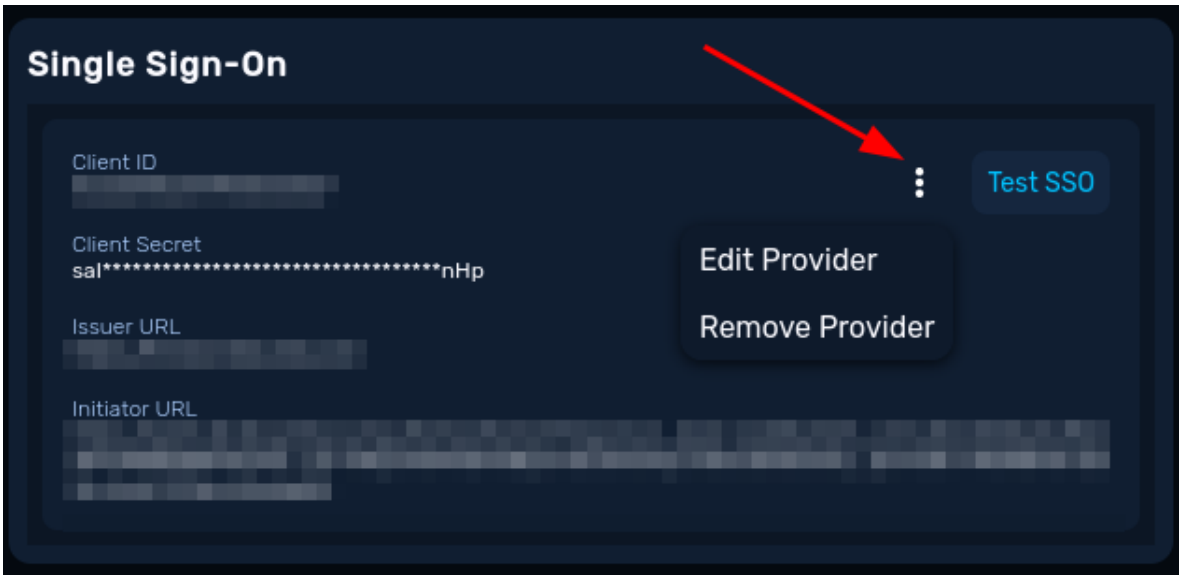
Configure Portal Access via IdP App

To be completed by someone with permissions for Identity Team Admin

Add the Initiator URL value as the **Initiate Login URI** to your IdP app to allow your users to access the NodeZero Portal via your IdP application.

EDIT/DELETE SSO PROVIDER

You can delete an SSO Provider by clicking the Actions (⋮) menu in the Single Sign-On section.



3.4.4 Identity Provider (IdP) Guides

Identity Provider (IdP) Guides

The NodeZero Portal supports OpenID Connect (OIDC) compatible identity providers (IdP). This includes but is not limited to: Okta, Keycloak, Ping, Google, and Azure AD. Any IdP that is OIDC compliant should be compatible.

Here are two guides on how to set up your IdP with

- Azure Guide
- Okta Guide

Azure

Warning

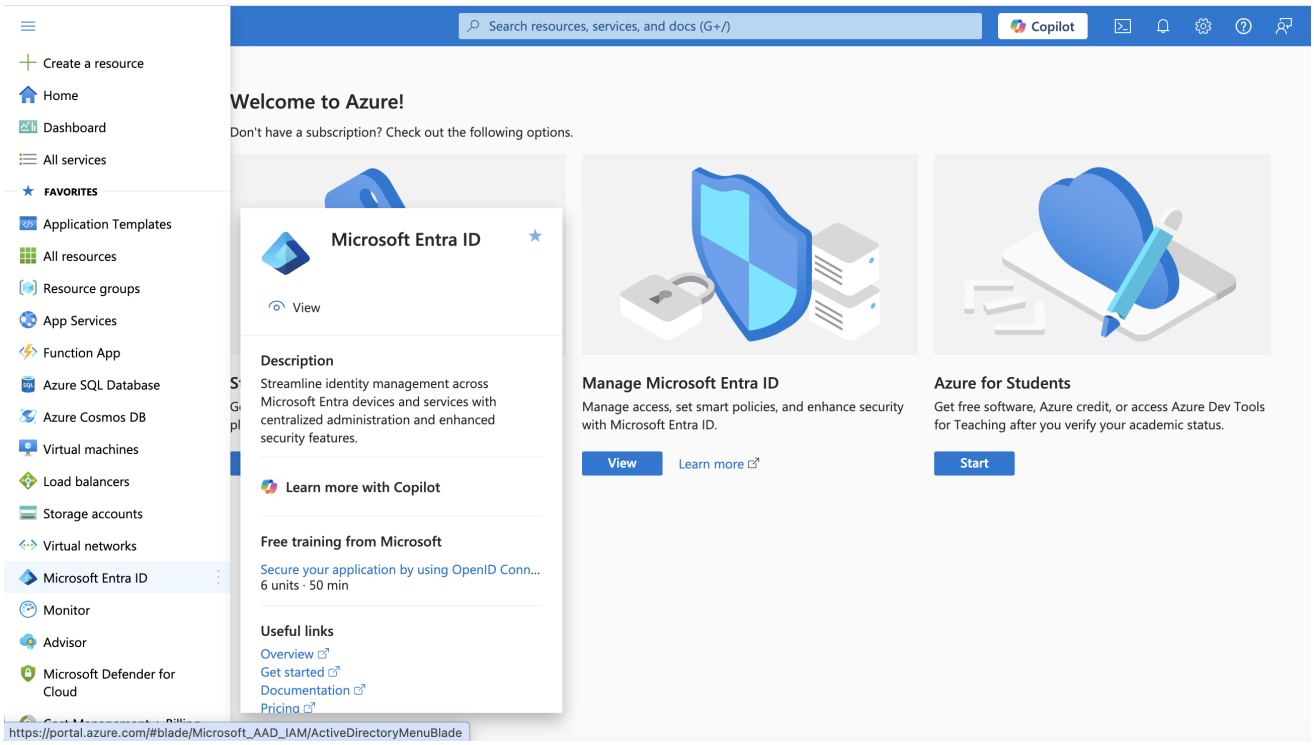
This guide should be used as a functional example only. Identity Admins should follow their company's policies and best practices when implementing Single Sign-On (SSO).

Similarly, because these guides are for services that Horizon3.ai does not control, screenshots and configuration options might be different then what you see here.

All sections of this page should be completed by someone with permissions for Identity Team Admin.

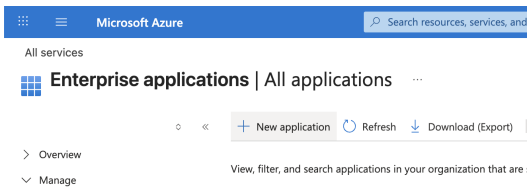
CREATE AZURE ENTERPRISE APPLICATION

1. Log into Azure Portal and browse to the `Microsoft Entra ID` service.



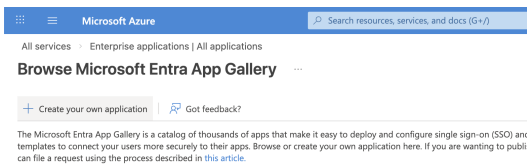
2. In the left-hand menu's `Manage` section, click `Enterprise applications`.

3. Then click `New Application`.



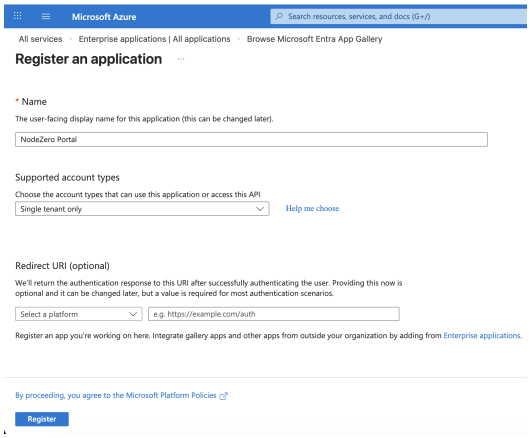
4. Then click `Create your own application`.

> **Required role**
>
> You will need to have one of the following Azure AD roles in order to create a new application: `Global Administrator` or `Application Administrator`.



5. Name your app `NodeZero Portal`.
6. Select `Register an application to integrate with Microsoft Entra ID (App you're developing)`.

7. Click `Create`.
8. On the `Register an application` page, you can choose to set a different user-facing name for the app, if desired.
9. Ensure that the `Supported account types` drop-down is set to `Single tenant only`.



10. Leave the `Redirect URI` section blank for now.
11. Click `Register`.

COPY CLIENT ID

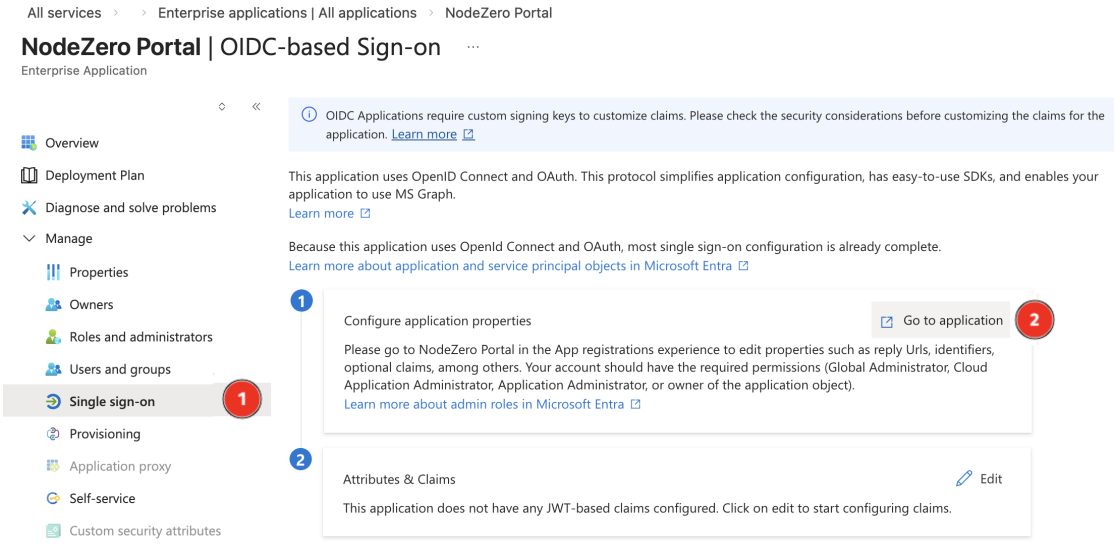
After registering the app, you'll be taken back to the `Browse Microsoft Entra App Gallery` page. Navigate back to the `Enterprise applications` page, find your newly created app, and click it.

1. Click on the `Overview` page.
2. Save the `Application ID`. This is the `Client ID` that you will need to provide to your Portal Org Admin later.

CONFIGURE SINGLE SIGN-ON

Under the `Manage` section of the left-hand menu:

1. Click `Single sign-on`.
2. Click `Go to application`.



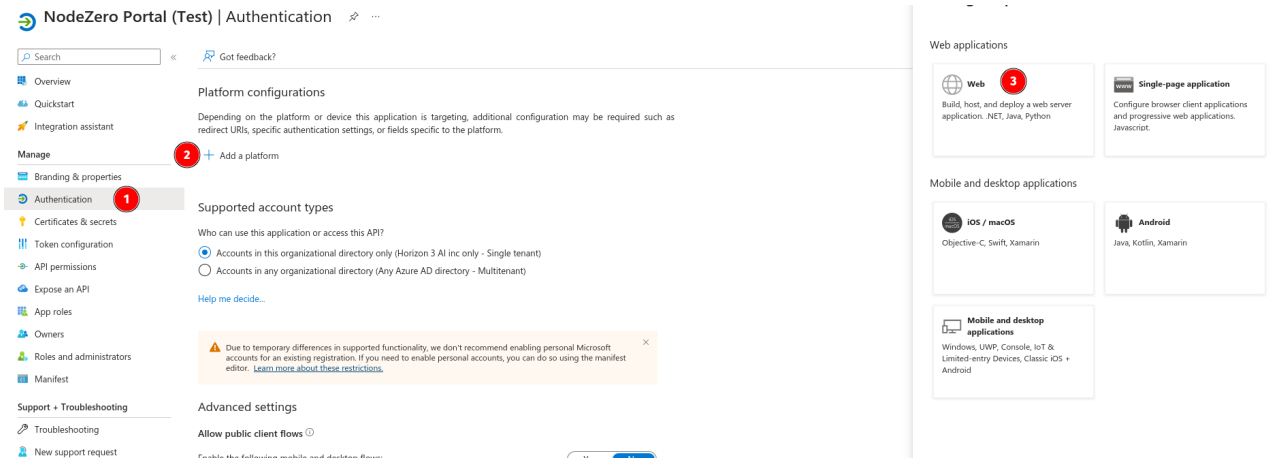
COPY ISSUER URL

On the new `Overview` page (step 1), click the `Endpoints` tab (step 2) and copy the `OpenID Connect metadata document` value (step 3). This is this `Issuer URL` that you will need to provide to your Portal Org Admin later.

CONFIGURE AUTHENTICATION

Under the `Manage` section,

1. Click `Authentication`.
2. Click `Add Redirect URI`.
3. In the `Web applications` section that opens to the right, click the `Web` button.



Use the information in the table below to fill out the `Redirect URIs` field. Be sure to select the correct tab, based on which regional Portal your users access.

US Portal **EU Portal**

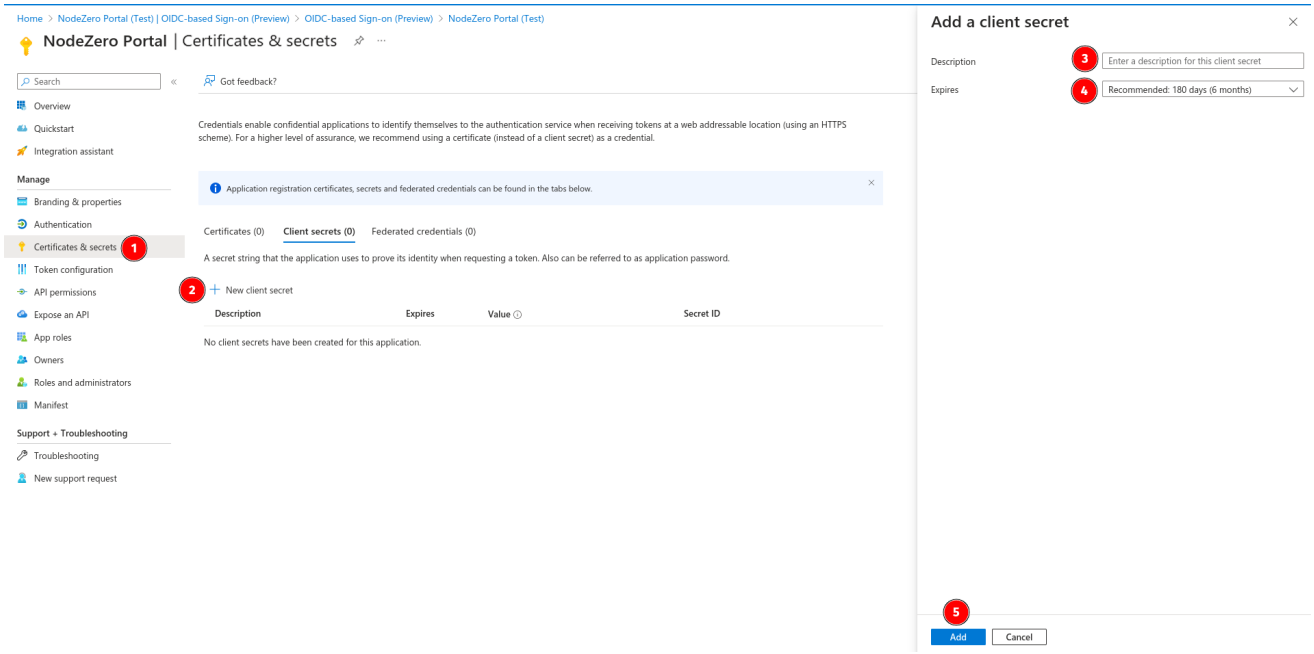
Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse

Field	Value
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse

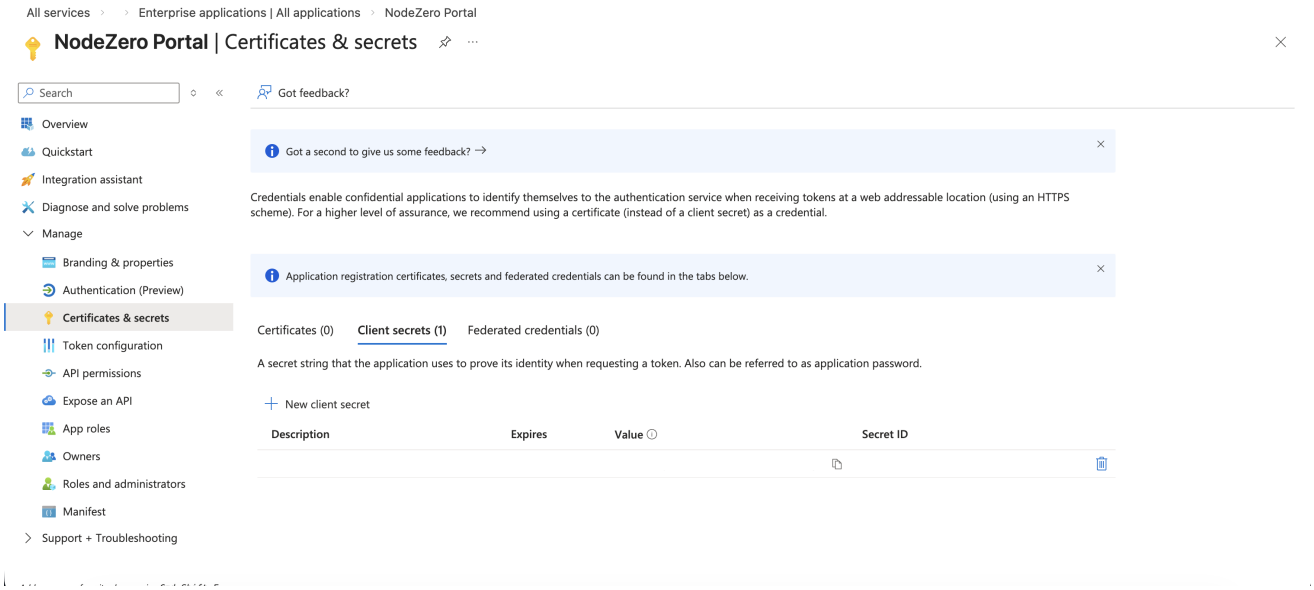
CREATE CLIENT SECRET

Under the Manage section,

1. Click `Certificates & secrets`.
2. Click `New client secret`.
3. Enter a description.
4. Set the `Expires` column to a value that aligns with your company's policies.
5. Click `Add`.



6. Copy the `Value`.

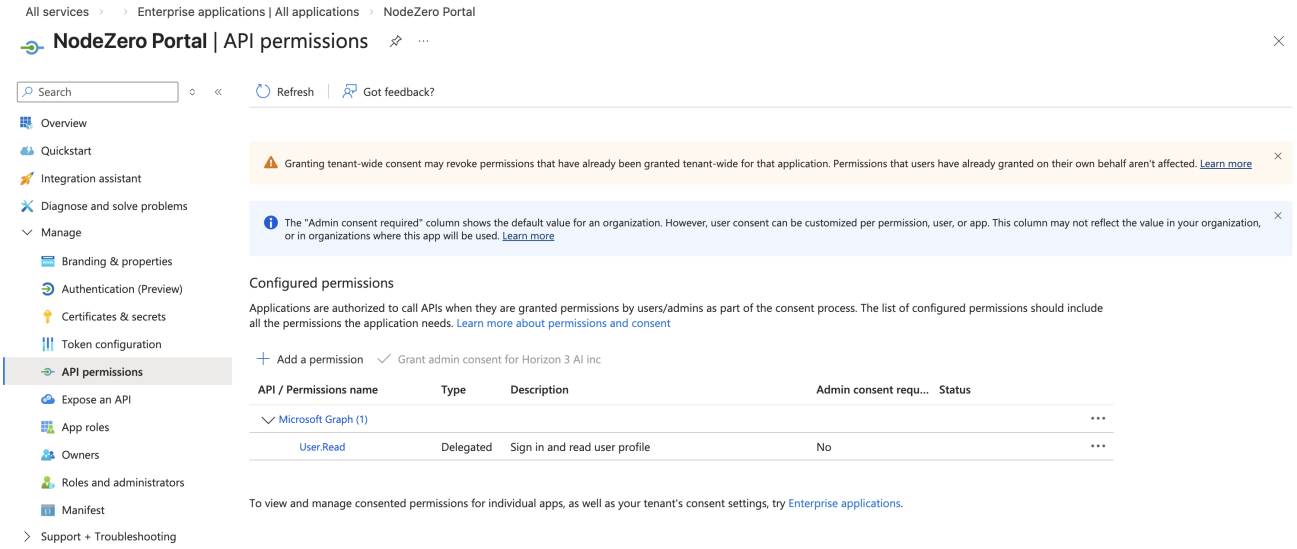


This is the `Value` that you will need to provide to your Portal Org Admin later.

CONFIGURE API PERMISSIONS

Under the `Manage` section,

1. Click `API permissions`.
2. Ensure that the Microsoft Graph `User.Read` permission is configured (it should be by default).



CONFIGURE APP ROLES

Under the `Manage` section,

1. Click `App roles`.
2. Click `Create app role`.
3. Fill out the form that opens on the right, using the information in the table below.

Field	Value
Display name	NodeZero Portal Users
Allowed Member Types	Users/Groups
Value	Read
Description	App role granting read to NodeZero Portal app.
Do you want to enable this app role?	✓

4. Click `Apply`.

PROVIDE INFORMATION TO ORG ADMIN

Provide the `Client ID`, `Client Secret`, and `Issuer URL` you copied in previous steps to your Portal Org Admin, so that they can configure the SSO Provider in the Portal. After the SSO Provider has been set up, your Portal Org Admin will need to provide you the `Initiator URL` so that you can complete the app configuration.

CONFIGURE BRANDING & PROPERTIES

> **Initiator URL**
>
> You will need the `Initiator URL` from your Portal Org Admin before you can proceed with this section.

Under the Manage section,

1. Click `Branding & properties`.
2. Fill out the form using the assets in the table below: the Name (step 2), Logo (step 3), and Home page URL (step 4).
3. Click `Save`.

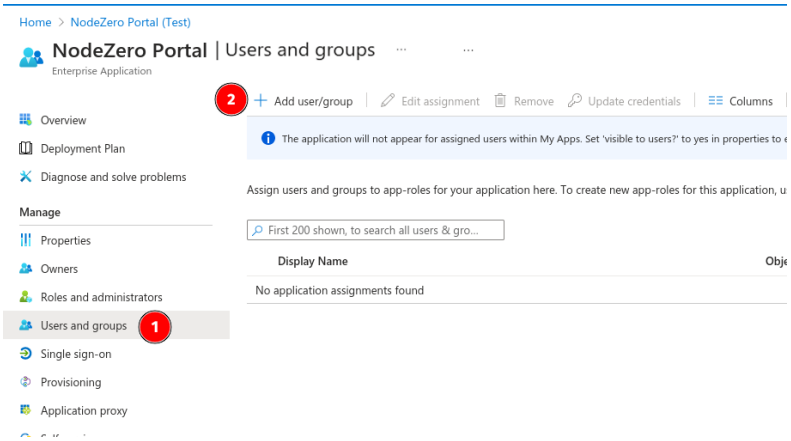
Field	Value
Name	NodeZero Portal
Logo	
Home page URL	Add Initiate login URI here

CONFIGURE USERS AND GROUPS

To grant users access to your new app, you will first need to navigate back to the `Enterprise applications` page we visited at the beginning of this guide.

Under the `Manage` section

1. Click `Users and groups`.
2. Click `Add user/group`.



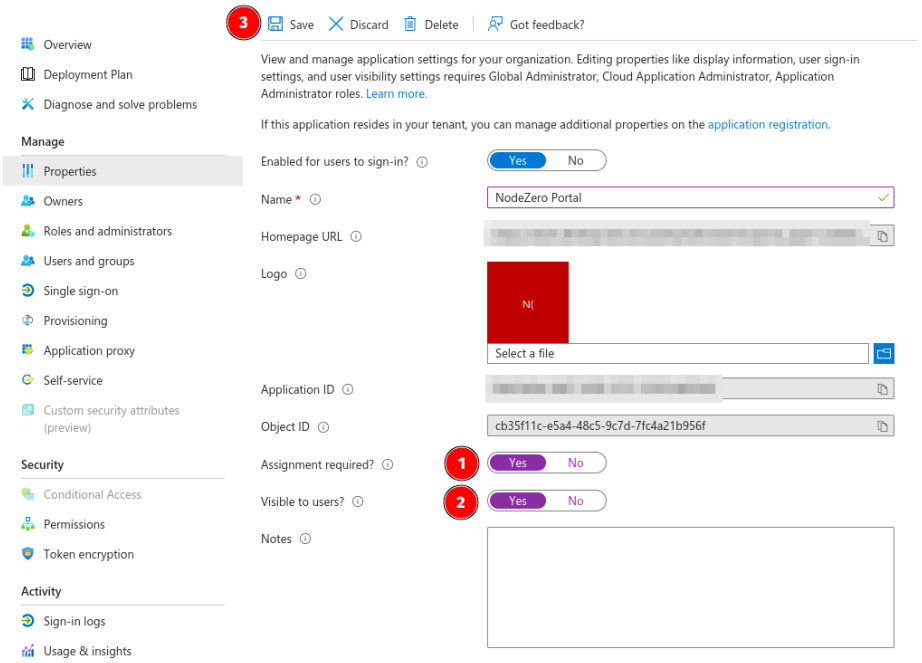
3. Select the appropriate users/groups.
4. Select the `NodeZero Portal Users` app role we created in a previous step.
5. Click `Assign`.

EDIT APP PROPERTIES

By default, the app will not appear for assigned users within `MyApps`. You will need to edit the visibility and assignment properties of the app.

Under the `Manage` section, click `Properties` and follow these steps:

1. Slide the toggle to `Yes` for both `Assignment required?` and
2. `Visible to users?`.
3. Click `Save`.



It can take 5–10 minutes for the app to appear in `MyApps`.

At this point, users can access by navigating to MyApps, logging in with their company credentials, and selecting the `NodeZero Portal` application tile.

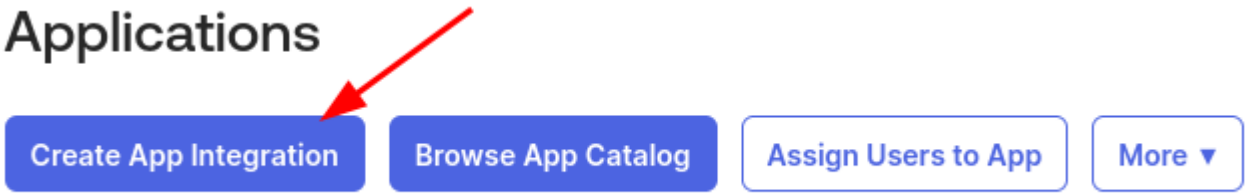
Okta

> **Warning**
>
> This guide should be used as a functional example only. Identity Admins should follow their Company's policies and best practices when implementing Single Sign-On (SSO).
>
> Similarly, because these guides are for services that Horizon3.ai does not control, screenshots and configuration options might be different then what you see here.

> All sections of this page should be completed by someone with permissions for Identity Team Admin

CREATE OKTA APP INTEGRATION

Log into your Okta Admin Console, browse to the Applications section, and click `Create App Integration`.



In the "Create a new app integration" form, select `OIDC - OpenID Connect` as the Sign-in method and then `Web Application` as the Application Type.

Create a new app integration
Sign-in method
Learn More
1
OIDC - OpenID Connect
Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.
SAML 2.0
XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.
SWA - Secure Web Authentication
Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.
API Services
Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.
Application type
What kind of application are you trying to integrate with Okta?
Specifying an application type customizes your experience and provides the best configuration, SDK, and sample recommendations.
2
Web Application
Server-side applications where authentication and tokens are handled on the server (for example, Go, Java, ASP.Net, Node.js, PHP)
Single-Page Application
Single-page web applications that run in the browser where the client receives tokens (for example, Javascript, Angular, React, Vue)
Native Application
Desktop or mobile applications that run natively on a device and redirect users to a non-HTTP callback (for example, iOS, Android, React Native)
Cancel
Next

Click `Next`. Use the information in the table below to fill out the `New Web App Integration` page. Be sure to select the correct tab, based on which regional Portal your users access.

US Portal **EU Portal**

Field	Value
App integration name	NodeZero Portal
Logo	H3 Logo
Grant Type	Authorization Code
Sign-in redirect URIs	https://portal.horizon3ai.com https://auth.horizon3ai.com/oauth2/idpresponse
Assignments	Limit access to selected groups OR Skip group assignment for now

Field	Value
App integration name	NodeZero Portal
Logo	H3 Logo
Grant Type	Authorization Code
Sign-in redirect URIs	https://portal.horizon3ai.eu https://auth.horizon3ai.eu/oauth2/idpresponse
Assignments	Limit access to selected groups, **or** skip group assignment for now

Note about Groups

If you set `Assignments` above to `Limit access to selected groups`, then in order to save your settings, you will need to add the Okta groups that should be assigned the app.

The NodeZero Portal does not pull groups or roles from the IdP, so they cannot be used for providing access via the Portal.

Click `Save`. Your page should refresh and you should be on the app console for your newly create app!

GATHER INFO

Under the `Client Credentials` section, copy the `Client ID`.

Under the `Client Secrets` section, copy the `Secret`.

Copy your `Issuer URL`. This is most often your okta domain. Include the `https://` prefix when submitting the `Issuer URL`

Provide the `Client ID`, `Client Secret`, and `Issuer URL` to your Portal Org Admin so they can configure the SSO Provider in Portal. After the SSO Provider has been set up, your Portal Org Admin will need to provide you the `Initiator URL` value so that you can complete the app configuration.

EDIT LOGIN SETTINGS

Initiate login URI

You will need the `Initiator URL` value from your Portal Org Admin before you can proceed with this section.

This is the value of the `Initiator URL` returned when the Org Admin creates the SSO Provider in the NodeZero Portal.

In the Okta Admin console, browse to your recently created `NodeZero Portal` app.

Scroll down to the `General Settings` section and click the `Edit` button. Under the `General Settings` > `Login` section, set the `Sign-out redirect URIs` to your Issuer URL. This is most often your okta domain.

Under the `General Settings` > `Login` section, click the `Login initiated by` drop-down, and select `Either Okta or App`. This should cause the `Application visibility` and `Login flow` fields to appear.

For the `Application visibility` field, check only the box for `Display application icon to users`.

For the `Login flow` field, select the `Redirect to app to initiate login (OIDC Compliant)` option.

For the `Initiate login URI` field, enter the value of the `Initiator URL` provided by your Portal Org Admin.

Click Save.

You can now assign this app to users. When you do, they should see a `NodeZero Portal` tile in their Okta App Dashboard and Okta Browser Plugin. Clicking it should log the user into the Portal.

3.4.5 Email Notifications

Notifications

NodeZero will send email notifications for the following events that occur during the operation lifecycle.

State types:

ℹ️ - informational ⚠️ - action required

Event / State	Type	Description	Sent To
Pentest started	ℹ️	NodeZero has launched and successfully connected back to the one-time-use ephemeral architecture to start the pentest	The user who created the pentest
Pentest completed	ℹ️	The pentest completed and results are available in the Portal	The user who created the pentest
Pentest paused	⚠️	The pentest paused due to lost connectivity with NodeZero	The user who created the pentest
Pentest resumed	ℹ️	The pentest resumed after being paused	The user who created the pentest
Pentest expired	ℹ️	The pentest never launched and was auto-canceled after 12 hours	The user who created the pentest
Paused pentest canceled	ℹ️	The pentest was stuck in a paused state for over a week and was auto-canceled	The user who created the pentest
Pentest waiting for NodeZero	⚠️	The pentest was created, but has not yet launched, because NodeZero has not yet connected back the H3 cloud	The user who created the pentest
Pentest ready to start	⚠️	The pentest launched in a paused state, typically for whitelisting NodeZero's IP, and is waiting to be started by the user	The user who created the pentest
Scheduled action initiated	ℹ️	The scheduled action was successfully initiated at its scheduled time, e.g. creating a scheduled pentest via the h3-cli or Portal	The user who created the scheduled action
Scheduled action failed	⚠️	The scheduled action failed to run	The user who created the scheduled action
NodeZero Runner hit an error	⚠️	The NodeZero Runner failed to launch NodeZero	The user who created the Runner, along with the user who created the pentest that failed to launch

3.4.6 Set a Proxy

Using a Proxy with NodeZero

If your organization uses a proxy for external traffic, you will need to set up your host to make use of that proxy in order to reach the NodeZero infrastructure. With these changes in place, all traffic between the NodeZero container and cloud infrastructure will use the proxy, and internal traffic will not.

Recommended If using the NodeZero OVA, the built-in `n0` utility can assist with setting up the proxy.

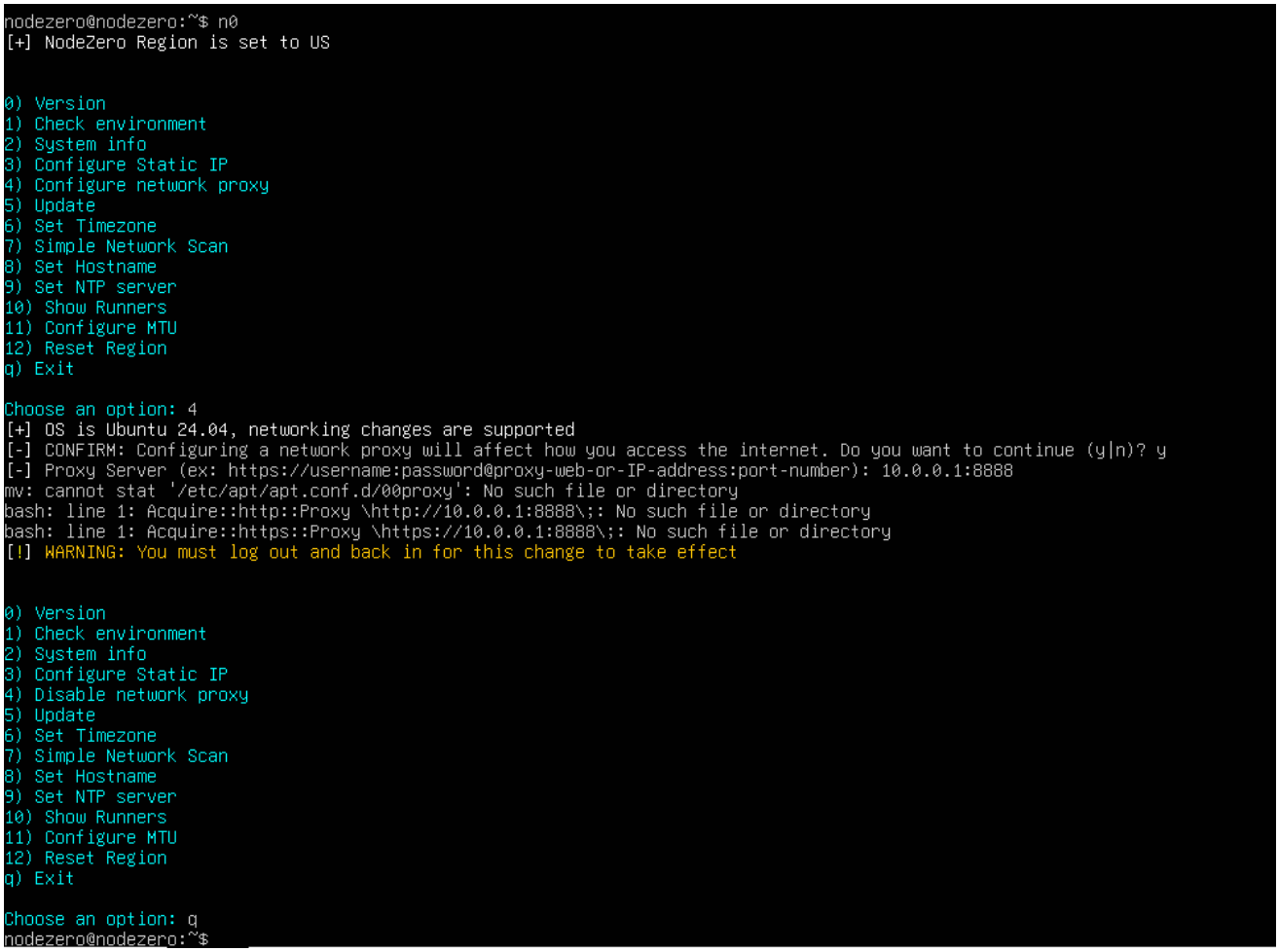
If running your own NodeZero host, steps are outlined below for Docker on Ubuntu and Podman on RHEL. Other OS configurations might vary.

CONFIGURE A NETWORK PROXY ON THE NODEZERO OVA

> **SOCKS proxies**
>
> NodeZero does not support SOCKS-type proxies. Please use only HTTP/HTTPS proxies.

To use NodeZero with a proxy download and configure the NodeZero Host VM (OVA) located at Downloads.

The NodeZero host VM (OVA) comes with a `n0` utility script that helps with management of the system. To set up a proxy with `n0`, start by entering `n0` into the command prompt and entering option `4`. In the example shown, the proxy server is set up on `10.0.0.1:8888` with no authentication.



For changes to take effect

After setting the proxy, the user must log out and back in to get the changes to take effect.

The NodeZero OVA host is now set up to utilize a proxy for your NodeZero pentests.

MANUALLY CONFIGURE A NETWORK PROXY ON NON-OVA HOST

Proxy format

Ensure that the proxy is formatted as follows: `https://username:password@proxy-web-or-IP-address:port-number`

Docker (Ubuntu) **Podman (RHEL)**

This guide assumes Docker is already installed. Four files must be updated to support a proxy with NodeZero on Docker:

```
/etc/environment
/etc/docker/daemon.json
/etc/systemd/system/docker.service.d/http-proxy.conf
~/.docker/config.json
```

Configure `/etc/environment`

This sets the `env` variables for your shell environment. Make the proxy available in the shell interface by adding the following to `/etc/environment`

```
HTTP_PROXY="your proxy settings"
HTTPS_PROXY="your proxy settings"
NO_PROXY="localhost,127.0.0.1,::1,172.17.0.0/16"
http_proxy="your proxy settings"
https_proxy="your proxy settings"
no_proxy="localhost,127.0.0.1,::1,172.17.0.0/16"
```

Configure `/etc/docker/daemon.json`

Create and fill out `/etc/docker/daemon.json`

```
sudo vim /etc/docker/daemon.json
```

The contents of `/etc/docker/daemon.json` should look like the following:

```
{
    "proxies": {
        "http-proxy": "your proxy settings",
        "https-proxy": "your proxy settings",
        "no-proxy": "localhost,127.0.0.1,::1,172.16.0.0/16,10.0.0.0/8,192.168.0.0/16"
    }
}
```

Configure `/etc/systemd/system/docker.service.d/http-proxy.conf`

Create and fill out `/etc/systemd/system/docker.service.d/http-proxy.conf`

```
sudo mkdir -p /etc/systemd/system/docker.service.d
sudo vim /etc/systemd/system/docker.service.d/http-proxy.conf
```

The contents of `http-proxy.conf` should look like the following.

```
[Service]

Environment="HTTP_PROXY=your proxy settings"
Environment="HTTPS_PROXY=your proxy settings"
Environment="NO_PROXY=localhost,127.0.0.1,::1,172.17.0.0/16"
Environment="http_proxy=your proxy settings"
Environment="https_proxy=your proxy settings"
Environment="no_proxy=localhost,127.0.0.1,::1,172.17.0.0/16"
```

Reload and restart the Docker service.

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Configure `~/.docker/config.json`

Set up the `config.json` in the local `.docker` directory.

Running as root?

If running NodeZero from `root`, change the directory to `/root/.docker/config.json`.

The following command will create the necessary directory and file if it is not present, and will make the current `$USER` the owner.

```
mkdir -p ~/.docker && touch ~/.docker/config.json
sudo chown -R $USER:$USER ~/.docker
```


Fill out `~/.docker/config.json` by adding the `proxies` section.

The NodeZero host is now configured to use a proxy with NodeZero.

3.4.7 Client Management

Client Management

Client Management in NodeZero enables organizations to manage multiple clients or business units within a single NodeZero deployment. This feature is particularly useful for Managed Security Service Providers (MSSPs) or large enterprises with multiple distinct security environments. It enables the parent account's owner to limit and monitor usage, and to adjust limits as needed.

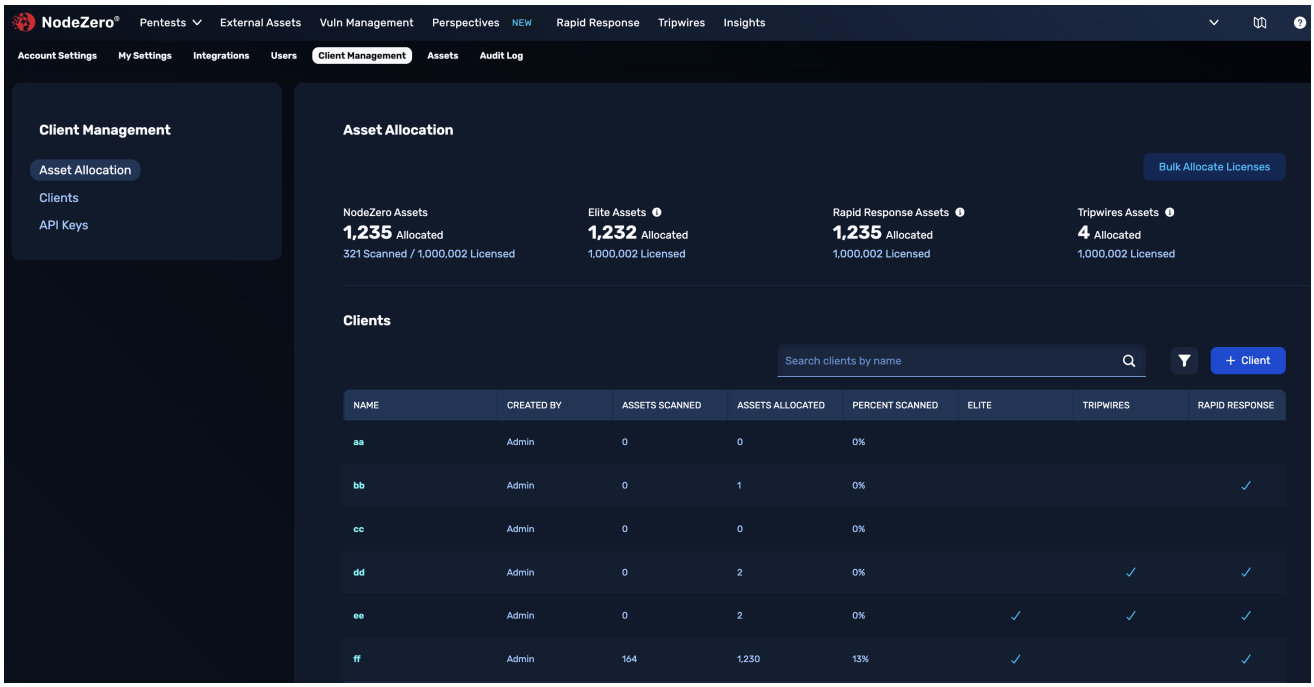
REQUIREMENTS

- The `Org Admin` role is required to access the **Settings > Client Management** tab and page.
- Your account must have multi-tenancy enabled. (All MSP/MSSP access levels include this feature.) To add multi-tenancy, please reach out to our sales team for assistance.

The `Org Admin` role can create clients within the parent organization, switch between client accounts, and manage all these accounts. This enables the Admin to allocate assets from the parent's pool across clients.

ACCESSING THE CLIENT MANAGEMENT PAGE

1. Verify that you have `Org Admin` permissions.
2. Open the user profile menu at the NodeZero Portal's top right, and select **Settings**.
3. From the resulting Settings submenu, select **Client Management**. This opens the page shown below.



ADDING NEW CLIENTS

1. On the **Client Management** page's right side, click the **+ Client** button.
2. Provide the company with a `Name` and a `Short Name`.
3. You can allocate assets and additional package features to this client now or later.
4. Click **Add**.

TRIPWIRES, RAPID RESPONSE, AND ELITE FEATURES

For each client, beyond setting the number of assets that can be scanned, you can also assign access to NodeZero **Tripwires**, **Rapid Response**, and the **Elite** feature set. The Elite package includes Insights, Threat Actor Intelligence, High-Value Targeting, Advanced Data Pilfering, and Vulnerability Risk Intelligence.

> **All or nothing**
>
> Selecting the check box for any of these features applies that feature to all assets allocated to this client.

Permissions around these features are subject to some cascading behavior and restrictions:

- When a parent account is enabled for Tripwires, Rapid Response, or Elite features, access (but not necessarily allocations) will cascade to all that account's clients.
- When a parent account's Tripwires, Rapid Response, or Elite access is disabled, access is revoked for all client accounts. Notably, all their tripwires are disabled.
- Org Admins within each client can see that client's Tripwires, Rapid Response, and Elite toggles, and these Admins can manage these features only on that client.
- Only Org Admins, or users designated by the Org Admin for the client account, are enabled to see Notifications for the account, and to see the NodeZero Portal's **Tripwires** tab.

ALLOCATIONS AND LICENSES

Beyond the vertical parent/client relationships covered in the preceding section, keep in mind these overall constraints on allocating assets:

- As the `Org Admin` of a parent account, the number of assets you can allocate to all clients (combined) is limited by the total pool of licenses purchased and held by the parent account.
- Separately, granting access to Tripwires, Rapid Response, or Elite features to each client draws down the parent account's pool of licenses for that feature.
- When you select the **Client Management** page's **Asset Allocation** left tab, the header (highlighted below) shows total entitlements **Allocated** across all clients, followed by your pool's overall **Licensed** count.

Allocations Across Clients

The same **Client Management** page also shows you the number and percentage of allocated assets that **each client** has scanned. Here, it is important to:

- Monitor the percentage of (allocated) assets scanned on each client. Clients' Admins will see a warning banner when this percentage crosses a certain threshold. To avoid this, allocate headroom well above current usage.
- Monitor the total number of scanned assets. Reach out to your Horizon3 account manager to proactively acquire additional licenses when needed.

A parent account's Admin can reallocate assets among clients. See also reallocation details in Deleting Clients.

Increasing or decreasing the number of assets allocated to a client also depletes or restores your pool of any additional package features you've enabled on that client.

Restrictive or Flexible Allocations

A parent account's Admin can choose to enforce asset limits on clients either tightly (by assigning each client a specific number of assets) or loosely (by assigning them zero assets, which actually means unlimited).

- If you assign a specific number of assets to a client, then once that client’s scanned assets exceed the number allocated, they will be prevented from running further pentests. To enable restarting pentests, you'll need to allocate the client a higher number of assets than their scanned total.

  In this case, client Admins will see in-product warning banners as they approach 100% of their allocated assets.
- If you assign no assets to a client (zero), they will be able to run pentests uninterrupted, and scan as many assets as they want. These clients will draw assets from your total licensed pool until your licensed limit is reached.

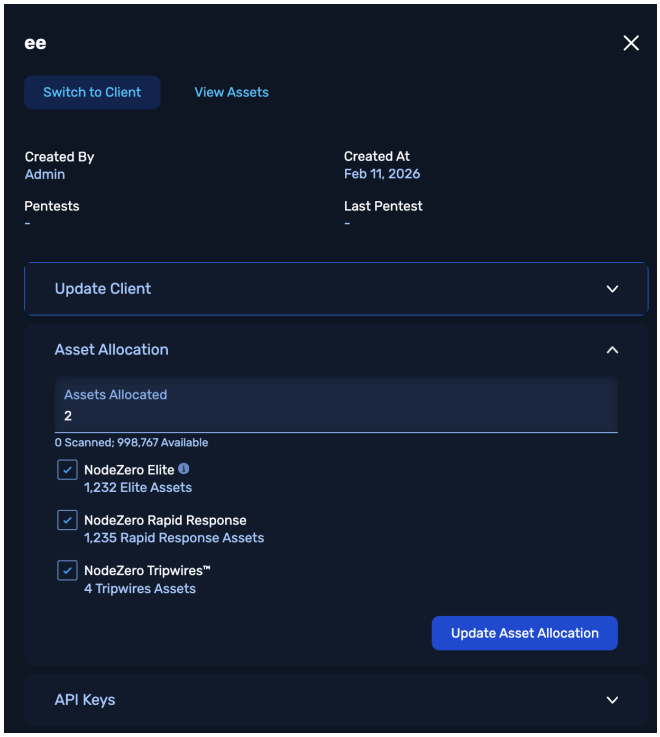
  In this case, you (as the parent Admin) will see in-product warning banners as you approach 100% of your licensed total, but clients will see no warning banners.

Bulk Allocations

To rebalance allocations among clients, the Admin can click the **Bulk Asset Allocation** button from the above view. This opens the **Allocate Licenses** page shown below. Subject to the limitations listed above (in Allocations and Licenses), this page makes it easy to update allocations of assets and features among multiple clients at once.

EDITING AN EXISTING CLIENT

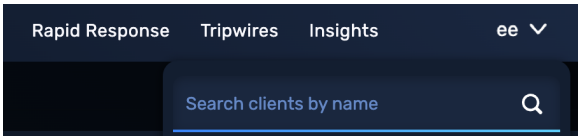
From the **Client Management** page, `Org Admins` of parent accounts can manage clients by selecting the account they wish to modify. In the resulting drawer, they can manage this client's account as outlined in the this page's preceding sections.



SWITCHING TO A CLIENT

From the client drawer shown above, you can click **Switch to Client** to pivot directly into a client account. This enables you to explore their penetration testing activity in depth – view results, findings, and trends as if you're operating within their environment.

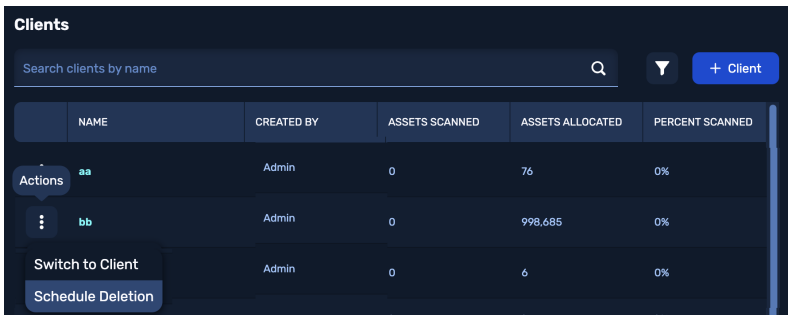
To switch back to the parent account, the top navigation bar provides the drop-down shown here.



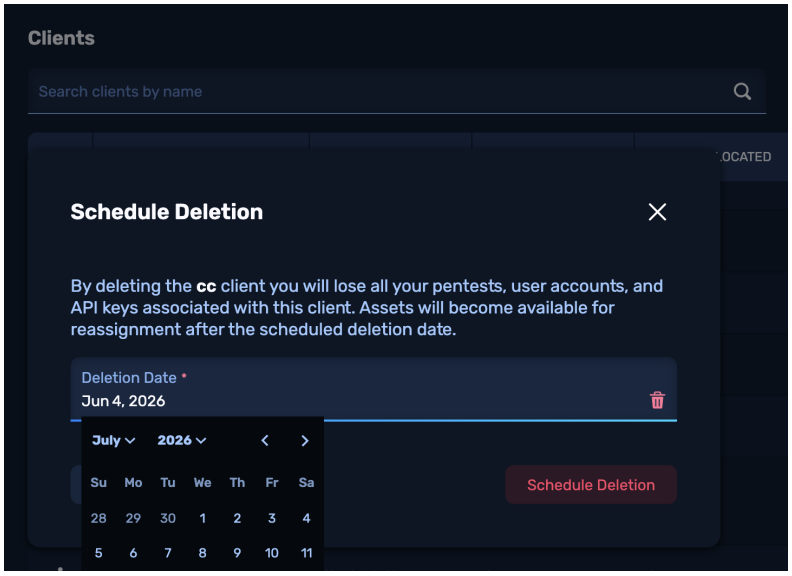
DELETING CLIENTS

To schedule deletion of an existing client:

1. Click the Actions menu (⋮) to the left of the client's name.
2. Select **Schedule Deletion**.



3. Accept the default deletion date (30 days out), or use the date picker to select a later date.
4. Select the **Schedule Deletion** button.



Until the scheduled date, you can prevent a client's deletion by reopening the client's Actions menu (⋮) and selecting **Cancel Scheduled Deletion**.

Marking a client for deletion affects the parent account's overall pool of assets as follows:

- The client's allocated number of assets, whether scanned or unused, are returned to the parent's pool on the deletion date.
- If you want to recover unused assets sooner, directly reduce this client's allocation to a non-zero number. This will immediately free up the additional assets for the parent to reallocate.

EXAMINING ASSETS

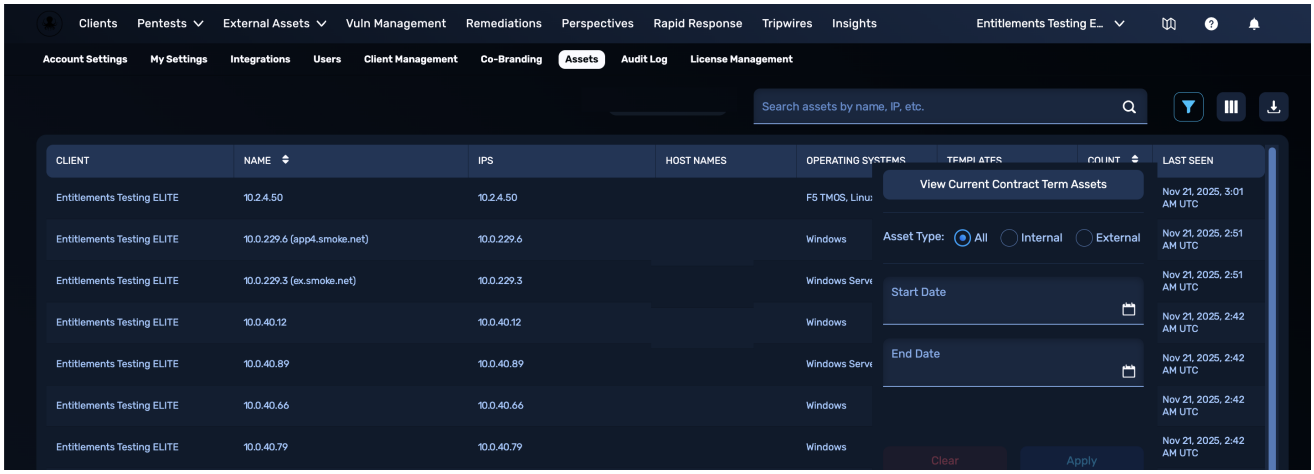
You can examine your inventory of assets at **Settings > Assets**. The table shows details about each asset's name, IPs, host names (where specified), OS, applied templates, count, and date last seen.

As shown below, the Filter button enables you to narrow down the displayed assets by:

- Internal.
- External.
- Date range.
- Current Contract Term authorization.

Click the Filter modal's **Apply** button to modify the display. The pagination controls at the page's lower-left corner will show the total count of assets matching your current filter conditions.

To view all your total assets again, reopen the Filter modal and click **Clear**.



3.4.8 Co-Branding

Co-Branding

Co-Branding enables you to customize the PDF reports that are generated once your pentest is complete. You can modify the logos, color scheme, and company information of the PDF reports so that they reflect your organization's unique brand.

> **Not Available on all license types**
>
> This setting applies to MSSP, C+, and eligible Flex parent accounts. If you need access to this feature, please contact your Customer Success representative.

In order for your co-branding settings to apply to generated PDF re..ports, use the following instructions to set up your Co-Branding page **before** starting a pentest.

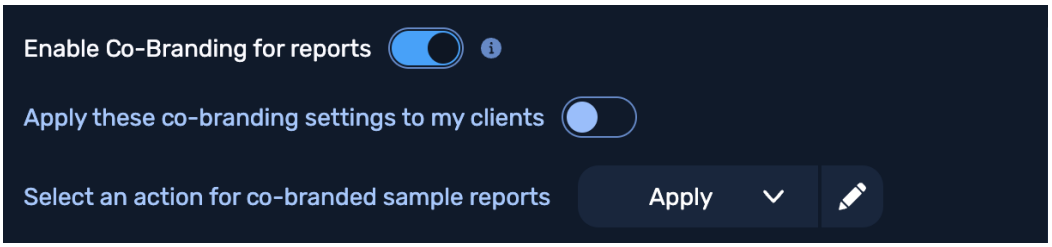
NAVIGATE TO THE CO-BRANDING PAGE

At the NodeZero Portal's top right, select your user profile, then select **Settings**. From the resulting **Settings** options, select **Co-Branding**.



ENABLE CO-BRANDING FOR ACCOUNTS

Turn on the **Enable Co-Branding for reports** toggle switch.



The toggle labeled **Apply these co-branding settings to my clients** has the following effects:

- When turned on, this enforces this parent account’s co-branding across all child accounts. (Admins of child accounts will be unable to override it.)
- Leave the toggle off to enable admins of each child account to independently configure their own co-branding. (Those admins will have access to the remaining controls and options described here.)

For the controls labeled **Select an action for co-branded sample reports**, see Co-Branding Sample Reports.

UPLOAD YOUR COMPANY LOGO

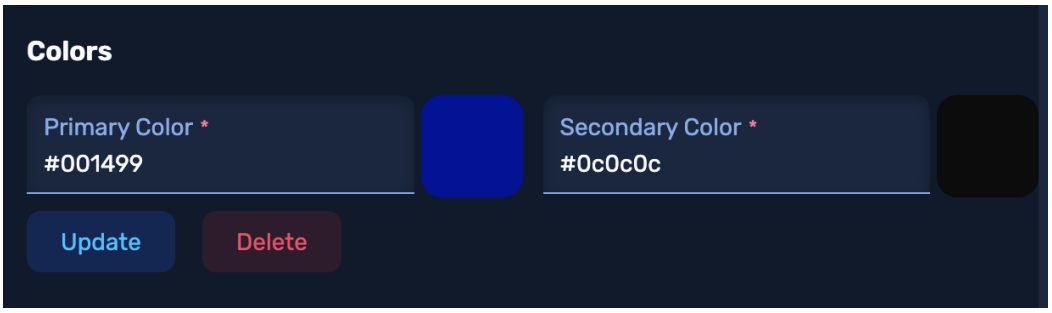
Upload a primary and (optional) secondary logo that meet the file format and size requirements.

Info

The primary logo will appear on the bottom of the PDF cover page. The secondary logo will appear on the top.

SELECT YOUR COMPANY COLORS

Select a primary and secondary color for the cover page, using the appropriate hex color value. Select the **Update** button to apply your changes.



Info

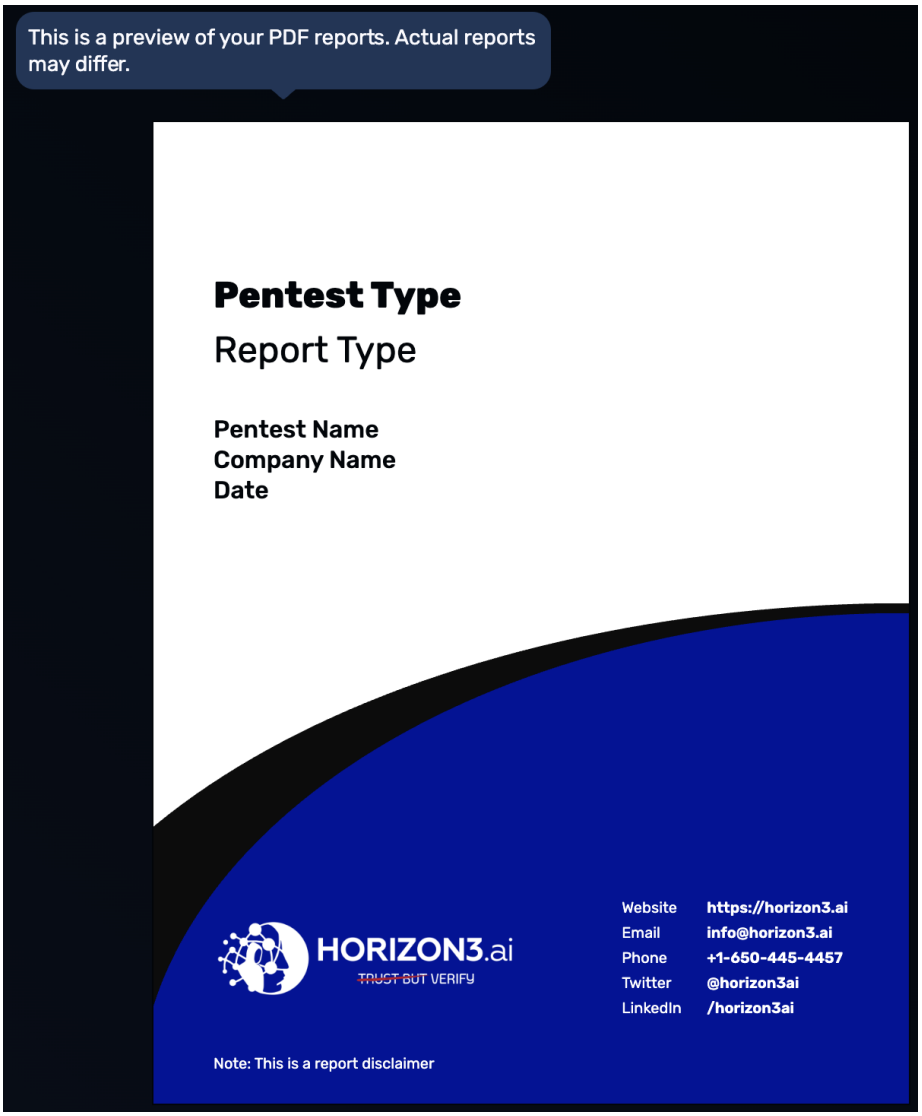
The primary color will appear as a background color on the bottom of the cover page. The secondary color will appear as an accent color above the primary color.

ENTER YOUR COMPANY INFORMATION

Enter your company information, such as your email address, website, phone number, X (Twitter) handle, and LinkedIn profile. Select the **Update** button to apply your changes.

PREVIEW YOUR PDF REPORTS

Preview the cover page of your co-branded PDF reports.

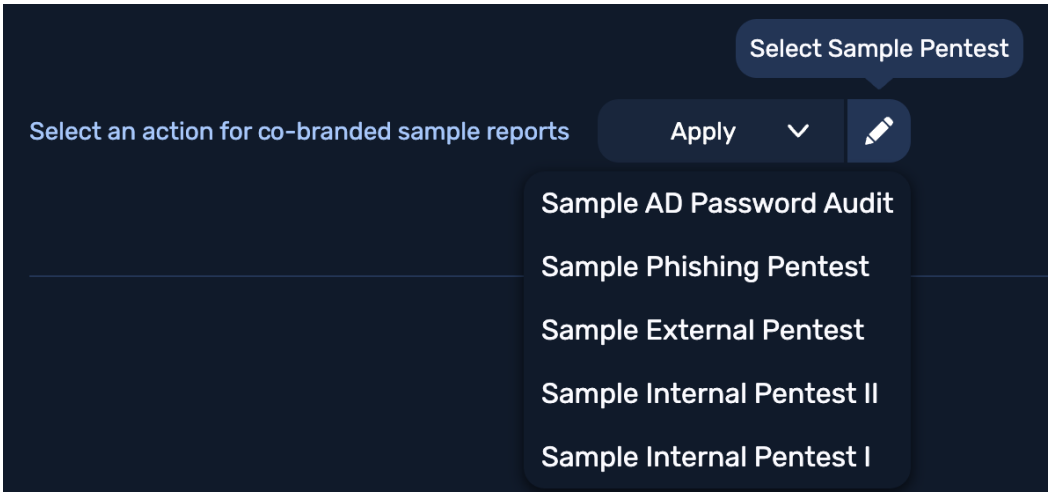


CO-BRANDING SAMPLE REPORTS

Two controls labeled **Select an action for co-branded sample reports** enable you to apply your organization's branding to the bundles of predefined reports available from the H3 Sample Account:

1. In the left drop-down, the default **Apply** option will email you a link to download your zipfile of reports after they're generated. Change this to **Download** if you prefer to download the zipfile (when ready) directly from the NodeZero Portal.
2. Click the pencil button on the right to open a drop-down that offers five sets of sample reports, as shown below.
3. Click the report set you want to generate.

These selections start a background job that generates the co-branded reports. Each of the five sets includes approximately four reports (e.g.: Executive Summary, Weaknesses, Fix Actions, and Segmentation Report).



The two **Internal Pentest** options differ in depth and primary audience:

- **Sample Internal Pentest I** reports typically demonstrate compromises that rely on relatively simple misconfigurations and weak credentials, rather than complex software exploits. Oriented toward an executive/management audience, these reports focus on basic security hygiene (robust password policies, disabling legacy protocols, etc.).
- **Sample Internal Pentest II** reports typically add technical depth, demonstrating how a chain of multiple weaknesses (such as lateral movement and privilege escalation) can add up to high-impact compromises like Domain Admin. Oriented toward a technical/SOC audience, these reports illustrate NodeZero's ability to execute complex technical maneuvers.

3.4.9 Integrations

Integrations

Configure 3rd-party integrations in NodeZero to seamlessly connect tools and workflows with our systems. We support the following integrations.

- Push integrations to support Vulnerability Management workflows:
- ServiceNow Vulnerability Response
- Ticketing platform APIs and Webhooks to support Vulnerability Management workflows:
- Jira
- Webhooks to support sending Tripwire alerts as events:
- Splunk
- Sentinel

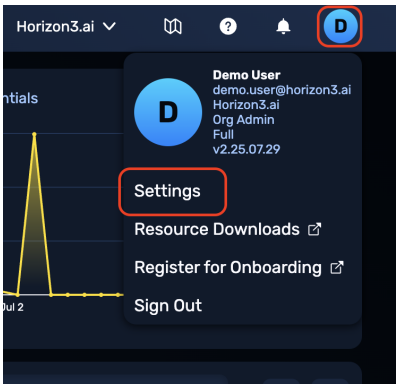
Jira Setup Guide

CONFIGURE NODEZERO VULNERABILITY MANAGEMENT APIS

Configure a ticketing platform integration for Vulnerability Management Hub

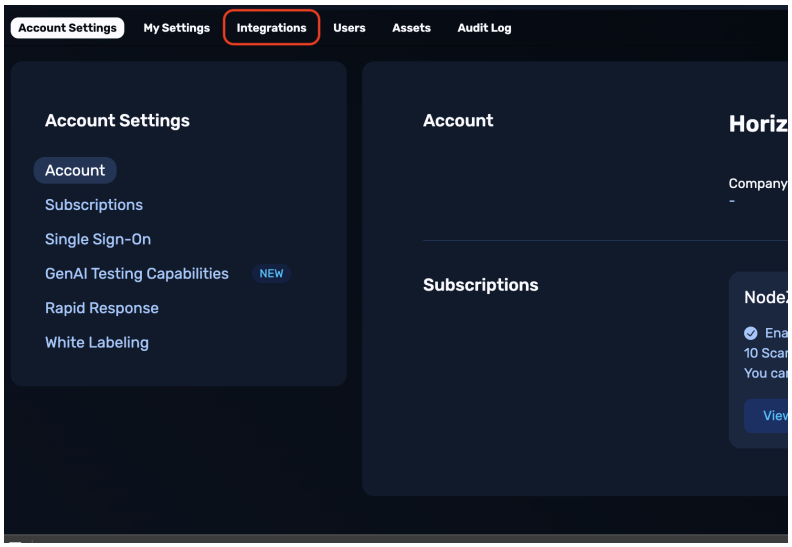
Navigate to the Settings Page

Click the user profile button in the top right of the Portal. Then in the drop-down, click `Settings`



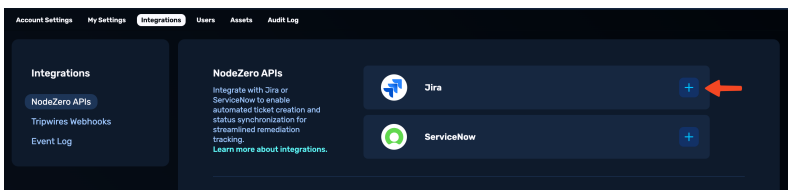
Navigate to the Integrations Page

From the subnav, click `Integrations`.



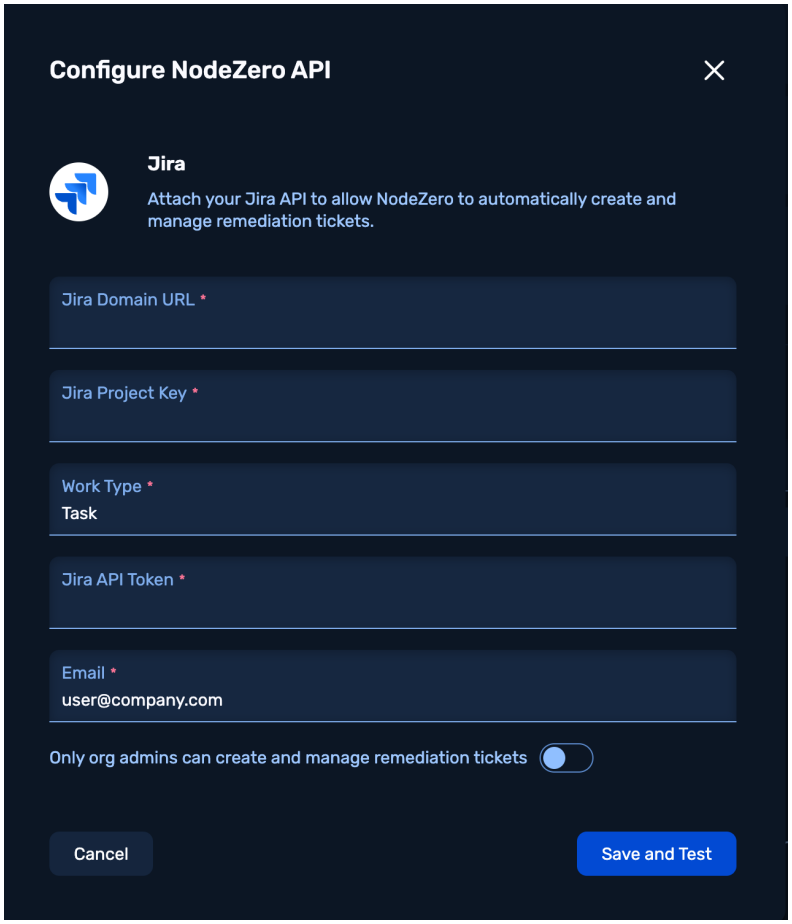
Start Configuration

In the `NodeZero APIs` section, choose **Jira** from the list. Then press the `+` button.



Save and Test Configuration

Enter the info required for the configuration:



Field	Description
Jira Domain URL	URL to your Jira instance (e.g. `https://[project name].atlassian.net`)
Jira Project Key	Enter the project key to the board where tickets will be dispatched. Project keys are used to prefix tickets (e.g. `NZDEMO-1` is a ticket in the project `NZDEMO`)
Work Type	Tickets are dispatched as Tasks. Make sure the `Task` issue type is enabled for the project.
Jira API Token	Token generated in Jira. For more instructions on how to create an API key for the integration, see Set Up API Key in Jira
Email	Email associated with the API key from above. Doesn't need to be the email address associated with the NodeZero account.
Only org admins can create and manage remediation tickets	Select whether only org admins can dispatch tickets or if all users within the org can.

CONFIGURE WEBHOOKS

Configure a webhook for Vulnerability Management. You must set up Jira ticketing integration before setting up a webhook (see Configure Vulnerability Management APIs for instructions)

> **Note**
>
> This section is for setting up Vulnerability Management Webhooks only. For Tripwires Webhooks, see Splunk or Sentinel.

Start Webhook Configuration

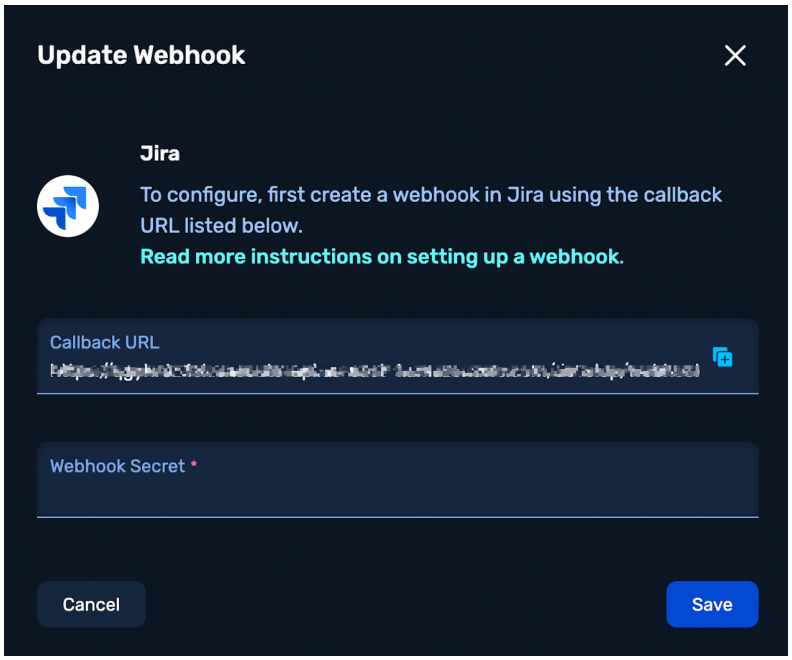
In the `Integrations` page in `Settings`, open up the details for the existing **Jira** integration. Click `Set Up Webhook`.



Save and Test Webhooks

A URL is provided to use when setting up a Webhook. For more instructions on how to create a Webhook for the Jira integration, see the section 4. Setup Webhook in Jira.

Once the Webhook is created in the ticketing platform, copy the token and paste in the form's required field.



SET UP API KEY IN JIRA

- Go to the API Tokens page in the Atlassian account settings.
- Click `Create API Token`. Fill out the form and click `Create`.
- Once created, copy the API token to paste in the Save and Test Configuration.

SET UP WEBHOOK IN JIRA

- In Jira, go to `System Settings`. Under `Advanced`, click `WebHooks`.
- You can also go directly to `{url}/plugins/servlet/webhooks` where `{url}` is the domain URL from the API integration config.
- Click `Create Webhook` and fill out the form.
- **URL**: URL provided from the Webhook form (see Save and Test Webhooks)

- **Secret**: Generate a secret. Then copy this field to paste in the Save and Test Webhooks
- **Issue Related Events**:
- In JQL, specify `project = {project}` where `{project}` is the full name of the project (not the key)
- Check `updated` under `Issue`
- Click `Create`

ServiceNow VR Setup Guide

This guide covers how to configure an integration that pushes vulnerability data from NodeZero to a ServiceNow Vulnerability Response (VR) module. The steps below are performed by (respectively) a ServiceNow and a NodeZero administrator. It includes the following major sections:

- How This Integration Works
- Prerequisites
- Run the Setup Script
- Create a Service Account
- Set Up Authentication
- Configure CI Matching
- Configure Integration in NodeZero
- What NodeZero Writes
- Verify the Initial Sync
- Using the NodeZero Integration

> **ServiceNow VR replaces ServiceNow Ticketing**
>
> This enhanced push integration replaces our earlier ServiceNow Ticketing integration, which NodeZero no longer supports.

HOW THIS INTEGRATION WORKS

The ServiceNow VR integration automatically pushes all weaknesses in your Vulnerability Management Hub (VMH) associated with hosts into ServiceNow's Vulnerability Response module.

The first sync between NodeZero and ServiceNow sends your all weakness series that is already in VMH. Each completed pentest after that triggers a delta sync as VMH gets updated – for new weaknesses and status changes only.

Every push includes the host attributes that Servicenow VR needs to match each finding to an existing Configuration Item in its CMDB (Configuration Management Database). These matches – using business rules and CI lookup rules – enable the resulting Vulnerable Items (VIs) to flow through ServiceNow VR's native auto-assignment and remediation processes.

What the Integration Doesn't Do

Bear in mind certain purposes that this integration was not designed to fulfill:

- No round-trips. Each sync is one-way only (NodeZero → ServiceNow). Nothing is read back.
- Never reopens a closed VI. (A regressed weakness becomes a new VI, per ServiceNow VR's out-of-the-box conditions.)
- Skips non-host findings by default. (Examples include domain credentials and cloud resources.)

PREREQUISITES

- A ServiceNow instance with the **Vulnerability Response** module installed.
- Admin access on the ServiceNow instance.
- Ability to run Background Scripts (**System Definition > Scripts > Background**) within ServiceNow.
- Admin access on the NodeZero instance.

RUN THE SETUP SCRIPT

A single idempotent setup script (provided below) provisions all the SecOps configuration elements that NodeZero needs — the dedicated integration role, ACLs (access control lists), the integration definition and instance, the severity mapping, and three business rules.

Run this script **before** you create the service account in Create a Service Account later in this procedure. The script creates the `sn_vul.nodezero_integration` role that you'll grant to the service account.

How to Run the Script

1. Click to expand the script below.
2. Copy its contents.
3. In ServiceNow, navigate to **System Definition > Scripts - Background**.
4. Set the scope drop-down (next to the **Run Script** button) to **Global**.
5. Paste the script into the editor.
6. Click **Run Script**.

Click to expand/collapse listing: `snow-setup.js`

```
/**
* NodeZero ServiceNow VR Integration — setup script (v2.1.0)
*
* Run from: System Definition → Scripts - Background (scope: Global).
* Idempotent — safe to re-run. Prints a summary at the end.
*
* Creates: dedicated `sn_vul.nodezero_integration` role (inherits
* `sn_vul.vulnerability_admin`), integration definition + instance,
* 6 ACLs + role grants (minimal validated set),
* 5 severity map rows, 3 business rules
* (CI match trigger via CILookupUtil + DI→VI cascades).
*
* Does NOT create: service account (credentials) or CI Lookup Rules
* (customer-designed against their CMDB). See Customer Setup Guide.
*
* Kept ES5-compatible (no template literals / arrow functions / let-const)
* so it runs on older SNOW Background Script engines without ES2021 mode.
*/
(function () {
    var VERSION = '2.1.0';
    var INTEGRATION_NAME = 'NodeZero';
    var INTEGRATION_ID = 'com.horizon3.nodezero';           // stable programmatic key for the integration definition
    var ROLE_NAME_BASE = 'sn_vul.vulnerability_admin';      // OOTB role we inherit from
    var ROLE_NAME_NZ = 'sn_vul.nodezero_integration';       // dedicated role granted to the integration service account

    var stats = { created: 0, skipped: 0, updated: 0, errors: 0 };
    var log = function (m) { gs.print('[NZ setup] ' + m); };
    var warn = function (m) { gs.warn('[NZ setup] ' + m); stats.errors++; };

    log('NodeZero setup script v' + VERSION + ' — starting');

    // ------------------------------------------------------------------ helpers

    function lookupSysId(table, field, value) {
        var gr = new GlideRecord(table);
        gr.addQuery(field, value);
        gr.setLimit(1);
        gr.query();
        return gr.next() ? gr.getUniqueValue() : null;
    }

    // lookup: string (addEncodedQuery) OR object {field: value} (addQuery per key).
    // Use object form when any value contains '^' (SNOW's AND separator).
    function upsert(table, lookup, fields, label) {
        try {
            var gr = new GlideRecord(table);
            if (typeof lookup === 'string') gr.addEncodedQuery(lookup);
            else for (var lk in lookup) gr.addQuery(lk, lookup[lk]);
            gr.setLimit(1);
            gr.query();
            if (gr.next()) {
                var changed = false;
                for (var k in fields) {
                    var cur = String(gr.getValue(k) || '');
                    var des = fields[k] === true ? '1'
                            : fields[k] === false ? '0'
                            : String(fields[k]);
                    if (cur !== des) { gr.setValue(k, fields[k]); changed = true; }
                }
                if (changed) {
                    gr.update(); stats.updated++;
                    log('UPDATED ' + label + ' (' + gr.sys_id + ')');
                } else {
                    stats.skipped++;
                    log('SKIPPED ' + label + ' — already matches (' + gr.sys_id + ')');
                }
                return gr.getUniqueValue();
            }
            var ins = new GlideRecord(table);
            ins.initialize();
            for (var f in fields) ins.setValue(f, fields[f]);
            var sysId = ins.insert();
            if (sysId) { stats.created++; log('CREATED ' + label + ' (' + sysId + ')'); return sysId; }
            warn('Insert returned no sys_id for ' + label); return null;
        } catch (e) { warn('Error on ' + label + ': ' + e); return null; }
    }

    function grantRole(aclSysId, roleSysId, label) {
        try {
            var gr = new GlideRecord('sys_security_acl_role');
            gr.addQuery('sys_security_acl', aclSysId);
            gr.addQuery('sys_user_role', roleSysId);
            gr.setLimit(1); gr.query();
            if (gr.next()) {
                stats.skipped++;
                log('SKIPPED role grant on ' + label + ' — already present');
                return;
            }
            var ins = new GlideRecord('sys_security_acl_role');
            ins.initialize();
            ins.setValue('sys_security_acl', aclSysId);
            ins.setValue('sys_user_role', roleSysId);
            if (ins.insert()) { stats.created++; log('CREATED role grant on ' + label); }
            else warn('Failed to grant role on ' + label);
        } catch (e) { warn('Error granting role on ' + label + ': ' + e); }
```

```
}

// ------------------------------------------------------------ resolve scopes

var scopeVR = lookupSysId('sys_scope', 'scope', 'sn_vul');
var scopeSecCmn = lookupSysId('sys_scope', 'scope', 'sn_sec_cmn');
var roleVulAdmin = lookupSysId('sys_user_role', 'name', ROLE_NAME_BASE);
if (!scopeVR || !scopeSecCmn || !roleVulAdmin) {
    gs.error('[NZ setup] FATAL — could not resolve required scopes or role. ' +
        'Is Vulnerability Response installed? Aborting.');
    return;
}
log('Resolved scopes: VR=' + scopeVR + ', SecCmn=' + scopeSecCmn +
    ', role ' + ROLE_NAME_BASE + '=' + roleVulAdmin);

// ------------------------------------------------- NodeZero Integration role
// Dedicated role granted to the integration service account. Inherits
// sn_vul.vulnerability_admin (so the service account gets the standard VR
// permissions) and on top of that receives the NodeZero-specific table
// write ACLs provisioned below. Customer assigns this role to the service
// account instead of granting sn_vul.vulnerability_admin directly, so the
// integration's privileges can be scope-tracked under one named role.

log('--- Role ---');
var roleNodeZero = upsert('sys_user_role', {name: ROLE_NAME_NZ}, {
    name: ROLE_NAME_NZ,
    description: 'Granted to the NodeZero integration service account. ' +
        'Inherits ' + ROLE_NAME_BASE + ' for read/write on standard VR tables; ' +
        'ACLs provisioned by snow-setup.js add the NodeZero-specific writes.',
    sys_scope: scopeVR
}, 'role ' + ROLE_NAME_NZ);
if (!roleNodeZero) {
    gs.error('[NZ setup] FATAL — could not provision NodeZero role. Aborting.');
    return;
}

// Inheritance: nodezero_integration contains vulnerability_admin
(function () {
    var gr = new GlideRecord('sys_user_role_contains');
    gr.addQuery('role', roleNodeZero);
    gr.addQuery('contains', roleVulAdmin);
    gr.setLimit(1); gr.query();
    if (gr.next()) {
        stats.skipped++;
        log('SKIPPED role inheritance ' + ROLE_NAME_NZ + ' contains ' + ROLE_NAME_BASE + ' — already present');
        return;
    }
    var ins = new GlideRecord('sys_user_role_contains');
    ins.initialize();
    ins.setValue('role', roleNodeZero);
    ins.setValue('contains', roleVulAdmin);
    if (ins.insert()) {
        stats.created++;
        log('CREATED role inheritance: ' + ROLE_NAME_NZ + ' contains ' + ROLE_NAME_BASE);
    } else {
        warn('Failed to create role inheritance');
    }
})();

// ------------------------------------------------------ Integration Definition

log('--- Integration Definition ---');
var integrationDefSysId = upsert('sn_sec_int_integration',
    'id=' + INTEGRATION_ID, {
        name: INTEGRATION_NAME,
        id: INTEGRATION_ID,
        source: INTEGRATION_NAME,
        integration_type: 'MANUAL',
        is_reapply_ci_lookup_supported: true,
        is_auto_close_supported: false,
        ire_source_name: 'VR-NodeZero',
        configurable: true,
        short_description: 'NodeZero pentest vulnerability data from Horizon3.ai'
    }, 'integration definition');
if (!integrationDefSysId) {
    gs.error('[NZ setup] FATAL — integration definition could not be created. Aborting.');
    return;
}

// ------------------------------------------------------- Integration Instance

log('--- Integration Instance ---');
upsert('sn_sec_int_impl',
    'name=' + INTEGRATION_NAME + '^integration=' + integrationDefSysId, {
        name: INTEGRATION_NAME,
        integration: integrationDefSysId,
        active: true,
        description: 'NodeZero pentest vulnerability data from Horizon3.ai'
    }, 'integration instance');

// ------------------------------------------------------------------- ACLs
// Identification: our ACLs are the only ones that grant the NodeZero role
// on this exact (name, operation, scope). Lookup is a JOIN against
// sys_security_acl_role — SNOW's ACLDescriber regenerates the description
// field after role grants, so a description marker wouldn't survive.

log('--- ACLs ---');
// Minimal set: inherited vulnerability_admin already covers record-level
// writes and most field writes; these add only what OOTB withholds.
var ACLS = [
```

```
    // [name, operation, scope]
    ['sn_vul_third_party_entry',             'create', scopeVR],     // TPE record-create (OOTB denies)
    ['sn_vul_third_party_entry.*',           'write',  scopeVR],     // TPE field writes (OOTB .* restrictive)
    // bypasses the OOTB conditional write ACL (vulnerabilityISEMPTY) that
    // otherwise strips this field on insert; net-new ACL.
    ['sn_vul_vulnerable_item.vulnerability', 'create', scopeVR],     // let vulnerability survive insert
    ['sn_vul_vulnerable_item.src_ci',        'write',  scopeVR],     // DI<->VI link (OOTB field ACL restrictive)
    ['sn_sec_cmn_src_ci',                    'write',  scopeSecCmn], // DI record update; also gates DI field writes
    ['sn_sec_cmn_src_ci.source',             'write',  scopeSecCmn]  // DI `source` tag (OOTB specific ACL restrictive)
];

function findAclByRoleGrant(aName, aOp, aScope) {
    var ar = new GlideRecord('sys_security_acl_role');
    ar.addQuery('sys_user_role', roleNodeZero);
    ar.addQuery('sys_security_acl.name', aName);
    ar.addQuery('sys_security_acl.operation', aOp);
    ar.addQuery('sys_security_acl.sys_scope', aScope);
    ar.setLimit(1);
    ar.query();
    return ar.next() ? ar.getValue('sys_security_acl') : null;
}

// Fallback: find our ACL directly if the role grant wasn't inserted yet
// (e.g., a prior run crashed between ACL insert and role grant insert).
// Without this, the next run would create a duplicate ACL.
function findAclDirect(aName, aOp, aScope) {
    var gr = new GlideRecord('sys_security_acl');
    gr.addQuery('name', aName);
    gr.addQuery('operation', aOp);
    gr.addQuery('sys_scope', aScope);
    gr.setLimit(1);
    gr.query();
    return gr.next() ? gr.getUniqueValue() : null;
}

for (var i = 0; i < ACLS.length; i++) {
    var aName = ACLS[i][0], aOp = ACLS[i][1], aScope = ACLS[i][2];
    var label = 'ACL ' + aName + '/' + aOp;
    // Don't set `description` — SNOW's ACLDescriber auto-generates it from
    // the role grant and overwrites anything we set.
    var fields = {
        name: aName, operation: aOp, type: 'record',
        decision_type: 'allow', active: true,
        advanced: false, admin_overrides: false,
        sys_scope: aScope
    };
    var aclSysId = findAclByRoleGrant(aName, aOp, aScope);
    var roleGrantExists = aclSysId !== null;
    if (!aclSysId) aclSysId = findAclDirect(aName, aOp, aScope);
    if (aclSysId) {
        // Update existing ACL in place
        var gr = new GlideRecord('sys_security_acl');
        gr.get(aclSysId);
        var changed = false;
        for (var k in fields) {
            var cur = String(gr.getValue(k) || '');
            var des = fields[k] === true ? '1'
                    : fields[k] === false ? '0'
                    : String(fields[k]);
            if (cur !== des) { gr.setValue(k, fields[k]); changed = true; }
        }
        if (changed) {
            gr.update(); stats.updated++;
            log('UPDATED ' + label + ' (' + aclSysId + ')');
        } else {
            stats.skipped++;
            log('SKIPPED ' + label + ' — already matches (' + aclSysId + ')');
        }
        if (roleGrantExists) {
            stats.skipped++;
            log('SKIPPED role grant on ' + label + ' — already present');
        } else {
            // ACL existed from a prior half-completed run; fix the missing role grant
            grantRole(aclSysId, roleNodeZero, label);
        }
    } else {
        // Create new ACL and grant role
        var ins = new GlideRecord('sys_security_acl');
        ins.initialize();
        for (var f in fields) ins.setValue(f, fields[f]);
        aclSysId = ins.insert();
        if (aclSysId) {
            stats.created++;
            log('CREATED ' + label + ' (' + aclSysId + ')');
            grantRole(aclSysId, roleNodeZero, label);
        } else {
            warn('Insert returned no sys_id for ' + label);
        }
    }
}

// ---------------------------------------------------------- Severity Mapping

log('--- Severity Mapping ---');
for (var sv = 1; sv <= 5; sv++) {
    upsert('sn_vul_severity_map',
        'source=' + INTEGRATION_NAME + '^source_value=' + sv, {
            source: INTEGRATION_NAME,
            source_value: String(sv),
            target_value: sv,
```

```
                integration_type: 'vr'
            }, 'severity map ' + sv + '→' + sv);
    }

    // ------------------------------------------------------------ Business Rules
    // Lookup by name — unique per NodeZero BR and doesn't collide with OOTB BRs
    // that may share the same collection + filter_condition. Names kept ≤40
    // chars to avoid SNOW's name-field truncation.

    log('--- Business Rules ---');

    function installBR(collection, filter, name, when, onInsert, order, script, desc) {
        upsert('sys_script',
            { name: name }, {
                name: name, collection: collection, when: when,
                action_insert: onInsert, action_update: true,
                action_delete: false, action_query: false,
                active: true, advanced: true, order: order,
                filter_condition: filter, script: script, description: desc
            }, 'BR: ' + name.replace(/^NodeZero\s*—\s*/, ''));
    }

    var NL = '\n';

    // CILookupUtil calls discoveredItem.update() internally; reevaluate_ci
    // must be reset to false beforehand or that nested update re-fires this BR.
    installBR(
        'sn_sec_cmn_src_ci', 'reevaluate_ci=true',
        'NodeZero — run CI match immediately', 'async', true, 100,
        [
            "(function executeRule(current, previous /*null when async*/) {",
            "    if (current.source.integration.id + '' != '" + INTEGRATION_ID + "') return;",
            "    current.setValue('reevaluate_ci', false);",
            "    new sn_sec_cmn.CILookupUtil().processDiscoveredItem(current);",
            "})(current, previous);"
        ].join(NL),
        'When a NodeZero DI is inserted/updated with reevaluate_ci=true, run CI Lookup Rules immediately via CILookupUtil with OOTB default
behavior. Unmatched DIs get a sn_sec_cmn_unmatched_ci placeholder linked as cmdb_ci, surfacing the host in the VR workspace.');

    // DI-side cascade — propagates cmdb_ci from a matched DI to its existing VIs.
    installBR(
        'sn_sec_cmn_src_ci', 'cmdb_ciISNOTEMPTY^cmdb_ciVALCHANGES^EQ',
        'NodeZero — DI→VI CI cascade (on DI)', 'after', false, 200,
        [
            "(function executeRule(current, previous /*null when async*/) {",
            "    if (current.source.integration.id + '' != '" + INTEGRATION_ID + "') return;",
            "    var vi = new GlideRecord('sn_vul_vulnerable_item');",
            "    vi.addQuery('src_ci', current.sys_id);",
            "    vi.setValue('cmdb_ci', current.getValue('cmdb_ci'));",
            "    vi.updateMultiple();",
            "})(current, previous);"
        ].join(NL),
        "When a NodeZero DI's cmdb_ci changes (e.g. matched by CI Lookup rule), cascade the CI reference to all VIs linked to this DI via
src_ci.");

    // VI-side cascade — fills cmdb_ci on new VIs from their linked DI.
    installBR(
        'sn_vul_vulnerable_item', 'source=NodeZero^cmdb_ciISEMPTY^src_ciISNOTEMPTY',
        'NodeZero — DI→VI CI cascade (on VI)', 'before', true, 100,
        [
            "(function executeRule(current, previous /*null when async*/) {",
            "    var diCmdbCi = current.src_ci.cmdb_ci + '';",
            "    if (diCmdbCi) current.setValue('cmdb_ci', diCmdbCi);",
            "})(current, previous);"
        ].join(NL),
        "New NodeZero VIs arrive after their DI has been inserted and CI-matched. The DI-side cascade rule updates existing VIs when DI.cmdb_ci
changes, but can't update VIs that don't exist yet. This rule fills in cmdb_ci on VI insert by dot-walking src_ci to cmdb_ci.");

    // ------------------------------------------------------------------ summary

    log('============================================================');
    if (stats.errors > 0) {
        // Use gs.warn directly — the `warn` helper increments stats.errors.
        gs.warn('[NZ setup] !!!  SETUP DID NOT FULLY COMPLETE — ' + stats.errors + ' ERROR(S) !!!');
        gs.warn('[NZ setup] Scan the log above for "WARN" lines to see what failed.');
        gs.warn('[NZ setup] The script is idempotent: fix the cause and re-run to finish setup.');
    } else {
        log('NodeZero setup complete.');
    }
    log('Created: ' + stats.created + ' / Updated: ' + stats.updated +
        ' / Skipped: ' + stats.skipped + ' / Errors: ' + stats.errors);
    log('============================================================');
    if (stats.errors === 0) log('Next: define CI Lookup Rules (see Customer Setup Guide).');
})();
```

What the Script Creates

- **Role**: `sn_vul.nodezero_integration` — dedicated role for the integration service account, inheriting `sn_vul.vulnerability_admin` for the standard VR permissions. Create a Service Account grants this role to the service account.
- 6 ACLs + role grants on the NodeZero data tables — each ACL granted to `sn_vul.nodezero_integration` (not `admin_overrides`), so the service account gets write access only via this specific role, rather than full admin override.

This is the minimal set: the inherited `sn_vul.vulnerability_admin` already covers record-level writes/updates and most field writes, so we only add what OOTB withholds:

- `sn_vul_third_party_entry` — create (record-level) + write (`*` field-level)
- `sn_vul_vulnerable_item` — create on the vulnerability field (works around an OOTB conditional write ACL that strips it on insert) + write on the src_ci field (OOTB-restricted)
- `sn_sec_cmn_src_ci` — write (record-level + source field-level; the record-level grant also enables the other DI field writes)
- `sn_sec_int_integration` record named `NodeZero` (integration definition to which CI Lookup Rules are scoped).
- `sn_sec_int_impl` record named `NodeZero`, linked to the definition (integration instance that tags every Discovered Item).
- 5 severity map rows (identity map, 1-to-1 on NodeZero's 1–5 scale).
- Three business rules on the CI matching path — installed on `sn_sec_cmn_src_ci` and `sn_vul_vulnerable_item` to trigger CI matching and cascade the CMDB (configuration management database) CI reference from Discovered Items to Vulnerable Items.

What the Script Does Not Create

- The service account — customer credentials (Create a Service Account below).
- CI Lookup Rules — these are customer-designed against your CMDB (Configure CI Matching below).

Script Output

Expected output on a first run:

```
[NZ setup] NodeZero setup script v2.1.0 — starting
[NZ setup] Resolved scopes: VR=..., SecCmn=..., role sn_vul.vulnerability_admin=...
[NZ setup] --- Role ---
[NZ setup] CREATED role sn_vul.nodezero_integration (...)
[NZ setup] CREATED role inheritance: sn_vul.nodezero_integration contains sn_vul.vulnerability_admin
[NZ setup] --- Integration Definition ---
[NZ setup] CREATED integration definition (...)
[NZ setup] --- Integration Instance ---
[NZ setup] CREATED integration instance (...)
[NZ setup] --- ACLs ---
[NZ setup] SKIPPED ACL sn_vul_third_party_entry/create — already matches (...)
[NZ setup] CREATED role grant on ACL sn_vul_third_party_entry/create
... (5 pre-existing OOTB ACLs: SKIPPED ACL + CREATED role grant)
[NZ setup] CREATED ACL sn_vul_vulnerable_item.vulnerability/create (...)
[NZ setup] CREATED role grant on ACL sn_vul_vulnerable_item.vulnerability/create
[NZ setup] --- Severity Mapping ---
[NZ setup] CREATED severity map 1→1 (...)
...
[NZ setup] --- Business Rules ---
[NZ setup] CREATED BR: run CI match immediately (...)
[NZ setup] CREATED BR: DI→VI CI cascade (on DI) (...)
[NZ setup] CREATED BR: DI→VI CI cascade (on VI) (...)
[NZ setup] ======================================================
[NZ setup] NodeZero setup complete.
[NZ setup] Created: 19
[NZ setup] Updated: 0
[NZ setup] Skipped: 5
[NZ setup] Errors:  0
[NZ setup] ======================================================
```

Script Results

5 of the 6 ACLs already exist OOTB on stock SecOps installs — the script **SKIPS** those ACL records and only **creates** the role grants linking them to `sn_vul.nodezero_integration`. The only net new ACL being created is `sn_vul_vulnerable_item.vulnerability` Counts mary vary slightly based on which OOTB records already exist on the target instance.

The script is idempotent. Re-running it produces `Created: 0 / Updated: 0 / Skipped: 32 / Errors: 0`. You can safely re-run it for verification, or to push script updates in future versions.

> **Do not rename script entities**
>
> The integration name `NodeZero` is a fixed backend contract, not a customer-configurable label. The NodeZero sync hard-codes it in multiple places — do not rename the integration definition, the instance, or other records that the script creates.

CREATE A SERVICE ACCOUNT

NodeZero connects to your ServiceNow instance via the REST Table API, using a dedicated service account. Authentication is via an API key (`x-sn-apikey` header).

1. In your ServiceNow dashboard, navigate to **User Administration > Users**.
2. Create a new user for the integration (e.g., `nodezero_integration`).
3. **Save** the record, then click **Set Password** (link below the form header) .
4. Scroll to the **Roles** tab at the bottom of the user record.
5. Click **Edit**.
6. Add `sn_vul.nodezero_integration` (created by the setup script in Run the Setup Script).
7. **Save** the service account.

This role inherits `sn_vul.vulnerability_admin` and adds the NodeZero-specific table write ACLs — together giving the integration the minimum permissions needed to create and update vulnerability records.

SET UP AUTHENTICATION

Here, you will create a dedicated API key token for API access. (For detailed configuration steps in the ServiceNow UI, see ServiceNow's Configure API key – Token-based authentication topic.)

> **Service account access**
>
> Enabling the API key plugin might disable basic auth for the service account. Creating an Inbound Authentication Profile (below) mitigates this risk.

Activate the API Key Plugin

Navigate to to **Admin Center > Application Manager**, search for `API Key and HMAC Authentication` (`com.glide.tokenbased_auth`), and activate this plugin.

Create an Inbound Authentication Profile

1. Navigate to **System Web Services > API Access Policies > Inbound Authentication Profile**.
2. Select **New**.
3. Select **Create API Key Authentication profiles**.
4. Enter a name for the profile.
5. Unlock the Auth parameter using the Lock button.
6. Select the target record: `x-sn-apikey` with Type: Auth Header.
7. Lock the Auth parameter.
8. Submit the profile.

Create the REST API Access Policy

1. Navigate to **System Web Services > API Access Policies > REST API Access Policies**.
2. Select **New**.
3. Enter a name for the policy.
4. From the **REST API** drop-down, select **Table API**.

5. Select **Apply to All Methods**, **Apply to All Resources**, and **Apply to All Versions**.
6. Leave **Apply to All Tables** unselected.

Select the Required Tables

Under the **Tables** related list, add the five tables that NodeZero needs:

- `sn_vul_third_party_entry` (read/write).
- `sn_vul_vulnerable_item` (read/write).
- `sn_sec_cmn_src_ci` (read/write).
- `sn_sec_int_impl` (read — used to resolve the NodeZero integration instance).
- `sn_vul_entry` (read) — parent table of TPE/NVD (Third-Party Entry/National Vulnerability Database) entries. Used to detect pre-existing vulnerability entries before insert so the out-of-the-box "Avoid duplicate Vuln entry" business rule doesn't abort our TPE writes.

Submit the Profile

Under **Inbound authentication profiles** at the bottom, click to add a new row, add the profile created above in Create an Inbound Authentication Profile, and Submit the change.

Generate and Copy the API Key

1. Navigate to **System Web Services > API Access Policies > REST API Key**.
2. Select **New**.
3. Set a name for the key.
4. Select the Service Account name you created above in Create a Service Account.
5. Submit the new key.
6. Navigate to the row that shows this new key's name.
7. Copy the generated token (click the Lock button to access it).
8. Save the token to later submit to NodeZero in Configure Integration in NodeZero.

CONFIGURE CI MATCHING

NodeZero writes Discovered Items (DIs) with host attributes in the `source_data` JSON field (and an os column at the top level). ServiceNow's CI Lookup framework uses those attributes to match each DI to a Configuration Item in your CMDB. The matched CI reference then cascades down to every Vulnerable Item (VI) for that host via business rules installed in Run the Setup Script.

The matching rules themselves are **yours to design** — attributes, priority, and CI class scope depend on your CMDB. The integration- and cascade-side plumbing installed by the setup script is rule-agnostic.

Available Attributes

For each discovered host, NodeZero provides the following attributes inside `source_data`:

Attribute	Notes
`ip_address`	Always available.
`dns_name` / `fqdn`	Available for most hosts.
`subnet`	CIDR notation in most cases (e.g., `10.0.0.0/24`). A small fraction (~1%) might contain a bare IP address for certain endpoint types — CI Lookup Rules that strictly require CIDR should include a null/invalid fallback.
`ldap_hostname`	AD-joined hosts only.
`host_name`	Hostname from endpoint discovery.
`mac_address`	Might not be available for most hosts — not recommended for CI matching.

Operating System is provided as a top-level field (`os`) on the Discovered Item record, not inside `source_data` .

Example: AD-Joined Windows Server (All Attributes)

```
{
  "ip_address": "192.168.1.10",
  "host_name": "dc01.corp.example.com",
  "dns_name": "dc01.corp.example.com",
  "fqdn": "dc01.corp.example.com",
  "ldap_hostname": "dc01",
  "subnet": "192.168.1.0/24",
  "mac_address": "00:50:56:84:46:30"
}
```

Example: Host with Differing Name Sources

```
{
  "ip_address": "10.20.30.40",
  "host_name": "app-server.internal.net",
  "dns_name": "ip-10-20-30-40.ec2.internal",
  "fqdn": "ip-10-20-30-40.ec2.internal",
  "ldap_hostname": "app-server",
  "subnet": "10.20.30.0/24"
}
```

Example: Linux Host (No AD, No MAC)

```
{
  "ip_address": "10.50.60.70",
  "host_name": "web-proxy-01.internal",
  "dns_name": "web-proxy-01.internal",
  "fqdn": "web-proxy-01.internal",
  "subnet": "10.50.60.0/23"
}
```

Define CI Lookup Rules

How NodeZero Discovered Items get matched to Configuration Items in your CMDB is **your call** — which attributes to trust, in what priority order, against which CI classes or custom fields, depends on your data model and the quality of each attribute in your environment. Rules live in `sn_sec_cmn_ci_lookup_rule.list` , each scoped to the NodeZero integration definition created by the setup script.

Source fields available (keys in the DI's `source_data` JSON, see Available Attributes): `fqdn` , `dns_name` , `ip_address` , `host_name` , `ldap_hostname` , `subnet` , `mac_address` .

Recommended Starter Rules - Sample (tune or replace based on your CMDB):

Name	Order	Lookup Method	Active	Source	Source Field	De
FQDN - NodeZero Ingest	100	`field_matching`	✔	`NodeZero`	`fqdn`	Mo hig
IP Address - NodeZero Ingest	200	`field_matching`	✔	`NodeZero`	`ip_address`	Au ow `ow`
Hostname - NodeZero Ingest	300	`field_matching`	✔	`NodeZero`	`host_name`	Fa ar

Lower `Order` = higher priority. The first rule that produces a match wins; subsequent rules don't run for that DI.

Rules Are Changeable

The NodeZero integration itself does not depend on these rules — the match-trigger business rule installed by the setup script iterates whatever rules exist for the NodeZero source. Add, remove, or rewrite rules at any time without touching the rest of the configuration.

Unmatched DI Behavior

DIs for hosts that don't match a CMDB CI get a **placeholder record** created in `sn_sec_cmn_unmatched_ci` and linked as the DI's `cmdb_ci`. These placeholders surface unmatched hosts in the VR workspace so that they can be found and resolved. To resolve a placeholder:

- Create or discover the real CI in your CMDB; the next NodeZero sync will match the DI to it (because every upsert resets `reevaluate_ci = true`); **or**:
- Match the DI manually: set `cmdb_ci` directly on the DI, and set `matching_type = matched_manually`.

Verify CI Matching

After the next NodeZero sync:

- `sn_sec_cmn_src_ci` filtered by `Source` = `NodeZero` + `Configuration item is not empty` > count equals the number of NodeZero hosts that match a CMDB CI via your rules. Zero is legitimate if no CIs yet exist for NodeZero's hosts.
- `sn_vul_vulnerable_item` filtered by Source = `NodeZero` + `Configuration item is not empty` > count grows as matched DIs' VIs inherit the CI reference.
- Spot-check a VI in the UI — the **Configuration item** field should resolve to the matched CMDB CI.
- As your CMDB grows (via Discovery or manual additions), re-running a NodeZero sync or setting `reevaluate_ci` = true on unmatched DIs picks up the newly available CIs.

Troubleshooting

DIs stay unmatched after a sync:

- Confirm the `sn_sec_int_impl` record has its Integration field pointing at the NodeZero definition (both created by the setup script; verify via `sn_sec_int_impl.list`).
- Confirm each lookup rule's Source field points at the NodeZero integration definition (not the instance).
- Confirm DIs carry `reevaluate_ci` = true (NodeZero sets this automatically on every upsert).
- Confirm the NodeZero — run CI match immediately business rule is active.

VI Configuration item **empty even though the DI is matched:**

- Confirm both NodeZero — DI > VI CI cascade business rules are active (one scoped to `sn_sec_cmn_src_ci`, one to `sn_vul_vulnerable_item`).
- Confirm the VI's `src_ci` points at the DI (NodeZero sets this automatically).

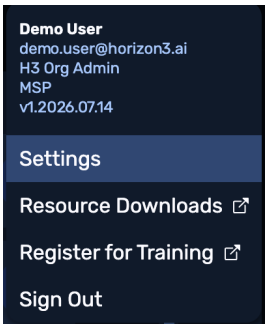
Placeholder CI records in `sn_sec_cmn_unmatched_ci` :

- Expected behavior — the setup script's CI matching business rule calls `CILookupUtil.processDiscoveredItem(current)` with the ServiceNow out-of-the-box default. When no CI Lookup Rule matches, the framework creates an unmatched CI placeholder via IRE, which surfaces unmatched hosts in the VR workspace. See Unmatched DI Behavior for resolution steps.

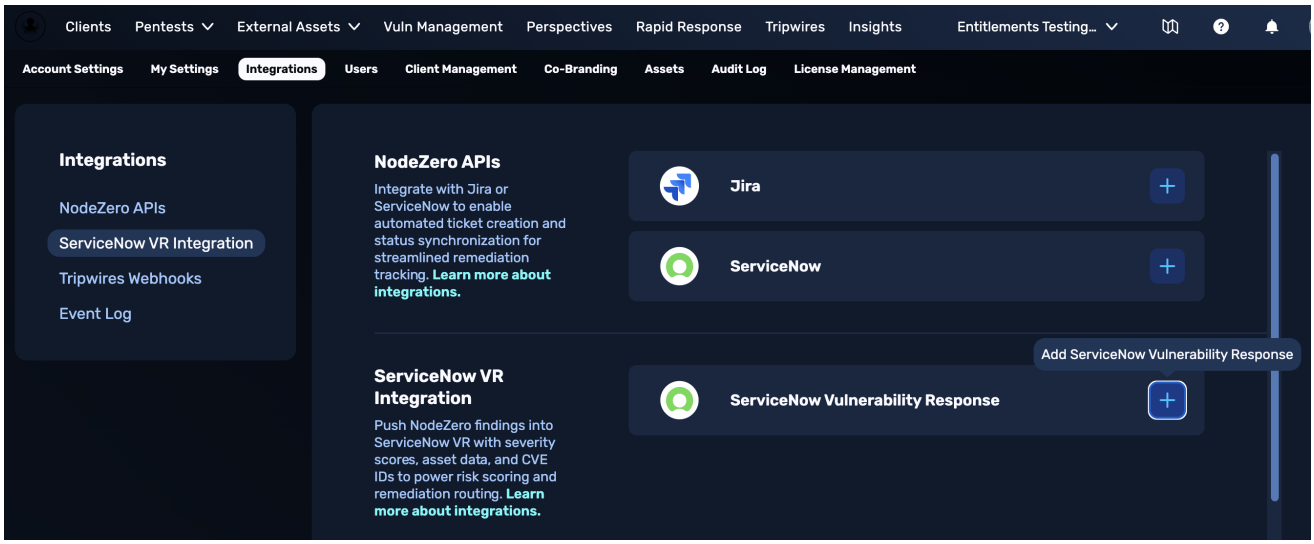
CONFIGURE INTEGRATION IN NODEZERO

In the NodeZero Portal, use the steps below to configure the **ServiceNow VR Integration** to connect with your ServiceNow instance.

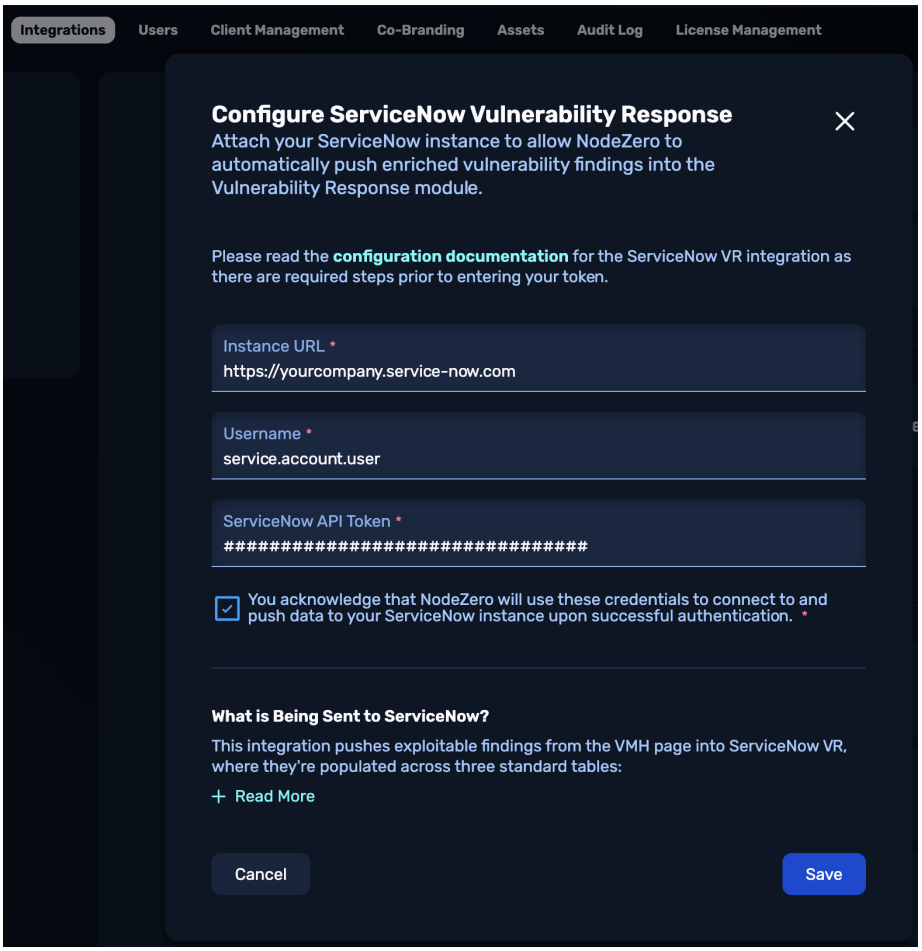
1. Click the user profile button at the top right, and from the resulting drop-down, select **Settings**.



2. As shown below: From the Portal's upper submenu, select **Integrations**.
3. Locate the **ServiceNow Vulnerability Response** tile and click its `+` button.



4. Enter your ServiceNow instance's URL, username, and token. See guidance in the table below.



Field	Value
Instance URL	URL to your ServiceNow instance (e.g., `https://yourcompany.service-now.com`)
Service account username	The username you assigned earlier in Create a Service Account.
API key token	The token you generated earlier in Generate and Copy the API Key.

5. Select the **You acknowledge...** check box, then click **Save**.

> **Credentials encryption**
>
> Credentials are encrypted at rest using per-client AWS KMS (Key Management Service) keys.

WHAT NODEZERO WRITES

NodeZero pushes data to three ServiceNow tables:

Table	What
`sn_sec_cmn_src_ci`	**Discovered Items** — one per unique host. Contains IP, hostname, DNS, MAC, subnet for CI matching.
`sn_vul_third_party_entry`	**Vulnerability Definitions** — one per unique CVE or NodeZero (H3 taxonomy) finding. Contains severity, CVSS scores, description, solution.
`sn_vul_vulnerable_item`	**Vulnerable Items** — one per vulnerability-on-host combination. Links to the discovered item and vulnerability definition.

Data is synced automatically after each pentest completes. Each sync is idempotent — existing records are updated, new records are created, and no duplicates are produced.

By default, NodeZero pushes only findings that can be linked to a host. Findings without a host context (e.g., domain-level credentials, cloud resources, dangling DNS) are filtered out, so that your ServiceNow vulnerability view stays actionable. Customers whose environments rely on non-host findings can opt in to pushing them as vulnerable items without a `src_ci` link — ask your Horizon3 AI representative to enable this.

VERIFY THE INITIAL SYNC

After the first NodeZero sync, verify in your ServiceNow instance:

1. **Vulnerable Items**: type `sn_vul_vulnerable_item.list` > filter by `Source` = `NodeZero`.
2. **Vulnerability Definitions**: type `sn_vul_third_party_entry.list` > filter by `Source` = `NodeZero`.
3. **Discovered Items**: type `sn_sec_cmn_src_ci.list` > filter by `Source` = `NodeZero`.

Check that:

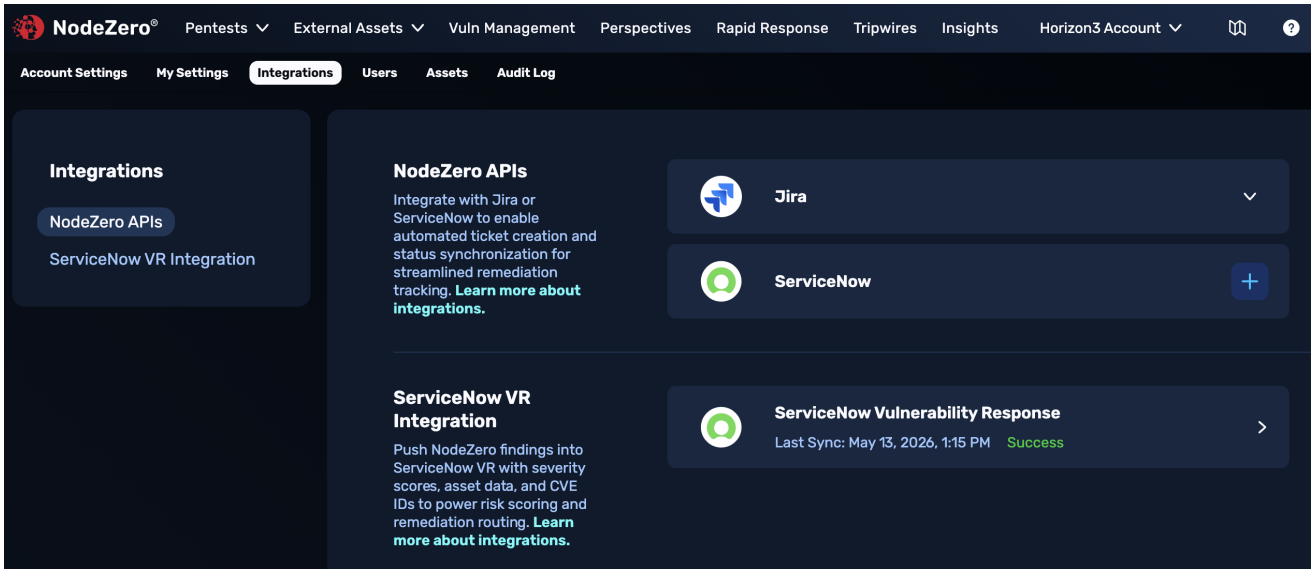
- Vulnerability definitions have severity, CVSS scores, and descriptions populated.
- Vulnerable items link to both a vulnerability definition and a discovered item.
- Discovered items have `source_data` populated with host attributes.
- Risk scores vary based on severity (not all 20 or all 0).
- For DIs that matched a CMDB CI, the linked Vulnerable Items have `cmdb_ci` populated as well (see Verify CI Matching for CI matching-specific verification).

If fields appear blank, or all risk scores are identical, the setup script might not have completed cleanly. Re-run it (idempotent) and check the log for errors.

If Discovered Items don't appear when filtering by Source = NodeZero, verify that the Integration Instance (`sn_sec_int_impl` record named `NodeZero`) exists and is linked to the NodeZero integration definition. The setup script creates both; re-running it restores missing state.

USING THE NODEZERO INTEGRATION

After the integration performs a successful sync with your ServiceNow instance, the **ServiceNow Vulnerability Response** tile will confirm this status.



Click anywhere on the tile to open the dashboard shown below. This summarizes past pushes to ServiceNow, with metadata and links for discovered vulnerabilities and vulnerable items.

NodeZero®
Pentests
External Assets
Vuln Management
Perspectives
Rapid Response
Tripwires
Insights
Horizon3 Account
Account Settings
My Settings
Integrations
Users
Assets
Audit Log
Back to Integrations
ServiceNow Vulnerability Response Push History
Last Sync: May 13, 2026, 1:15 PM Success
Sync Status
Success
Discovered Items to CI
3
Discovered Items
765
Vulnerability Definition
278
Vulnerable Items
1,097
Refresh Stats
DATE
TYPE
STATUS
ROWS
FAILED
DISCOVERED ITEMS
VULNERABILITY DEFINITION
VULNERABLE ITEMS
DURATION
May 13, 2026, 1:05 PM Full Success 3,691 0 412 171 3,691 10m 12s
May 13, 2026, 12:53 PM Full Success 3,691 0 412 171 3,691 9m 58s
May 7, 2026, 10:42 AM Full Success 3,691 0 412 171 3,691 8m 19s
May 6, 2026, 10:45 AM Full Failed 0 0 0 0 0 5m
May 6, 2026, 10:31 AM Full Success 3,691 0 412 171 3,296 8m 14s

ServiceNow Ticketing Setup Guide (Deprecated)

Horizon3.ai's original ServiceNow integration path is deprecated and we no longer support it. Please instead use our upgraded ServiceNow Vulnerability Response (VR) push integration.

3.5 NodeZero Runner

3.5.1 NodeZero Runner

The NodeZero Runner automates internal penetration testing, enabling you to schedule and manage recurring pentests remotely from the Portal. By deploying a Runner on your NodeZero Host, you can eliminate manual script execution and ensure consistent security assessments across your network.

What Is a Runner

A NodeZero Runner is a component of your NodeZero Host that automates the deployment of the NodeZero container, for pentests internal to your physical or virtual infrastructure. It syncs with the Horizon3.ai API to identify new pentest assignments, and launches NodeZero to execute them.

This eliminates the need for manual intervention. The Runner acts as a bridge between your internal (or virtual) network and the Horizon3.ai Portal, enabling automated security testing.

Where a Runner Applies

You can use a Runner to automate pentests that simulate attacks that originate within your physical network or your cloud infrastructure. These include the following tests:

- Internal
- Kubernetes
- Azure Entra ID
- AD Password Audit
- Segmentation
- Phishing
- Insider Threat

Runners are neither compatible nor required with external pentests or other tests that simulate attacks from an outsider's perspective. To launch these tests, NodeZero deploys automatically in the Horizon3.ai cloud. You can schedule these tests via the NodeZero Portal, CLI, or API.

When to Use a Runner

Deploy a NodeZero Runner in these scenarios:

- **Automate Testing:** You want to streamline pentest deployment without manually running scripts on your NodeZero Host.
- **Schedule Recurring Tests:** You need to perform regular pentests to maintain continuous security monitoring.
- **Scale Across Networks:** You aim to test multiple network segments or perspectives by running multiple Runners.

3.5.2 Installation

Installation: Setting Up a Runner

There are two methods to set up a NodeZero Runner, each tailored to different user needs and technical preferences. Both methods enable you to automate and schedule internal penetration tests efficiently through the Horizon3.ai Portal.

1. **Easy Install Script (Recommended)**

   This method uses a pre-generated script from the NodeZero Portal to automate the entire setup process with a single command. It's ideal for users who want a quick, hassle-free setup with minimal manual intervention, especially in standard environments where default settings are sufficient.

2. **Manual Setup with h3-cli**

   This method involves a hands-on approach, giving you full control over each step of the setup process. It's best for advanced users who need to customize the configuration or troubleshoot specific network conditions.

Both methods allow you to leverage the Runner's capabilities for automation, including scheduling recurring pentests and monitoring results directly from the Portal.

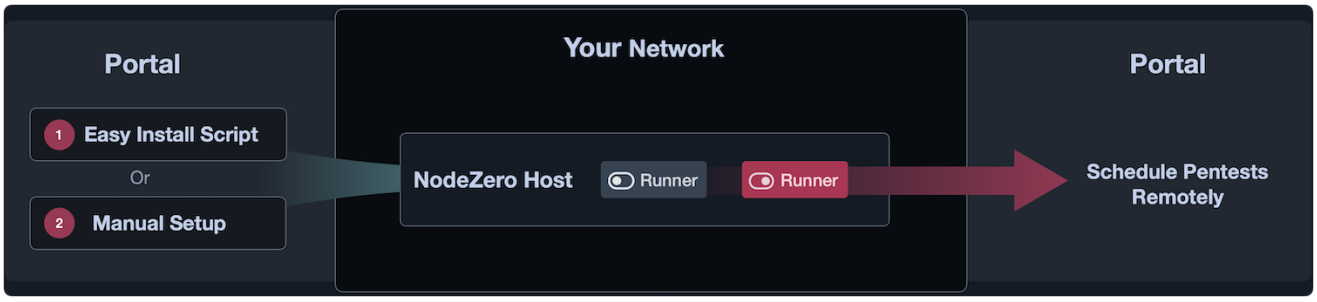


Figure 1 - Overview of the two activation methods: Easy Install Script and Manual Setup.

CHOOSE A SETUP METHOD

Select one of the methods below to set up your NodeZero Runner.

Easy Install Script — Recommended

Deploy a NodeZero Runner by creating an Easy Install Script in the Portal.

Manual Setup

Deploy a NodeZero Runner by using the h3-cli Command Line Tool.

Easy Install Script

NODEZERO RUNNER WITH EASY INSTALL SCRIPT

This guide walks you through setting up a NodeZero Runner on your NodeZero Host using the Easy Install Script. A NodeZero Runner simplifies internal pentesting by automating the deployment of the NodeZero container, allowing you to schedule and execute pentests directly from the Horizon3.ai Portal without manual intervention.

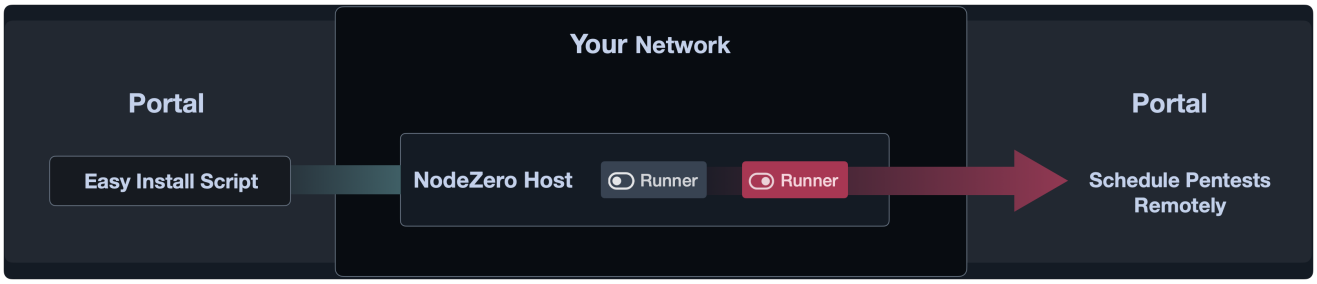


Figure 1 - Setting up a NodeZero Runner using the Easy Install Script, from initiation in the Portal to scheduling pentests and viewing results.

What is the Easy Install Script?

The Easy Install Script is a one-step solution for setting up a NodeZero Runner on your NodeZero Host. It automates the installation of necessary tools, configures the Runner, and registers it with the NodeZero Portal. This method is ideal for users who want a fast, hassle-free setup with minimal manual steps.

Note: NodeZero Runners are designed exclusively for internal pentests. For external pentests, NodeZero is deployed automatically in the Horizon3.ai cloud.

When to use

The Easy Install Script is the best choice in the following scenarios:

- You need to quickly deploy a NodeZero Runner with default settings.
- You prefer an automated setup process over manual configuration.
- Your environment supports `systemd` for auto-restart functionality (Linux-based NodeZero Hosts).

If you require custom configurations or your system lacks `systemd`, consider the Manual Setup method instead.

Requirements

Run the Easy Install command as the same non-root user who will own and manage the Runner over the long term. Make sure that user has Docker or Podman privileges. The script installs `h3-cli` into that user's environment (by default, under the current directory and `~/.h3`) and starts/manages the Runner from that account.

Before proceeding, ensure you have:

- A NodeZero Portal account.
- A NodeZero Host within your private network, running a Linux distribution with a container runtime (Docker or Podman) installed.
- SSH or terminal access with permissions to run installation commands.

If you do not meet these requirements, see steps 1 and 2 in the Quickstart Guide.

Step-by-step instructions

Follow these steps to install and configure a NodeZero Runner using the Easy Install Script:

1. **Access the Runners page:**

- Log in to the NodeZero Portal and navigate to Runners.

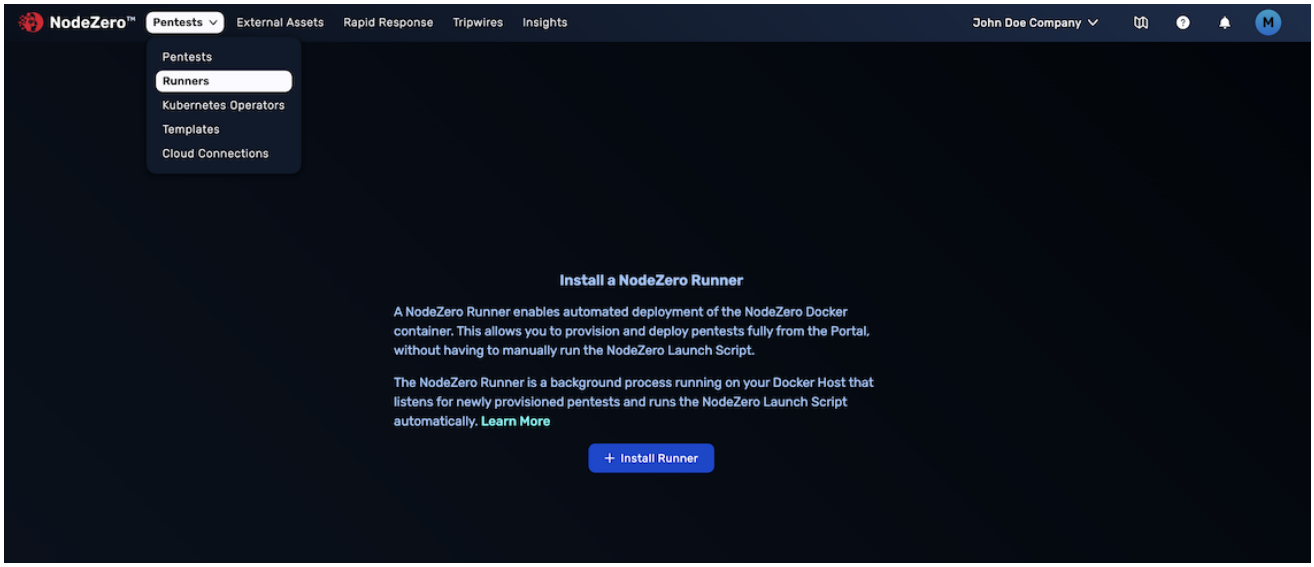


Figure 2 - Screenshot of the NodeZero Runner page

2. **Initiate Runner installation:**

- Click the **Install Runner** button.
- Enter a unique, descriptive name for your Runner (e.g., "Main-Office-Runner"). Choose a name that reflects the Runner's location or purpose for easy identification later.
- Click **Submit**.

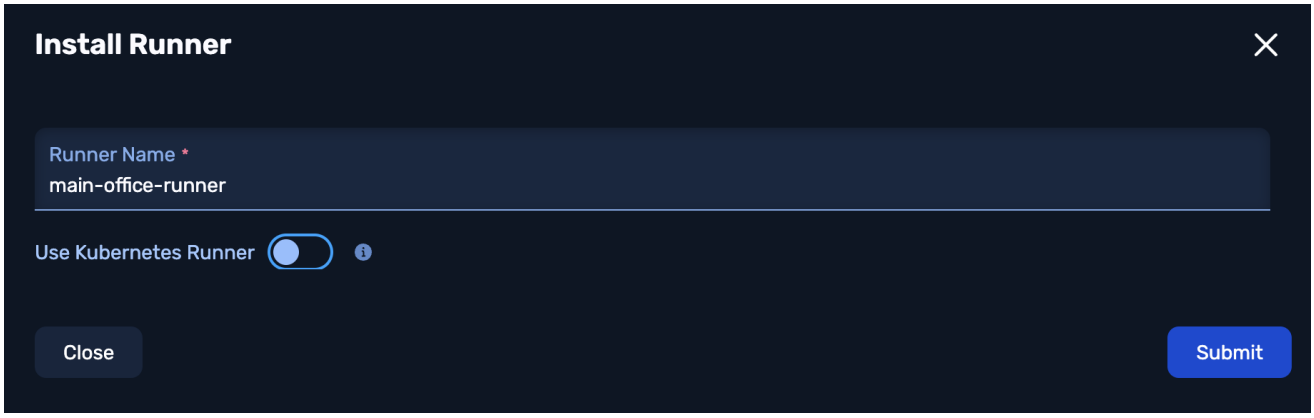


Figure 3 - Screenshot of the Install NodeZero Runner form

3. **Obtain the installation command:**

- The Portal will display a command starting with `curl` or `wget`. Copy this command to your clipboard.

Figure 4 - Screenshot of the NodeZero Runner initialization command

4. **Execute the command on your NodeZero Host:**

- SSH into your NodeZero Host or open a terminal.
- Paste and run the copied command.

```
ssh nodezero@xxx.xxx.xxx.xxx  // Replace with IP address of your NodeZero Host.
```



Figure 5 - Screenshot of pasting Runner command into Docker host terminal

- Wait for the script to complete; it will install `h3-cli`, configure the Runner, and start the service.
- If you encounter permission issues, ensure you are running as a user with Docker privileges.

```
● nodezero-runner-main-office-runner.service - NodeZero Runner main-office-runner
     Loaded: loaded (/etc/systemd/system/nodezero-runner-main-office-runner.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 21:11:02 UTC; 14ms ago
       Docs: https://docs.horizon3.ai/
   Main PID: 107146 (h3-start-agent)
      Tasks: 3 (limit: 9445)
     Memory: 1.0M ()
     CGroup: /system.slice/nodezero-runner-main-office-runner.service
             ├─107146 /bin/bash /home/nodezero/h3-cli/bin/h3-start-agent main-office-runner /tmp/main-office-runner.log
             └─107152 dirname /home/nodezero/h3-cli/bin/h3

Mar 03 21:11:02 nodezero systemd[1]: Started nodezero-runner-main-office-runner.service - NodeZero Runner main-office-runner.
[Mon Mar  3 09:11:02 PM UTC 2025] INFO: NodeZero Runner service "nodezero-runner-main-office-runner" is active and enabled under systemd.
[Mon Mar  3 09:11:02 PM UTC 2025]       The Runner service is configured to start automatically on system boot.
[Mon Mar  3 09:11:02 PM UTC 2025] INFO: Stop the service via:
[Mon Mar  3 09:11:02 PM UTC 2025]       systemctl stop "nodezero-runner-main-office-runner"
[Mon Mar  3 09:11:02 PM UTC 2025] INFO: Restart the service via:
[Mon Mar  3 09:11:02 PM UTC 2025]       systemctl start "nodezero-runner-main-office-runner"
[Mon Mar  3 09:11:02 PM UTC 2025] INFO: Stream the NodeZero Runner logs via:
[Mon Mar  3 09:11:02 PM UTC 2025]       h3 tail-runner "main-office-runner"
[Mon Mar  3 09:11:02 PM UTC 2025] INFO: NodeZero Runner installation complete.
nodezero@nodezero:~$
```

Figure 6 - Screenshot of successful Runner command in your Docker host terminal

5. **Verify the Runner registration:**

- Return to the Runners page in the Portal and refresh it.
- Your new Runner should appear in the list.

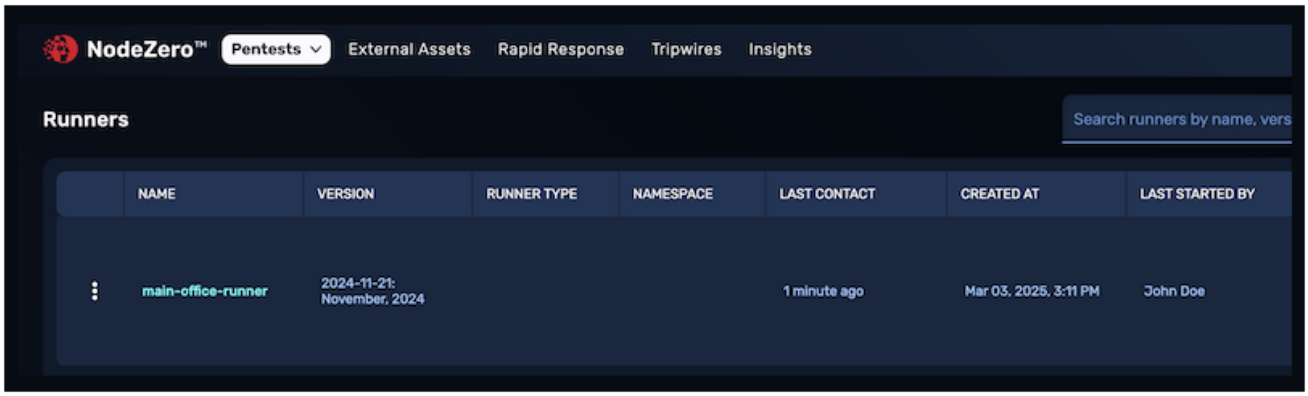


Figure 7 - Screenshot of NodeZero Runner listed in the Portal

Behind the scenes

Here’s what happens in the background when setting up the Runner:

- An API key with **NodeZero Runner** permissions is created, restricting it to polling for pentests and running the launch script.
- The `h3-cli` tool is installed in a new `h3-cli` directory on your NodeZero Host.
- The Runner starts via the `h3 start-runner` command.
- On Linux systems with `systemd`, the script registers the Runner as a service for automatic restarts after reboots.

How to use after setup

Once your NodeZero Runner is operational, you can:

1. **Assign pentests:**

- In the Portal, when provisioning a new Internal Pentest, select your Runner to immediately start the pentest remotely—without needing to SSH into the NodeZero host or manually run a command in the terminal.
- Optionally, save your pentest as a Template to easily apply it when setting up a schedule for automated, recurring runs.

  For step-by-step instructions on setting up internal pentests, see Run an Internal Pentest.

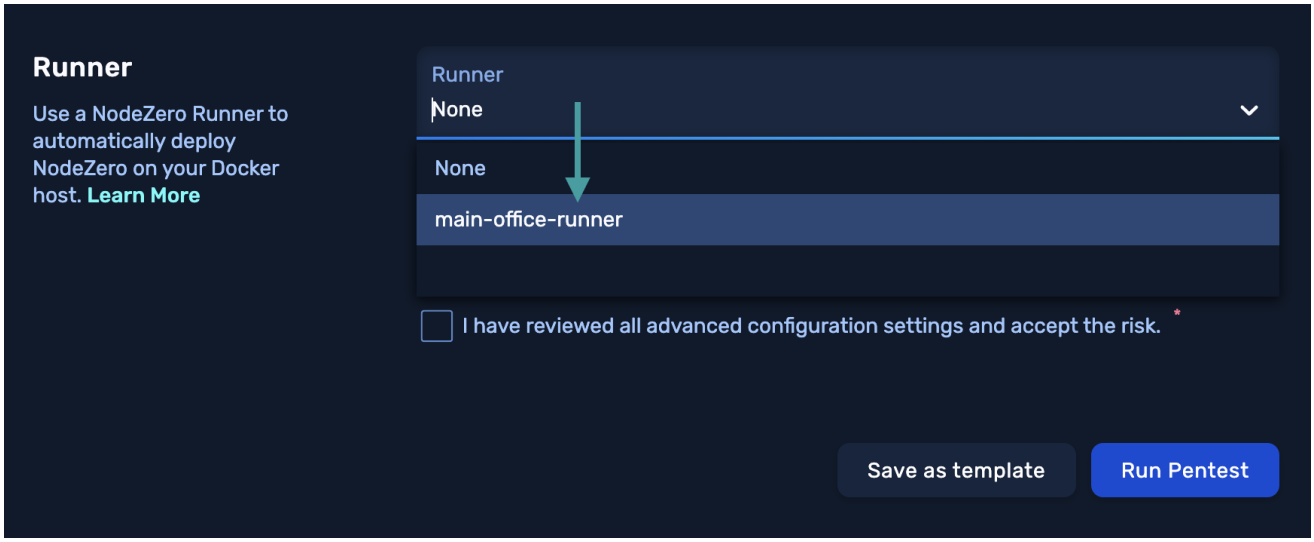


Figure 8 - Screenshot of selecting a Runner (and optionally as a template) for an internal pentest

2. **Schedule recurring tests:**

- Use the Portal's scheduling feature to automatically run pentest templates with a provisioned Runner at scheduled intervals.

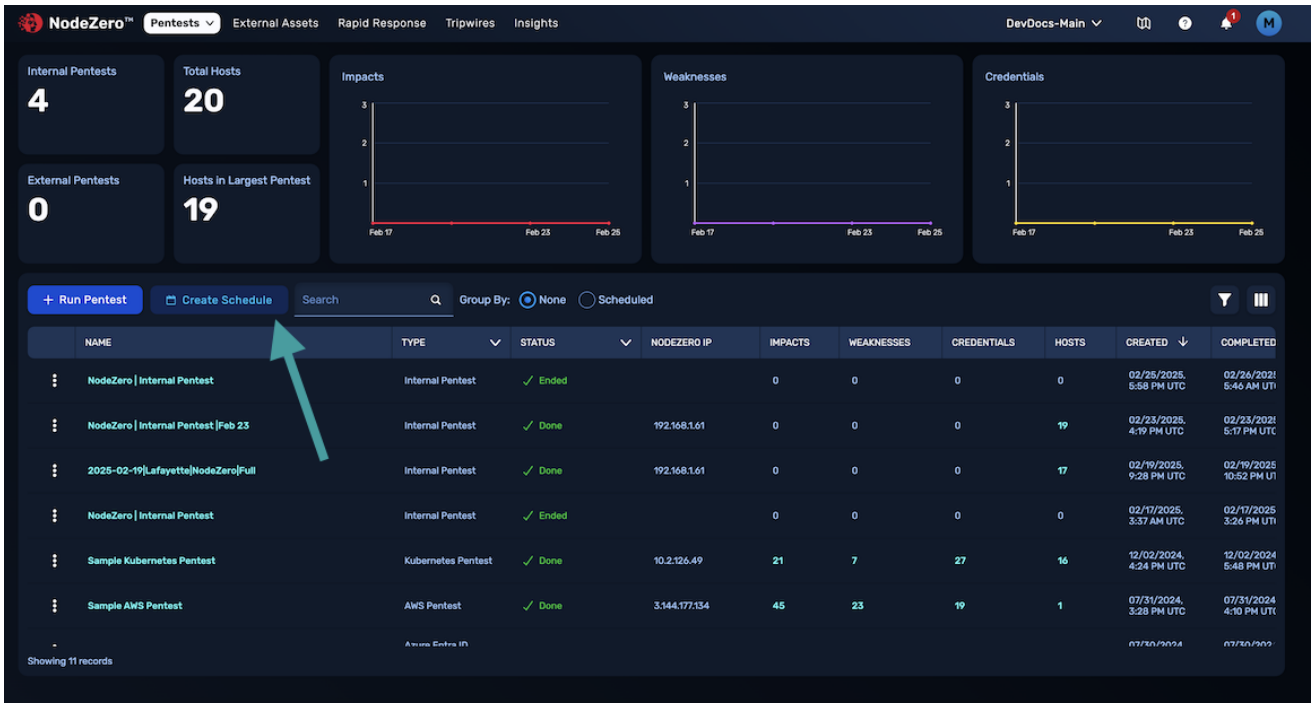


Figure 9 - Screenshot of Create Schedule button for pentests

- Select a Pentest Template with a provisioned Runner to schedule automated, recurring runs.

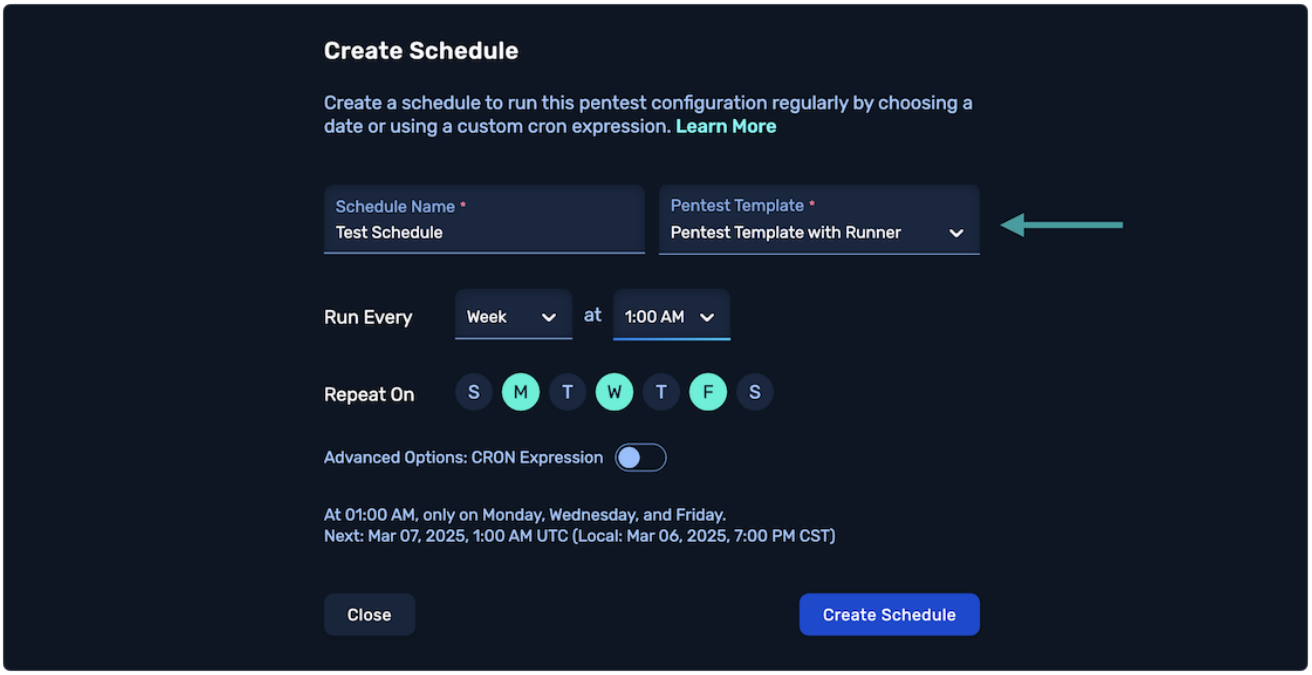


Figure 10 - Screenshot of schedule settings for an internal pentest

3. **Monitor activity:**

- View real-time progress and results in the Portal.

Common issues

Explore solutions to frequent NodeZero Runner problems to quickly get back to testing. For a comprehensive list of potential issues, see Troubleshooting your NodeZero Runner.

Managing your Runner

Learn how to effectively control and maintain your NodeZero Runner with Portal controls, essential CLI commands, and best practices. See Managing Runners and Tests and Commands to Manage a NodeZero Runner for details.

Conclusion

Congratulations! You've successfully set up a NodeZero Runner using the **Easy Install Script**. This automation will streamline your internal pentesting process, saving time and ensuring consistent security assessments. As your needs grow, consider deploying additional Runners across different network segments for comprehensive coverage.

NodeZero Runner

Portal home

Manual via h3-cli

NODEZERO RUNNER WITH H3-CLI

This guide provides a detailed walkthrough for setting up a NodeZero Runner on your NodeZero Host using the h3-cli tool. This method offers greater transparency and control, allowing you to customize the setup process for your specific environment. By the end, you'll have a fully operational Runner ready to automate internal pentests through the NodeZero Portal.

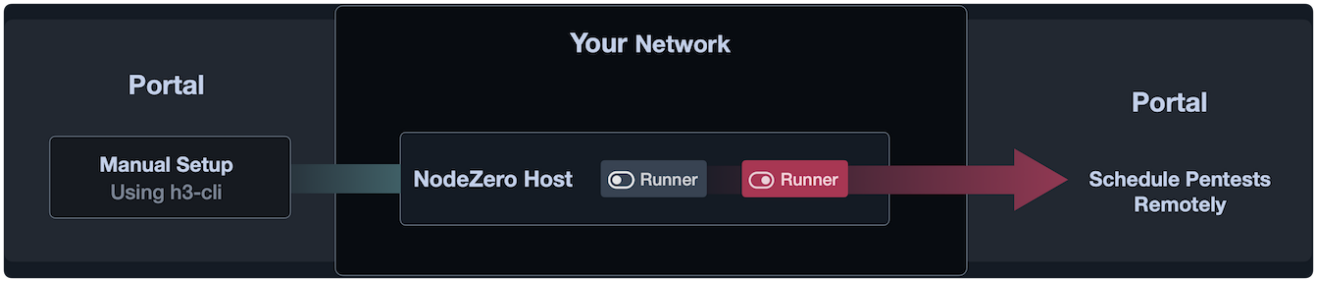


Figure 1 - Manual setup process using h3-cli, starting the Runner, and scheduling pentests.

When to use

Choose the Manual Setup if:

- You prefer hands-on control over each setup step.
- You need to customize the Runner configuration or troubleshoot deeply.

For a quicker setup, consider the Easy Install Script method.

NodeZero Runners are for internal pentests only. External pentests are deployed automatically in the Horizon3.ai cloud.

Requirements

Before starting, ensure you have:

- A NodeZero Portal account with access to one of the following:
- Account Settings (or EU Account Settings)
- A NodeZero host in your private network with Docker or Podman installed. See Quickstart Guide
- SSH or terminal access to the NodeZero host (with Git, `bash`, and Docker or Podman privileges).
- Familiarity with command-line operations.

Step-by-step instructions

Follow these steps to manually set up your NodeZero Runner:

Create an API key for the Runner

The NodeZero Runner uses the `h3-cli` to communicate with the Horizon3.ai API, requiring an API key with specific permissions.

- **Navigate to API key settings:**
- In the Portal, click the **Profile Menu** (upper right corner), then click **Settings**
- Click **My Settings** to display the **API Keys** section.

Figure 2 - My Settings page with the API Keys section visible.

- **Generate a key:**
- Create a new API key and set its permission level to **NodeZero Runner**.
- This role restricts access.
- It can only poll for assigned pentests and run the NodeZero Launch Script
- It **cannot** read existing pentest data or provision new tests.
- Save the API key securely. You will need it in the next step.

Install h3-cli on your NodeZero host

The h3-cli tool is required to start and manage the Runner process. Install it using the following commands:

- **Download and install h3-cli:** Run these commands in a terminal on your NodeZero host:

tar.gz **zip**

```
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.tar.gz -o h3-cli.tar.gz
mkdir -p h3-cli && tar -xzf h3-cli.tar.gz -C h3-cli
cd h3-cli
bash install.sh {your-api-key-here}
export H3_CLI_HOME=`pwd`
export PATH="$H3_CLI_HOME/bin:$PATH"
```

```
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.zip -o h3-cli.zip
unzip h3-cli.zip -d h3-cli
cd h3-cli
bash install.sh {your-api-key-here}
export H3_CLI_HOME=`pwd`
export PATH="$H3_CLI_HOME/bin:$PATH"
```

Replace `{your-api-key-here}` with the API key from Step 1.

What this does:

- Downloads and extracts the h3-cli package.
- Installs h3-cli using the provided API key.
- Adds h3-cli to your system's `$PATH`.

Verify installation:

Confirm h3-cli is working and using your API key:

```
h3 whoami
```

You should see details associated with your API key. If not, double-check the key and retry.

Start the NodeZero Runner

Use h3-cli to launch the Runner process on your NodeZero host:

Run the Runner:

- In this example we will name the Runner `main-office-runner`.
- Execute the following command, replacing `main-office-runner` with a unique name for your Runner:

```
h3 start-runner main-office-runner /tmp/main-office-runner.log
```

The Runner runs in the background, logging to `/tmp/main-office-runner.log`.

Naming the Runner:

- Treat Runner names as unique identifiers. Do not reuse names across your account.
- Avoid reusing names across multiple Runners in your account to prevent conflicts.

The process persists after you close the terminal session.

Verify the Runner connection: Confirm the Runner is registered with the Horizon3.ai API:

```
h3 runners
```

Look for an entry for `main-office-runner` with a recent `last_heartbeat_at` timestamp.

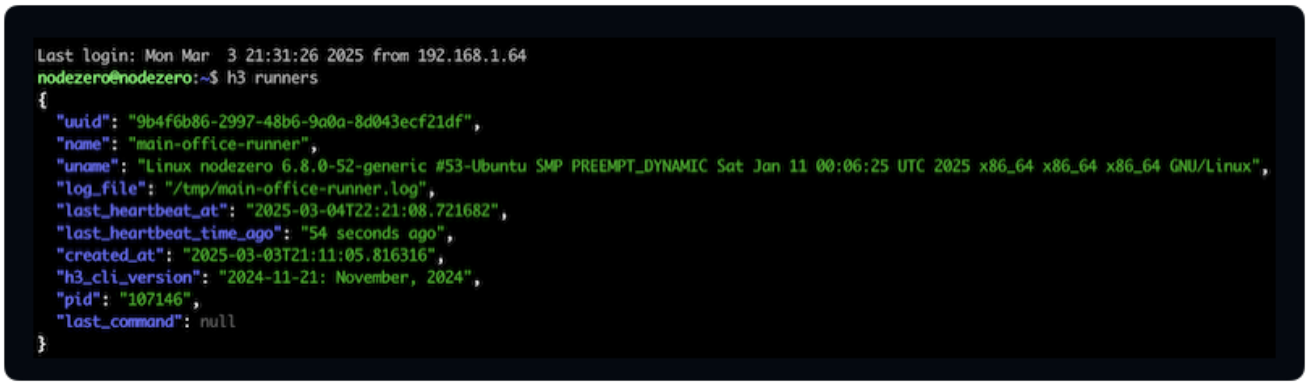


Figure 3 - Output of `h3 runners` command.

How to use after setup

Once your NodeZero Runner is operational, you can:

1. **Assign pentests:**

- In the Portal, when provisioning a new Internal Pentest, select your Runner to immediately start the pentest remotely—without needing to SSH into the NodeZero host or manually run a command in the terminal.
- Optionally, save your pentest as a Template to easily apply it when setting up a schedule for automated, recurring runs.

  For step-by-step instructions on setting up internal pentests, see Run an Internal Pentest.

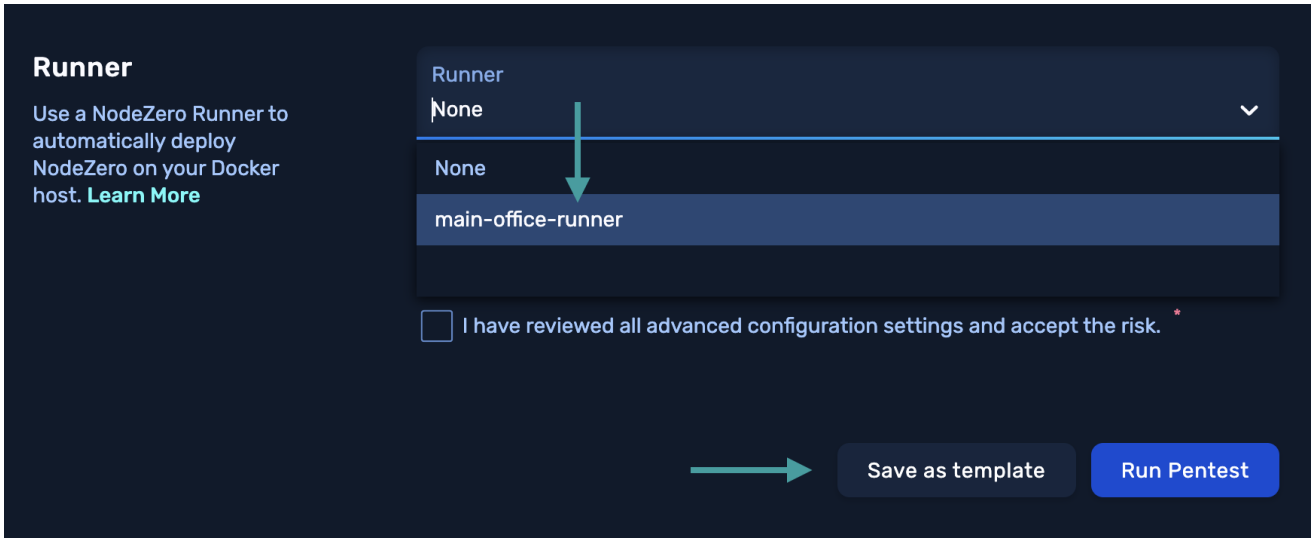


Figure 4 - Screenshot of selecting a Runner (and optionally as a template) for an internal pentest

2. **Schedule recurring tests:**

- Use the Portal's scheduling feature to automatically run pentest templates with a provisioned Runner at scheduled intervals.

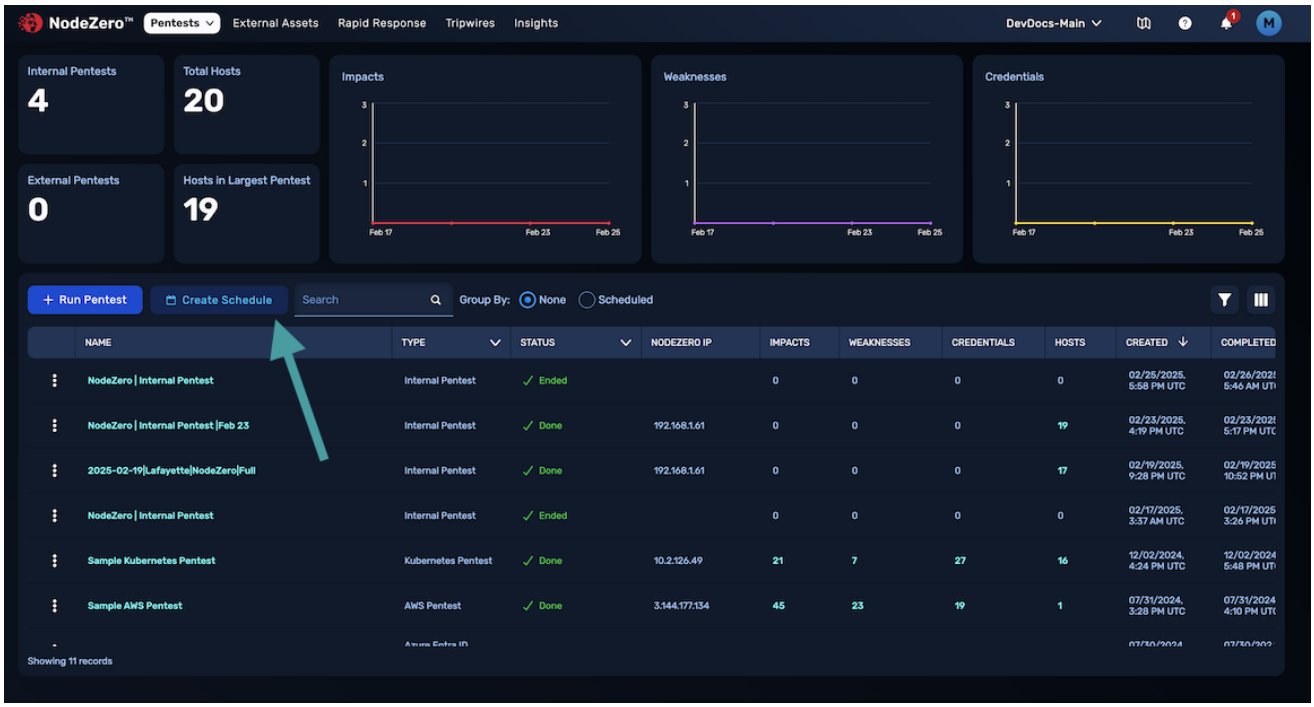


Figure 5 - Screenshot of Create Schedule button for pentests

- Select a Pentest Template with a provisioned Runner to schedule automated, recurring runs.

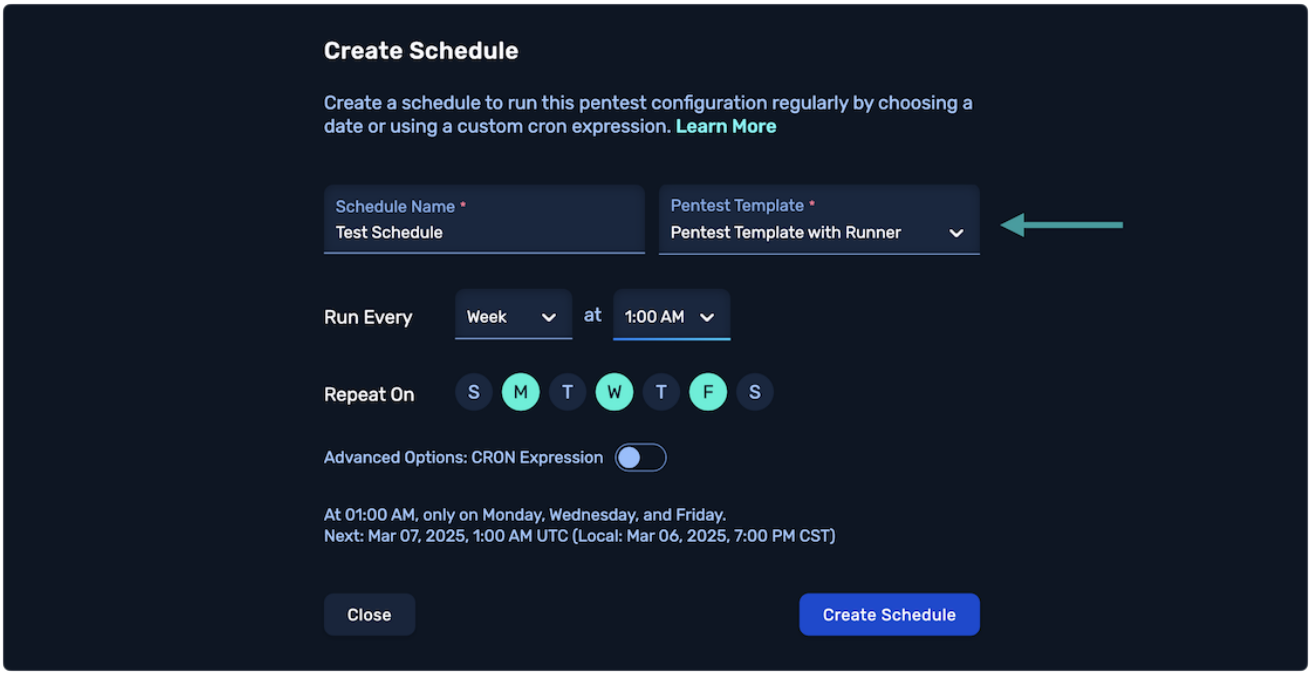


Figure 6 - Screenshot of schedule settings for an internal pentest

3. **Monitor activity:**

- View real-time progress and results in the Portal.

Common issues

Explore solutions to frequent NodeZero Runner problems to quickly get back to testing. For a comprehensive list of potential issues, see Troubleshooting your NodeZero Runner.

Managing your Runner

Learn how to effectively control and maintain your NodeZero Runner with Portal controls, essential CLI commands, and best practices. See Managing Runners and Tests and Commands to Manage a NodeZero Runner for details.

Conclusion

Congratulations! You've set up a NodeZero Runner **manually using h3-cli**, giving you full control over the process. Your Runner is now ready to automate internal pentests, schedule recurring tests, and help secure your network. For broader coverage, consider adding more Runners in other network segments.

NodeZero Runner

Portal home

3.5.3 Upgrading

Upgrading the Runner

Keeping your NodeZero Runner current ensures that it stays compatible with the latest NodeZero platform, benefits from ongoing reliability and security fixes, and continues to launch tests reliably.

An upgraded Runner benefits from ongoing improvements to job scheduling, Docker deployment, and API communication. Upgrades are required to use newer automation features, such as touchless deployment and auto-injected credentials. Also, upgrades simplify troubleshooting, and prevent unexpected breakage as the platform evolves.

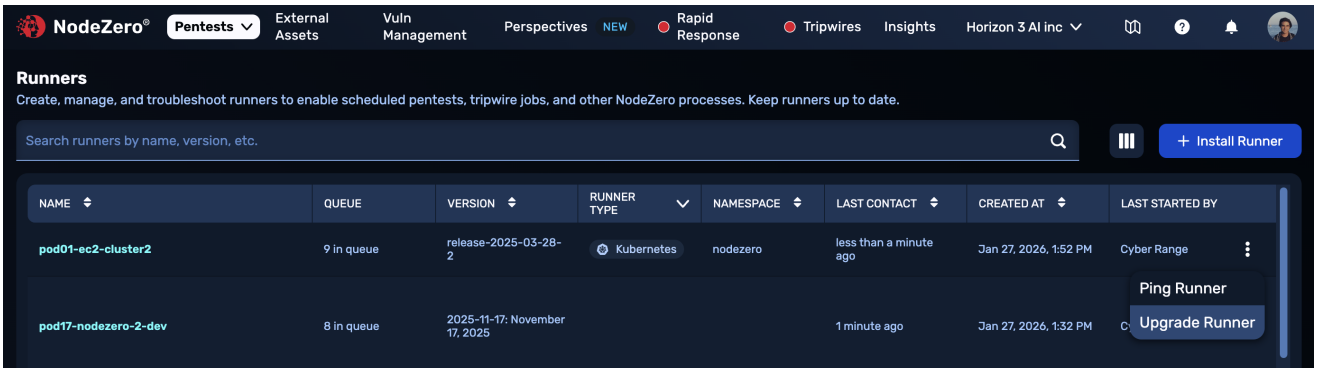
> **Minimum Runner version for Queues**
>
> In particular, queue and priority management features are available only with Runners on version `2026-03-04` or later.

Upgrade your installation using one of the methods below.

UPGRADE RUNNER FROM PORTAL

You can upgrade a Runner easily from the NodeZero Portal's UI. On the **Pentests** > **Runners** page, open the Actions (⋮) menu to the right of any Runner.

Here, a single click on the **Upgrade** option enables you to update the Runner without redeploying your host.



Upgrading Runner from NodeZero Portal UI

Behind the scenes, this option:

- Runs the CLI upgrade flow (that is, runs `h3 upgrade` on the host), to download and install the latest supported version of `h3-cli`.
- Restarts the Runner process to use the new CLI version.
- Reports in to NodeZero as upgraded.
- Does not change existing pentest data, configurations, templates, or credentials in the NodeZero Portal. The update applies only to the CLI bits and Runner service on the selected host.

Verify an upgrade

After allowing a short delay for the upgrade to complete, refresh the Runners page and open the selected Runner's details.

Check the **Command History** to confirm that the upgrade completed successfully (exit status `0`). Check the **Last Contact** to confirm an up-to-date last heartbeat.

UPGRADE RUNNER VIA CLI

A CLI upgrade also upgrades the Runner. If you prefer to manage the Runner and its underlying CLI directly on the host, you can instead log in and run `h3 upgrade` – or follow any of the other options listed at Upgrade the CLI. You can incorporate the CLI and script options into into your own automation (for example, change-management pipelines).

WHEN TO UPGRADE A RUNNER

Beyond general guidance to keep your Runners current, you should especially upgrade Runners when:

- You want to make sure a Runner is on the latest CLI before running or scheduling tests.
- You see guidance in release notes, or from Horizon3 Support, that a Runner upgrade is required to incorporate a new feature or fix.
- You want to standardize multiple Runners on the same CLI version.

LIMITATIONS AND TROUBLESHOOTING

Keep in mind these considerations around upgrading a Runner, especially through the NodeZero Portal UI:

- **Minimum CLI version**: Very old versions of `h3-cli` do not support the Portal's **Upgrade** option. You'll need to upgrade the CLI version from the host's command line before you can run upgrades from the UI.
- **Runner availability during upgrade**: While the Runner is upgrading and restarting, it cannot launch new internal pentests. Any pentests assigned to that Runner will wait until the Runner is back online. This is typically a brief interval. But if that Runner is the only one in a critical environment, with scheduled time-sensitive or long-running tests, you might prefer to upgrade outside peak hours.
- **Host permissions and environment**: An upgrade might fail because the host environment or permissions prevent the CLI from downloading or writing files (for example, due to an invalid working directory or insufficient permissions). You'll see these failures reflected with error statuses in the Runner's recent commands and logs.
- **No "already latest" check in UI**: The NodeZero Portal will execute a requested **Upgrade** process without first checking whether the Runner is already using the latest CLI version. This is harmless, but just causes needless network traffic and Runner downtime.

3.5.4 Managing

Managing Runners and Tests

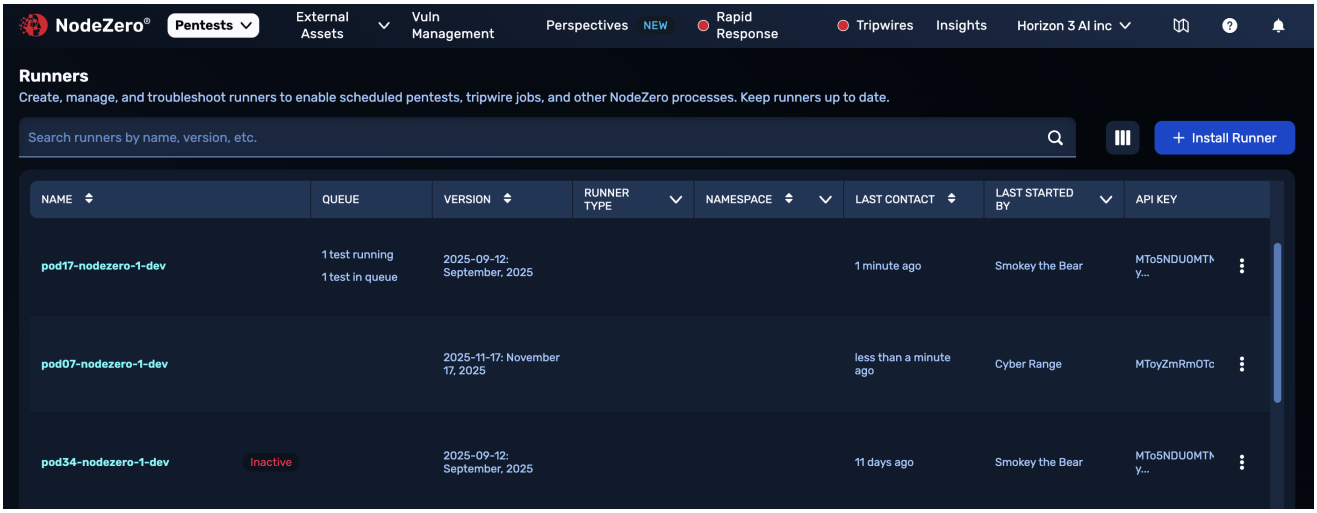
This page covers options in the NetZero Portal UI for:

- Monitoring the queue order and scheduling of pentests that are assigned to Runners.
- Changing the priority of pentests assigned to Runners.
- Changingg the priority of 1-Click Verify operations.
- Adding injected credentials to a Runner, along with other options available in the Runners UI.

> **Minimum Runner version for Queues**
>
> The queue and priority management features outlined in the next section are available only if you upgrade your Runners to the `2026-03-04` version or later.

MONITORING RUNNER QUEUES

Within the NodeZero Portal, the **Pentests** > **Runners** page's **Queue** column indicates any ops currently running or queued for each pentest. You'll see these indicators regardless of your license type.



Monitoring Queue status on Pentests > Runners page

PENTEST QUEUEING PRIORITY

NodeZero provides several ways to control the execution order of pentests assigned to Runners.

> **Not available on all license types**
>
> Flex licenses do not support the scheduling and active queueing controls outlined in this section.

- **Queue tests:** Stack multiple pentests (including One-Click Verity tests) onto a Runner, and let NodeZero work through them in order.
- **Skip the queue:** When you configure a new pentest or a One-Click Verity test, you can assign it priority to execute immediately.
- **Run Now:** Prioritize an already queued test to execute immediately.
- **Run Next:** Prioritize a queued test to execute right after the current test completes.

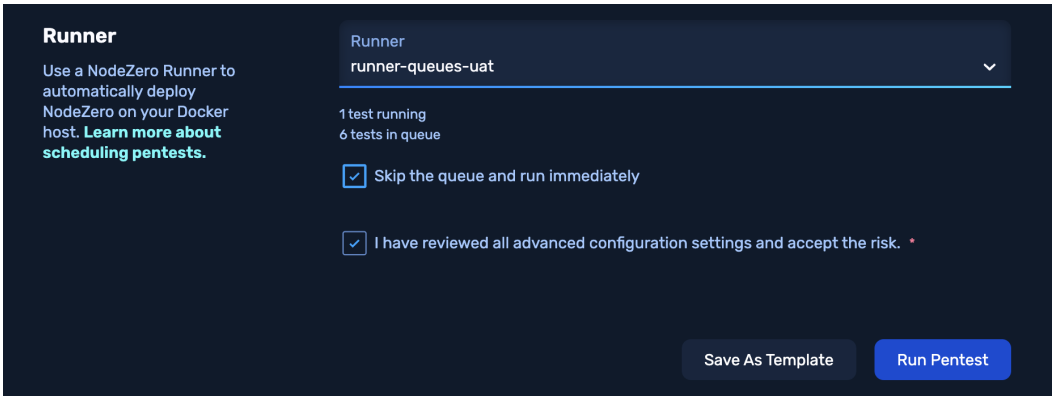
These options enable you to use a single Runner to manage multiple ops, while running high-priority ops as soon as you need to.

When you select **Skip the queue** or **Run Now**,, NodeZero will pause any test currently running, and will maintain its state. That op will resume in queue order, after the op you've accelerated.

New Test: Run Now or Queue

When you define a new pentest and assign it to a Runner, a **Skip the queue and run immediately** check box appears. Select this check box if you want to preempt any running or queued ops.

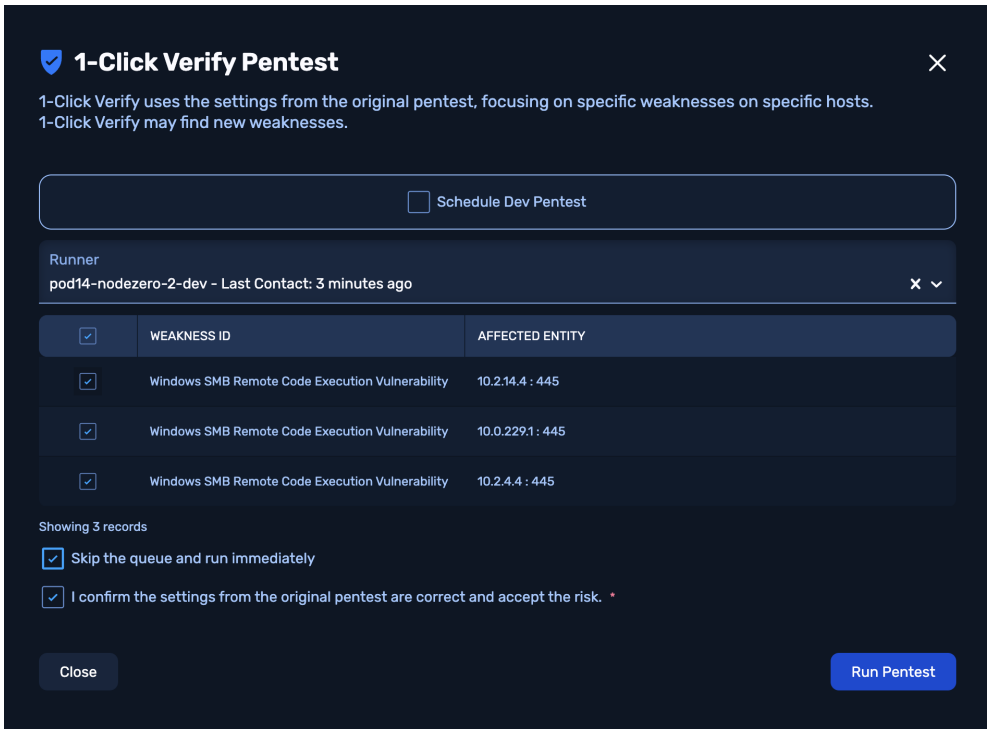
If you don't select it, and you proceed to **Run Pentest** while tests are currently running or queued, NodeZero will automatically add the new test to the bottom of the queue.



Specifying immediate execution of a new pentest

Verify Immediately

Like the pentest config, the 1-Click Verify modal offers a **Skip the queue and run immediately** check box, which works the same way.



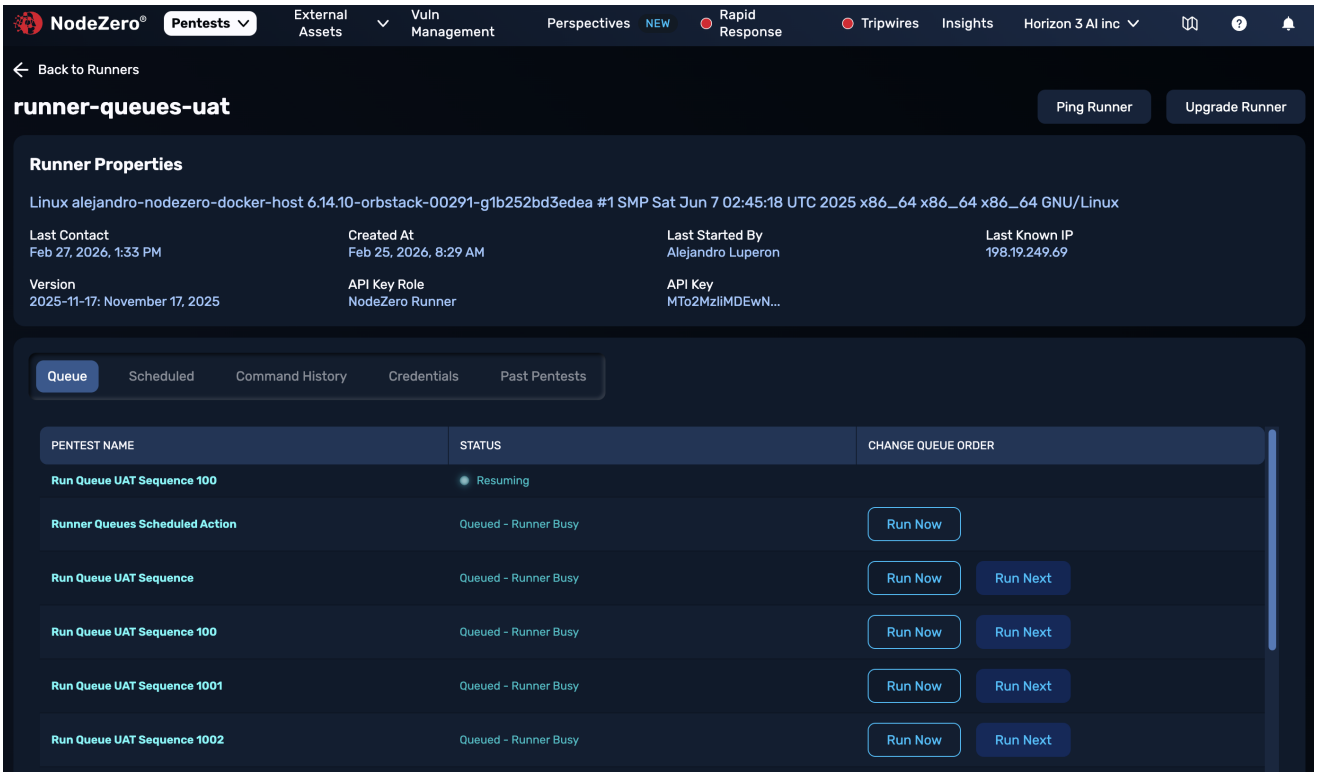
Specifying immediate execution of a 1-Click-Verify test

Reorder the Test Queue

On a Runner's details page, the **Queue** tab enables you to specify pending ops' priority. These controls appear in the **Change Queue Order** column, in specific conditions.

A **Run Now** button appears when at least one pentest is preparing, running, or queued. Click this button to run a specified op immediately – potentially pausing a pending or running op.

A **Run Next** button appears when at least one pentest is queued behind the currently running test. Click this button to move a specified op to the top of the queue, to execute after the current op.



A Runner queue with Run Now and Run Next options

Ensuring Host Capacity

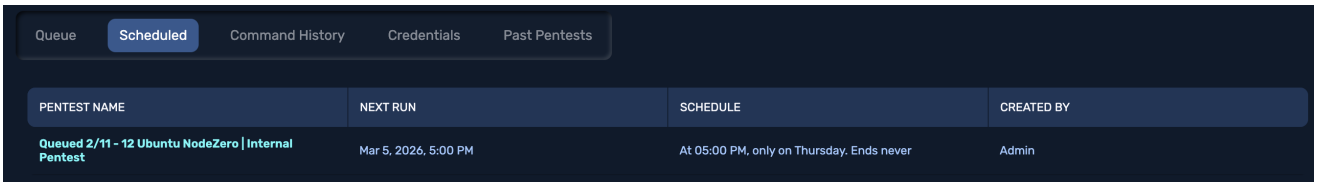
To support the pause-and-resume behavior described above, NodeZero creates a separate container to maintain each paused pentest's state. If you pause multiple tests on a single host, this might require provisioning disk space beyond minimal Host Requirements. Container footprints will vary as tests proceed and discover weaknesses.

RUNNER CONTROLS

Each Runner's details page provides the following monitoring and management tabs to the right of the **Queue** tab highlighted above.

Monitoring Schedules

The **Scheduled** tab shows schedules assigned to execute on this Runner.



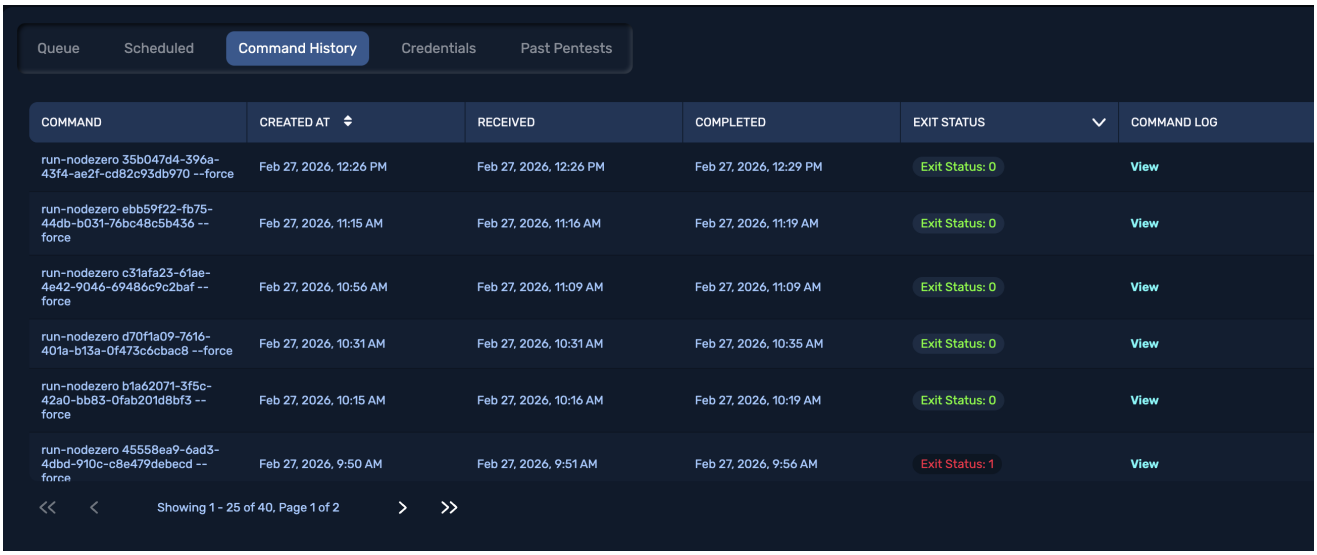
A Runner's Scheduled tab

> This tab does not appear with a Flex license, which does not support scheduling.

Command History

This tab lists recent commands sent to this Runner, with columns displaying the:

- Command name.
- Received and Completed timestamps (UTC).
- Exit status.
- Log (click a **View** link to display the `stdout` / `stderr` for the corresponding command).



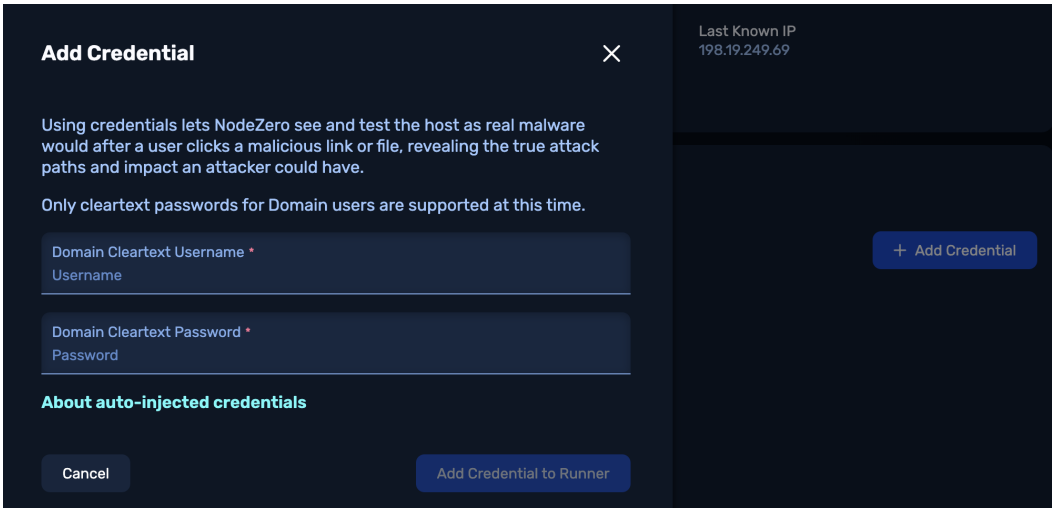
A Runner's Command History tab

Injecting Runner Credentials

On a Runner's **Credentials** tab, you can click **+ Add Credential** to automatically inject a credential into the Runner, as you would do on a scheduled test.

Injecting credentials emulates the impact that an attacker might have if using credentials that they've compromised. By injecting a credential at the Runner level, you can auto-inject the credential into this Runner's tests without having to apply it to specific pentests.

Selecting this option might prompt you to first upgrade the Runner.

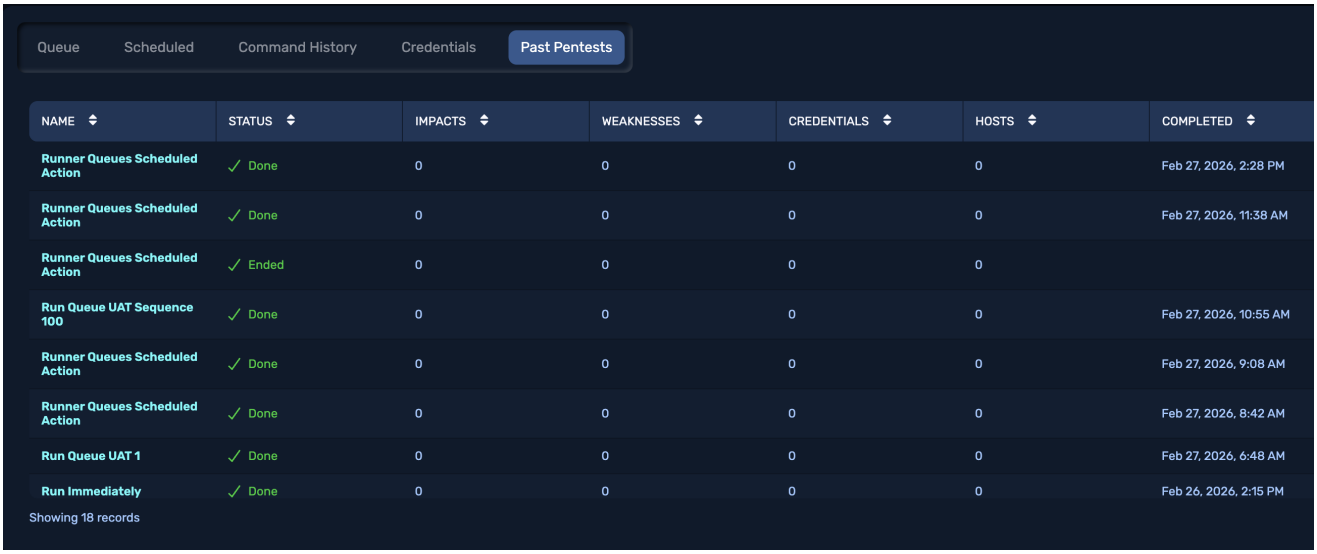


Adding a credential to a Runner

Past Pentests

This tab lists pentests that have completed on this Runner, with columns displaying the:

- Op name.
- Status (typically **Done** or **Ended**).
- Number of discovered Impacts and Weaknesses.
- Number of Credentials and Hosts involved.
- Completion time (UTC).



A Runner's Past Pentests tab

3.5.5 Commands

Commands to Manage a NodeZero® Runner

This page provides essential commands and best practices for controlling your Runner, from checking its status and viewing logs to configuring auto-restarts and handling advanced features like credential injection. Whether you're scheduling recurring tests or troubleshooting, these tools help you maintain consistent security testing across your network.

> **UI queue management options**
>
> To monitor and manage Runner pentest queues via the NetZero Portal UI, see Managing Runners and Tests.

RUNNER SERVICE COMMANDS USING SYSTEMD

These commands use `systemd` to manage the NodeZero Runner as a service on your NodeZero Host:

Check status:

```
systemctl status nodezero-runner-my-nodezero-runner
```

Stop the service:

```
sudo systemctl stop nodezero-runner-my-nodezero-runner
```

> **The queues abide**
>
> If you stop a Runner using this command, any Runner queue will resume in the same order when the Runner is available again.

Disable auto-start:

```
sudo systemctl disable nodezero-runner-my-nodezero-runner
```

Re-enable and start:

```
sudo systemctl enable nodezero-runner-my-nodezero-runner
sudo systemctl start nodezero-runner-my-nodezero-runner
```

RUNNER MANAGEMENT COMMANDS USING H3-CLI

These commands leverage the `h3-cli` tool to manage your NodeZero Runner directly:

View logs:

Check the Runner log for activity or errors:

```
h3 tail-runner my-nodezero-runner
```

Stop the Runner:

Terminate the Runner process:

```
h3 stop-runner my-nodezero-runner
```

> **The queues abide**
>
> You manage a Runner's queue ordering and preemption not via CLI commands, but in the NetZero Portal. If you stop a Runner using this command, the queue will resume in the same order when the Runner is available again.

Delete the Runner:

Remove the Runner registration from Horizon3.ai. If the Runner is still running, it will re-register on its next heartbeat.

```
h3 delete-runner my-nodezero-runner
```

Auto-start Runner at system startup

To ensure your Runner restarts automatically after a system reboot:

- **Register with systemd:** If your NodeZero host uses `systemd`, use the following command:

```
h3 start-runner-service my-nodezero-runner /tmp/my-nodezero-runner.log
```

- This requires `sudo` for `systemd` operations.
- The Runner will be registered as `nodezero-runner-my-nodezero-runner`.
- **Benefits:**
- Starts automatically at boot.
- Restarts if the process fails.

OTHER TASKS:

Renaming a Runner

- Stop (and optionally delete) the Runner.
- Start a new Runner with the desired name.
- Update any templates with the new name.

Auto-Inject credentials

Use the Runner to auto-inject credentials into pentests for enhanced testing.

NodeZero Runner

Portal home

3.5.6 Common Issues

Troubleshooting Your NodeZero Runner

COMMON ISSUES AND FIXES

Encountering issues with your NodeZero Runner can disrupt your internal pentesting workflow. This page provides step-by-step solutions to common problems, such as connectivity failures, Docker permission errors, and systemd configuration issues. Whether your Runner isn't connecting, failing deployments, or encountering log errors, these troubleshooting steps will help you restore functionality and keep your scheduled pentests running smoothly.

Runner not connecting:

Diagnose and resolve issues when your Runner fails to connect to the Horizon3.ai API.

Check the last heartbeat:

```
h3 runners
```

If `last_heartbeat_at` is over 60 seconds old, the Runner may have stopped or lost connectivity.

Verify the process is running:

```
h3 ps-runner
```

Inspect logs for errors:

```
h3 tail-runner my-nodezero-runner
```

Recent commands:

Review the most recent commands executed by your Runner to identify potential errors or failures.

View the last 5 commands executed by the Runner:

```
h3 runner-commands my-nodezero-runner
```

Look for non-zero exit statuses or error messages.

Docker permission errors:

Address permission issues that prevent the Runner from accessing Docker on your NodeZero Host.

If you see an error like:

```
[!] FAILED: Failed to validate Docker. Verify this account has permissions to run Docker and retry.
```

Run the Runner with `sudo`:

```
sudo h3 start-runner my-nodezero-runner /tmp/my-nodezero-runner.log
```

Or add the user to the `docker` group (e.g., for user `ubuntu`):

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
```

Log out and back in to apply group changes.

Retry a failed deployment:

Retry a NodeZero deployment that failed due to temporary issues with your Runner.

Queue a new NodeZero deployment to retry a failed attempt:

```
h3 run-nodezero-on-runner {op_id} my-nodezero-runner
```

Find the `op_id` of the latest pentest:

```
h3 pentest
```

For further assistance, contact support@horizon3.ai.

Troubleshoot systemd:

Resolve issues related to the systemd service configuration for auto-starting your Runner.

209/STDOUT Error: If `systemd` can't write to the log file (e.g., `/tmp/my-nodezero-runner.log`), adjust permissions:

```
chmod 666 /tmp/my-nodezero-runner.log
```

Or delete it to let `systemd` recreate it.

Resources:

- How to Use systemctl
- systemctl Man Page
- systemd Wiki

For systems not using `systemd`, contact support@horizon3.ai for assistance.

NodeZero Runner

Portal home

3.6 Deployment Strategy

3.6.1 Deployment Strategy

The deployment of your NodeZero Host depends on the specific insights you aim to gain from your pentest. Below are some options to help you choose the best setup for your goals.

Placement and Intent

The **placement** of the NodeZero Host, and its defined **intent** determine the scope and behavior of the assessment.

- **Placement**: Defines where the NodeZero Host is deployed within the network.
- **Intent**: Guides the scope and actions of the pentest based on the desired objectives.

Placement

Select the appropriate NodeZero Host placement based on your testing goals:

Scope refers to the network boundaries or specific targets (e.g., IP ranges, endpoints, or domains) that NodeZero will assess during the pentest, as determined by the host's placement.

Placement	Type	Description
1. Inside the Scope	Custom Scope	Position the host within the target network segment to simulate an internal attacker.
2. Outside the Scope	Custom Scope	Place the host outside the target network, mimicking an external attacker.
3. Outside @ Endpoints (i.e., /32s)	Custom Scope	Deploy at specific endpoints (e.g., single IPs) outside the target scope.
4. Attack Starting Point	Intelligent Scope	Use as a defined entry point to simulate a targeted attack vector.
5. Full Private Scope	RFC1918	Cover the entire private network accessible from the host’s location.
6. OSINT Focused	OSINT	Launch externally to assess cloud-facing assets.

Intent

Select an intent to guide the scope and actions of the NodeZero assessment:

Placement	Intent	Will Enumerate and Exploit	Won't Execute	Use Cases
1. Inside the Scope	Limit the attacker's scope to a defined range from within the network.	- In-scope hosts, services, domains, web, credentials, data resources - ProTip: Ensure a DC is "in scope"	- On anything outside the prescribed scope	- Internal Pentest - SOC SLAs - Verify Policies - Verify EDM/ SIEM
2. Outside the Scope	Limit the attacker to assess accessible hosts and data from an external perspective.	- In-scope hosts, services, web, credentials (except MITM or PTH attacks)	- Man-in-The-Middle attacks	- External Pentest - Verify Segmentation - Verify access to a sensitive VLAN - Third-party security assessment
3. Outside @ Endpoints (i.e., /32s)	Focus on specific hosts to check for endpoint vulnerabilities and misconfigurations.	- Specified hosts, ports, services, web, exploitable vulnerabilities	- On infrastructure not chained nor vulnerable	- Test EDR endpoint vulnerabilities
4. Attack Starting Point	Discover what an attacker can access from a specific starting point.	- Discovered hosts, services, domains, web, credentials, data resources	- On inaccessible hosts, services, domains, web, credentials	- Internal Pentest - Segmentation - Verify Policies - Verify EDR/ SIEM - Test Blast Radius - Test ZeroTrust
5. Full Private Scope	Search every nook and cranny of the private IP space accessible in the environment.	- Discovered hosts, services, domains, web, credentials, data resources	- On inaccessible hosts, services, domains, web, credentials	- Internal Pentest - Asset Discovery - Assess hybrid env - Verify Policies - Verify EDR/ SIEM
6. OSINT Focused	Assess publicly available data vulnerable to an external attacker.	- Publicly available usernames, subdomains (from TLDs), web-facing attack surface	- On internal assets - *NOTE: Combined with an internal op/ access to a DC will verify user/ password access	- Public-facing Reconnaissance - Company recon - User recon - Subdomain recon - *Cred stuffing

Notes

- **OSINT Focused:** This external deployment focuses on cloud-facing assets and can be combined with internal operations for comprehensive user/password validation.
- **Execution Limits:** NodeZero respects defined scopes and avoids actions on out-of-scope or inaccessible assets unless explicitly chained or vulnerable.

Tip

Select the placement and intent that align with your security goals. Consult Horizon3.ai support for complex scenarios.

Placement – More Info

This section offers further guidance on placing and scoping your test.

1. INSIDE THE SCOPE

To limit the scope and assess what an attacker could exploit within a defined range, place the NodeZero Host inside the desired test scope. When configuring the scope for your pentest, ensure the NodeZero Host is within one of the specified CIDR ranges for the test.

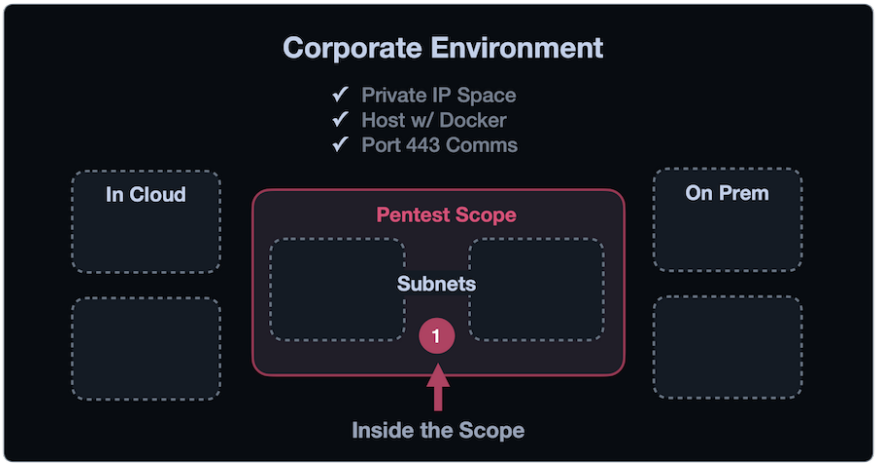


Figure 1 - Deployment inside your test scope

Tip

Ensure that a Domain Controller is in-scope as well, and NodeZero will attempt to exploit any vulnerabilities or misconfigurations on this host, as well as verifying weak domain defaults and credentials.

Use cases for Inside the Scope

This placement method facilitates rapid assessments, allowing for a streamlined Find-Fix-Verify loop that enhances your ability to quickly improve and adapt your security posture.

2. OUTSIDE THE SCOPE

To gain an “outside-in” perspective and evaluate whether an attacker could access critical data and assets within a specific scope, place the NodeZero Host outside the scope you want to test. When configuring the pentest scope, ensure that the NodeZero Host is **not** within the specified CIDR range(s) for the test.

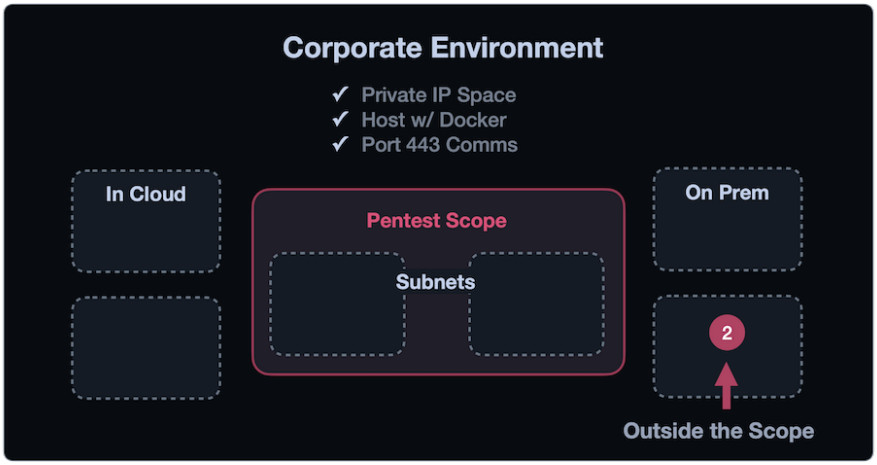


Figure 2 - Deployment outside of your test scope

> **Outside limitations**
>
> When NodeZero is **not** in the same IP range as the scope, it will not execute Man-In-The-Middle or pass-the-hash attacks.

Use cases for Outside the Scope

This approach offers an unrestricted assessment, providing a clear view of what is accessible, valuable, and vulnerable from an external starting point.

3. ENDPOINTS-ONLY SCOPE

To quickly verify whether the vulnerability you just remediated had the desired effect, select a single host or a range of hosts using /32 CIDR blocks. When configuring the pentest scope, ensure the NodeZero Host has access to the specific host(s) identified by the /32 CIDR range(s), but is NOT within the specified CIDR range(s) for the test.

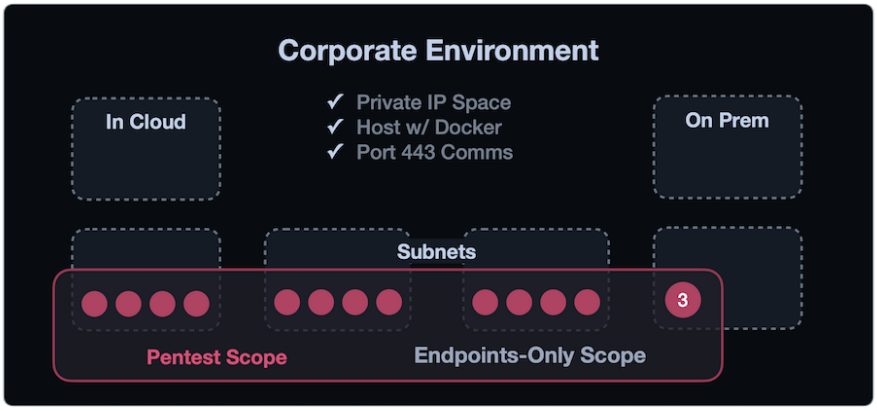


Figure 3 - Deployment with Endpoints-only scope

> ⚠ **Endpoints-only limitations**
>
> When NodeZero is **not** in the same IP range as the scope, it will not execute Man-In-The-Middle or pass-the-hash attacks. Further, with this restricted scope, NodeZero will chain neither weaknesses nor paths, because you have limited the scope to a specific endpoint for this assessment.

Use cases for Endpoints-only scope

This is a focused assessment designed for quick validation, allowing you to verify that your fix has been implemented successfully and that the vulnerability now poses a lower risk to your attack surface.

4. INTELLIGENT SCOPE

To simulate a true “black box” pentest and see what a non-credentialed attacker could enumerate and exploit from a specific starting point in your network, use Intelligent Scope.

In the Scope section of your Op configuration, select the “Intelligent Scope” option and leave the “Include” box blank. NodeZero will begin with the /16 subnet of the host IP it was deployed on, then expand organically across the infrastructure as it discovers additional hosts and subnets. This mimics how an attacker would explore the environment.

The more vulnerabilities, weaknesses, and misconfigurations are exploitable, the greater the visibility and testing coverage NodeZero will provide. This process demonstrates how an attacker could navigate the environment by chaining together tactics, techniques, and procedures (TTPs) based on their findings and exploiting the attack surface.

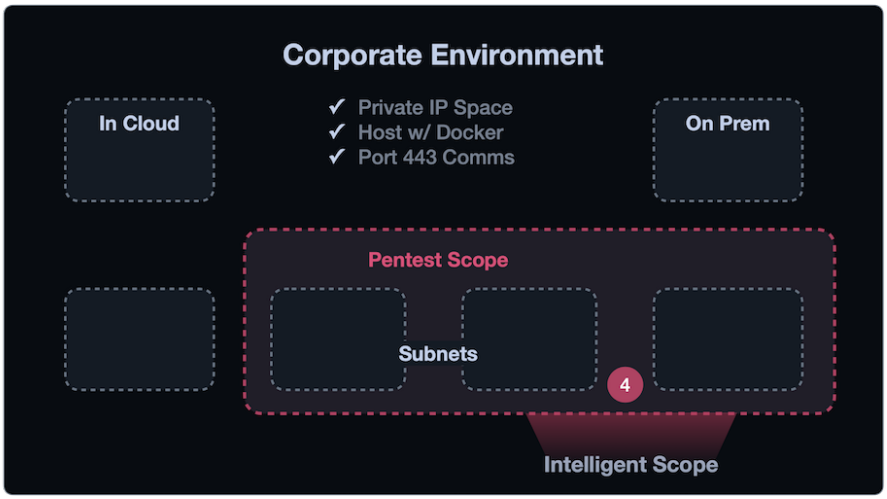
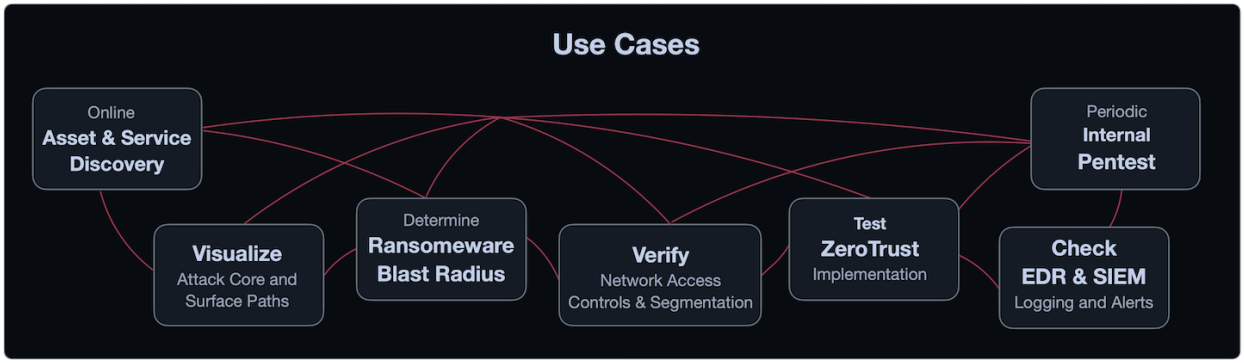


Figure 4 - Deployment with Intelligent scope

Use cases for Intelligent Scope

This is a proactive assessment that provides a deep understanding of what is accessible, valuable, and vulnerable from any entry point, helping you identify potential risks and attack paths across your environment.



5. ALL PRIVATE IP SCOPE (I.E., RFC 1918)

Use RFC 1918 to conduct a private scope pentest, quickly and safely enumerating everything accessible within your private network.

When configuring the scope for your pentest, simply select the **Use RFC 1918** option, and NodeZero will handle the rest. If there are specific IP addresses or ranges you wish to exclude from the assessment, add them to the **Exclude** box during scope configuration.

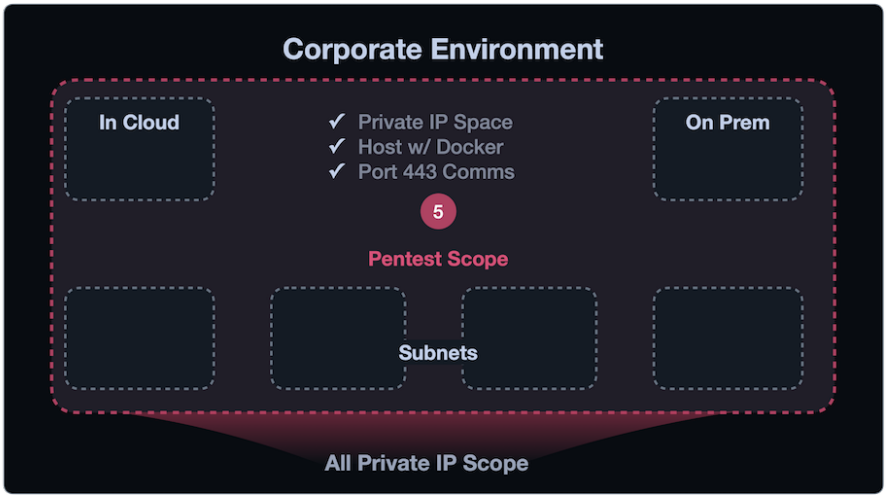


Figure 5 - Deployment with All private IP scope

> **Note**
>
> This operation might take longer, as NodeZero enumerates any accessible IPs and DNS names, including edge routers. If your routers are misconfigured for routing private IPs, NodeZero might also attempt to enumerate those external private IPs.

> **Tip**
>
> If you want NodeZero to discover every detail of your environment, place it in an unrestricted ACL. This will allow it to explore all accessible areas, uncovering every nook and cranny of your network.

Use cases for All private IP scope

This is your comprehensive and unrestricted enterprise assessment, designed to provide a complete view of your network. It should be run regularly to ensure continuous security monitoring and vulnerability management.

6. OSINT FOCUSED

Available with any of the pentest types, our Open-Source Intelligence (OSINT) assessment allows NodeZero to gather publicly available information and incorporate it into the pentest. During the second step of configuring your pentest, you can gain a true external perspective; your company name will be auto-filled, and you'll have the option to provide TLDs and weak password terms for NodeZero to test using any discovered information.

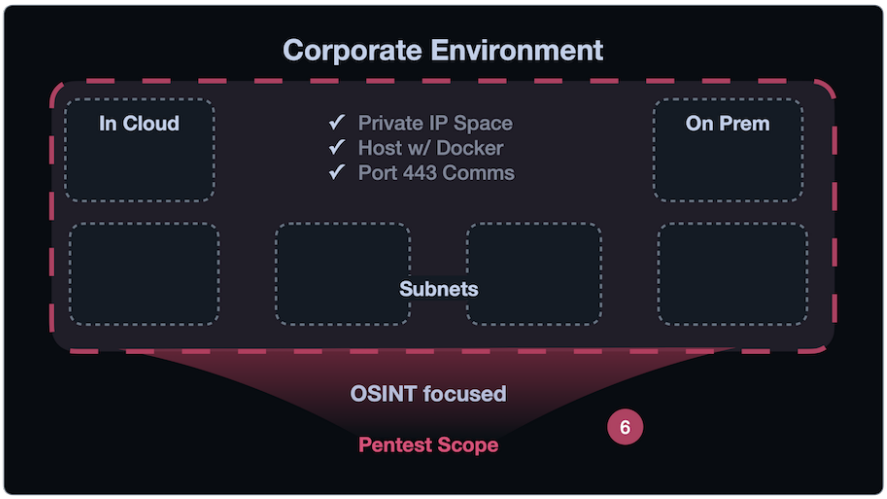


Figure 6 - Deployment with OSINT focused

OSINT + internal

NodeZero's OSINT gathering operates outside your environment, so NodeZero placement isn't as critical. However, when combined with an internal pentest that includes a domain controller in-scope, NodeZero will validate domain users and passwords against those found publicly. This will provide you with deeper insight into your attack surface and potential risks.

Use cases for OSINT focused

This is your external reconnaissance capability, allowing you to see what attackers can discover and leverage to initiate their campaigns and establish a foothold in your environment.

NEXT STEP

Set Up NodeZero Host

Deploy a NodeZero host inside your network to perform internal security assessments.

4. Tripwires

4.1 NodeZero Tripwires

NodeZero Tripwires is an add-on feature.

NodeZero Tripwires integrates seamlessly with NodeZero autonomous pentesting, adding an extra layer of early threat detection to your security stack. These "tripwires" act as decoys, appearing as legitimate files or credentials within your environment. During a pentest, NodeZero automatically places these tripwires along critical attack paths based on identified vulnerabilities. If a malicious actor triggers one of these tripwires, you'll be immediately alerted, allowing for rapid response and containment.

Tripwires are a form of honeytokens – decoys that look and feel like they belong in your network but are designed to attract attackers. These tokens are strategically deployed to assets at high risk, based on NodeZero autonomous pentest results. They are lightweight, resource-efficient, and remain silent until triggered. Only when activated does a tripwire communicate with Horizon3.ai's cloud infrastructure, ensuring minimal impact on your network's performance.

Features

Tripwires includes the following key capabilities:

- **Automated Deployment**: NodeZero deploys tripwires in high-risk assets automatically, simplifying the process of adding deception to your security stack.
- **Effective Threat Detection**: Tripwires are highly attractive to attackers and are always active.
- **Immediate Alerting**: As soon as malicious activity is detected on a tripwire, you'll receive alerts through the Portal and via email.
- **Seamless Integration**: Tripwire alerts can be integrated into your existing workflows through webhooks, allowing notifications to flow into your SIEM or other monitoring tools.

What are Tripwires?

Tripwires serve as highly attractive targets for attackers, often placed in assets where NodeZero detected vulnerabilities during testing. Unlike traditional beacons, they don't communicate until they're triggered, at which point they alert Horizon3.ai's systems.

In most cases, NodeZero may deploy up to two tripwires per asset during testing, depending on the asset type. Tripwires are placed where NodeZero has gained access through exploiting a vulnerability, deploying either a Remote Access Tool (RAT) or by directly writing to the asset.

Tripwire Types

Currently, there are three types of tripwires that NodeZero deploys:

Type	Details	Where they're placed	Detection method
AWS Credentials File	Real credentials generated by Horizon3.ai that blend into your network environment.	Network shares, Windows/Linux machines	When an actor attempts to use the credentials, NodeZero detects the attempt.
MySQL Dump File	Deceptive dump file, crafted to attract attention from attackers.	Network shares, Linux servers	An actor attempting to import or load the contents triggers a DNS callback to NodeZero systems, allowing detection.
Windows Suspicious Process Monitor	Monitors these processes: tasklist.exe, certutil.exe, systeminfo.exe, netstat.exe, at.exe.	Windows 7 (and later) machines	When an actor executes monitored commands, NodeZero detects the activity and sets off a DNS callback to NodeZero systems.

Final Notes

NodeZero Tripwires act as a passive yet powerful defense mechanism, silently waiting until a malicious actor attempts to interact with them. By strategically placing these decoys in high-value areas, you gain crucial insight into potential threats, enabling swift, decisive action when an attack occurs.

Best of all, Tripwires integrate seamlessly into your existing workflows, enhancing your security posture without adding manual overhead. This makes them an invaluable asset in defending your organization from increasingly sophisticated adversaries.

Ready to take the next step? Explore the Getting Started guide to dive deeper into configuring and using Tripwires, and learn how you can start leveraging them to strengthen your security defenses today!

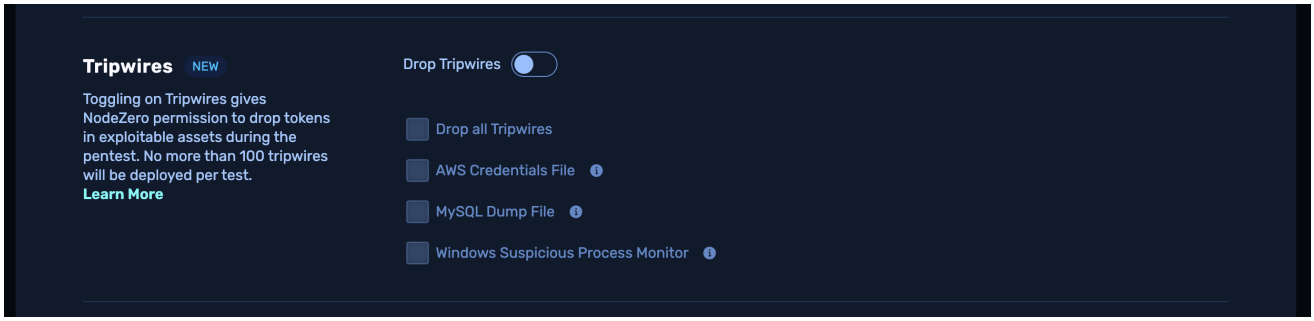
4.2 Getting Started with NodeZero Tripwires

Tripwire deployment can be enabled in any **Internal**, **External**, **Phishing Impact**, or **Rapid Response** pentest, giving you the power to integrate early threat detection seamlessly into your existing security tests.

4.2.1 Enabling Tripwires in a Pentest

To activate tripwire deployment, follow these steps:

1. In the configuration or template for any **Internal**, **External**, **Phishing Impact**, or **Rapid Response** pentest, scroll to the new **Tripwires** section.
2. Click the toggle to enable NodeZero to deploy tripwires once the test begins.



NodeZero RAT

Note: Enabling Tripwires also activates the NodeZero Remote Access Tool (RAT), which is essential for deploying tripwires in your environment. Only Rapid Response tests that make use of the RAT are able to drop tripwires.

4.2.2 Injecting Credentials

If you inject one or more credentials as part of the pentest, NodeZero will also leverage those credentials to deploy tripwires where the credential grants write access.

4.2.3 Viewing and Managing Deployed Tripwires

After tripwires have been deployed in your environment, you'll be able to monitor them in the Portal, even while the test is still running. From the top navigation bar, select **Tripwires** and go to the **Manage** page.

On the **Manage** page, you can:

- View all deployed tripwires.
- See details such as the exact location of each tripwire in your network, the pentest during which it was deployed, and instructions on how to remove it.
- Use search and filter options to find specific tripwires.
- Test tripwires to simulate alerting and understand how the system will respond to an actual attack.

For more information on managing tripwires, visit the Tripwire Management page.

4.2.4 Testing a Tripwire

Testing a tripwire simulates an action that a malicious actor might take to trigger the tripwire. The simulation may take a few minutes to complete. Once the test is finished, you'll see a new alert indicated by red dots on the **Tripwires** in the top nav and in **Alerts** within the sub nav.

When an alert is triggered, users with access to NodeZero Tripwires will also receive an email notification. If you've configured webhook integrations, an alert event will be sent to the connected system as well.

4.2.5 Viewing and Managing Alerts

All tripwire alerts, whether from tests or real incidents, are shown on the **Alerts** page, accessible under **Tripwires** in the top navigation bar. Each alert includes comprehensive details, such as:

- Information on the actor and the specific activity detected.
- Relevant tripwire information to help your incident response team act quickly.

For details on alert management, check out the Alerts page.

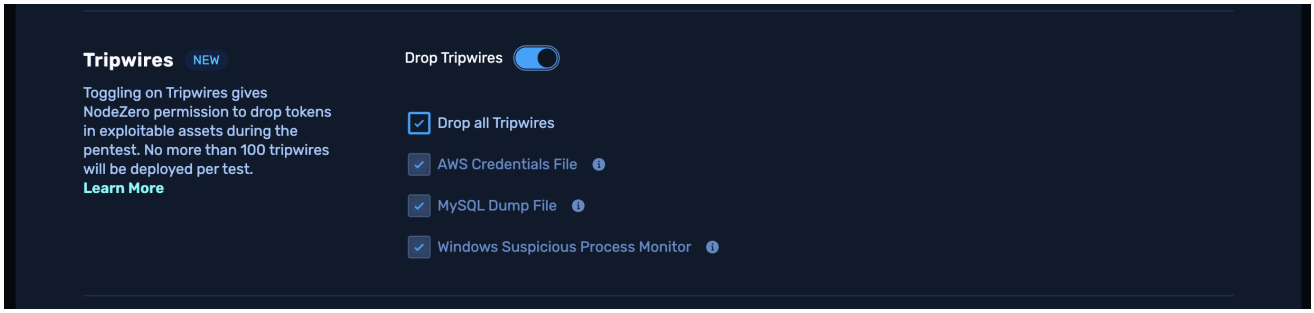
4.3 Tripwire Management

4.3.1 Dropping Tripwires

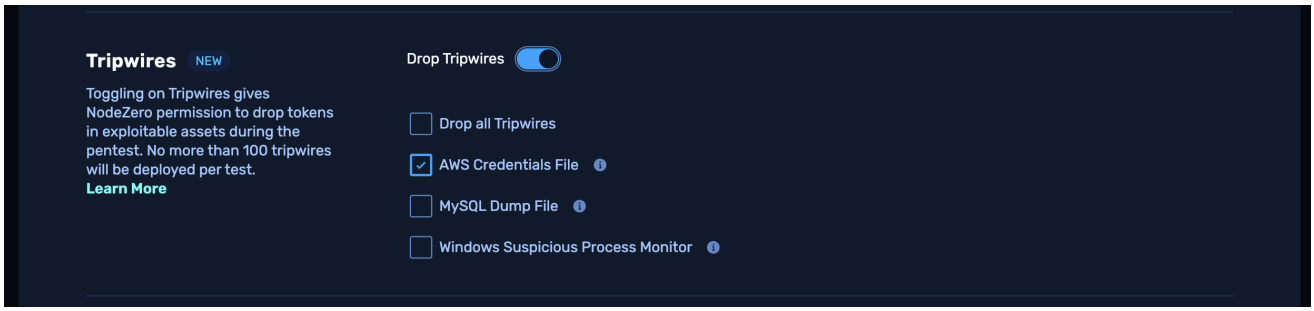
Tripwire deployment can be enabled during any **Internal**, **External**, **Phishing Impact**, or **Rapid Response** pentest. If you're running an ongoing scheduled test, you can update the test template to enable tripwire deployment.

Directions

1. Go to the configuration or template for an **Internal**, **External**, **Phishing Impact**, or **Rapid Response** pentest.
2. Scroll to the **Tripwires** section.
3. Click the toggle to enable **Tripwires**. Enabling Tripwires will also activate the NodeZero Remote Access Tool (RAT), which is required for tripwire deployment.



4. (Optional:) Select the specific types of tripwires you want in your environment; NodeZero will only drop the selected types. By default, NodeZero will drop all types as appropriate.



5. Save your changes or start the pentest.

> **Rapid Response pentest**
>
> Only Rapid Response tests that make use of the RAT are able to drop tripwires.

During the pentest, NodeZero will attempt to drop tripwires on hosts that it successfully compromised and deployed the RAT to. If NodeZero gains write access to a network share, it will also attempt to drop tripwires on that share.

After the pentest is complete, the test results will show the number of tripwires deployed, and the person who initiated the pentest will receive an email notification with details about the newly dropped tripwires, along with the pentest completion summary.

Injecting a credential

If you inject one or more credentials as part of the pentest, NodeZero will also leverage those credentials to deploy tripwires where the credential grants write access.

4.3.2 Managing Tripwires

All tripwires deployed by NodeZero can be viewed and managed in the **Manage** page under **Tripwires** in the top navigation.

The table on this page provides detailed information about each tripwire, including:

- The type of tripwire.
- When it was deployed.
- The asset it was deployed to.
- Its current status.
- Links to the pentest during which the tripwire was dropped.

You can filter tripwires by **Status** or **Type**, or use the search functionality to find specific tripwires, such as by the pentest in which they were deployed.

Clicking on a tripwire's name will bring up a details page with additional information, such as the tripwire’s location and links to the impacts related to the asset from the pentest during which the tripwire was dropped.

The details page also includes instructions on how to delete the tripwire.

4.3.3 Removing Tripwires

If you want to remove tripwires, there are two actions to take to fully remove a tripwire: delete and deactivate. Just deleting a tripwire will not automatically set the tripwire as inactive from NodeZero's perspective.

Deleting Tripwires

NodeZero is an agentless solution, so it does not have the capability to delete tripwires after deployment. Once dropped, the tripwire is active and will trigger alerts if any activity is detected on it.

Each tripwire details page includes instructions on how to delete the tripwire. Follow the instructions to delete the tripwire.

Deactivating Tripwires

Deactivating a tripwire means NodeZero will no longer listen for activity on the tripwire. It does not mean your tripwire has been deleted from your environment.

Active, or "live", tripwires will have the **Active** status, and deactivated tripwires will have the **Inactive** status.

To deactivate one or more tripwires: 1. Click the checkbox next to the name(s) of the desired tripwire. 2. Click **Deactivate Tripwires** button at the top right of the table.

To deactivate all tripwires: 1. Click the checkbox in the top left of the tripwires table 2. Click **Deactivate Tripwires** button at the top right of the table.

Deactivation is permanent

Once you deactivate a tripwire, you will not be able to reactivate it.

4.3.4 Testing Tripwires

From a tripwire’s details page, you can simulate an attacker's behavior by testing the tripwire. This test is run within Horizon3.ai’s infrastructure and simulates the kind of action that would trigger the tripwire in a real scenario.

The test may take a few minutes to complete. Once finished, a test alert will be generated, appearing in the **Portal**. Notifications will also be sent via email, and a webhook event (if configured with a vendor) will be triggered. Test alerts are clearly labeled as such in the **Portal**, email, and webhook event.

4.4 Tripwire Alerts

All tripwire alerts – both test and real – will appear in the NodeZero Portal under **Tripwires > Tripwire Alerts**.

Test alerts are clearly labeled as such in the Portal, email notifications, and webhook events.

4.4.1 Alert Contents

Any potentially malicious activity that NodeZero detects on a tripwire will trigger an alert. Each alert includes detailed information about:

- The actor involved.
- The specific activity detected.
- Relevant tripwire details to assist incident responders in taking swift and informed action.

If there are any unread alerts, a red dot will appear next to **Tripwires** in the top navigation and **Alerts** in the sub-navigation. You can mark an alert as read by clicking the **Mark as Read** button on each individual alert.

4.4.2 Timestamp Offsets

While NodeZero generates an alert as soon as it detects activity on a tripwire, there may be a slight difference between the **Tripped On** timestamp and the alert timestamp. This is due to the multiple detection methods NodeZero uses, some of which may take a bit more time.

4.4.3 Consolidated Alerts

NodeZero will generate one alert per tripwire per asset, even if an attacker takes multiple actions on the same tripwire in a short period. This is designed to reduce alert noise, and to avoid overwhelming your team with redundant notifications.

4.5 Testing

4.5.1 Testing Methodology

Testing your Tripwires deployment is essential to ensure that the system is correctly configured and will detect real attacks when they occur. This section covers how to safely test your Tripwires without causing false alarms or disrupting your production environment.

Overview

Tripwires testing helps you:

- **Verify Detection**: Confirm that tripwires are correctly detecting simulated attack activities
- **Validate Alerting**: Ensure alerts are properly configured and delivered to the right recipients
- **Check Integration**: Test webhook integrations with SIEM systems and other monitoring tools
- **Assess Response**: Validate that your incident response procedures work with Tripwire alerts

Testing Types

GENERAL TRIPWIRES TESTING

Test standard tripwires including AWS credentials, MySQL dumps, and Windows process monitors. General tripwire testing only validates Horizon3 infrastructure and does not impact your customer environment.

Learn more about General Tripwires Testing →

AD TRIPWIRES TESTING

Test Active Directory-specific tripwires including domain user scraping detection, Kerberoasting, and AS-REP roasting. AD tripwire testing generates real events in your customer environment and requires careful coordination.

Learn more about AD Tripwires Testing →

Testing Methodology

SAFE TESTING PRACTICES

When testing Tripwires, follow these guidelines to avoid disrupting production systems:

1. **Use Test Credentials**: Only use designated test accounts and credentials
2. **Coordinate with Security Team**: Inform your security team before conducting tests
3. **Test During Maintenance Windows**: Schedule tests during off-peak hours when possible
4. **Document Test Activities**: Keep detailed logs of all testing activities
5. **Clean Up After Testing**: Remove test artifacts and reset systems as needed

TEST TYPES

Functionality Testing

Verify that tripwires are correctly placed and functional:

- Check that tripwire files are present on target systems
- Confirm tripwire services are running where applicable
- Validate network connectivity to Horizon3.ai monitoring systems

Detection Testing

Test the actual detection capabilities:

- Simulate attacks against deployed tripwires
- Verify that alerts are generated within expected timeframes
- Check that alert details contain accurate information

Integration Testing

Validate external system integrations:

- Test webhook delivery to SIEM systems
- Verify email alert delivery
- Check Portal notification functionality

Testing Best Practices

PRE-TEST PREPARATION

Before conducting any tripwire testing:

- [ ] Notify security and operations teams
- [ ] Verify backup alert recipients are configured
- [ ] Document baseline system state
- [ ] Prepare rollback procedures if needed
- [ ] Set up monitoring for test results

DURING TESTING

While conducting tests:

- [ ] Follow documented test procedures
- [ ] Monitor for unexpected system behavior
- [ ] Record detailed test logs
- [ ] Track alert generation timing
- [ ] Note any integration failures

POST-TEST VALIDATION

After completing tests:

- [ ] Verify all expected alerts were generated
- [ ] Check alert delivery to all configured endpoints
- [ ] Review test logs for anomalies
- [ ] Clean up test artifacts
- [ ] Document test results and recommendations

Troubleshooting Test Issues

COMMON TESTING PROBLEMS

Alerts Not Generated

- Check network connectivity from target systems
- Verify tripwires are properly deployed
- Confirm monitoring services are operational

Delayed Alerts

- Network latency may cause delays
- Check for DNS resolution issues
- Verify webhook endpoint availability

Missing Integration Alerts

- Test webhook endpoints independently
- Check SIEM system connectivity
- Verify authentication credentials

GETTING HELP

If you encounter issues during testing:

1. Check the specific testing guides for detailed troubleshooting steps
2. Review system logs for error messages
3. Contact Horizon3.ai support with detailed test logs
4. Provide specific error messages and timestamps

Testing Considerations

WHEN TO USE UI TESTING VS MANUAL TESTING

Use UI Testing When:

- You want to validate alert delivery and integration functionality
- Testing in production environments where you want to avoid impact
- Performing regular testing schedules
- Verifying webhook and email configurations

Use Manual Testing When:

- You need to validate end-to-end tripwire functionality
- Testing in development or lab environments
- Troubleshooting specific detection issues
- Performing comprehensive security validation

TESTING FREQUENCY

- **Weekly**: UI testing of critical alert delivery mechanisms
- **Monthly**: Mixed UI and manual testing of different tripwire types
- **Quarterly**: Comprehensive manual testing in lab environments
- **Annually**: Full-scale testing as part of security assessments

Continuous Testing

Regular testing ensures ongoing tripwire effectiveness:

MONTHLY TESTING SCHEDULE

- Test a subset of deployed tripwires
- Rotate testing across different tripwire types
- Validate alert delivery mechanisms

QUARTERLY REVIEWS

- Comprehensive testing of all tripwire types
- Review and update testing procedures
- Assess integration health and performance

ANNUAL ASSESSMENTS

- Full-scale testing exercise
- Review tripwire placement strategy
- Update testing documentation and procedures

Regular testing of your Tripwires deployment ensures that your deception technology remains effective and your security team is prepared to respond to real threats when they occur.

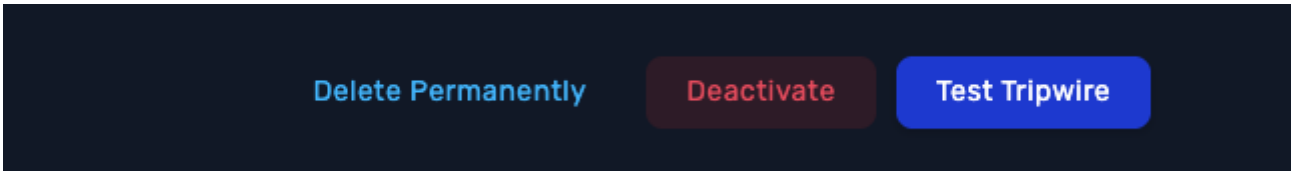
4.5.2 General Tripwires Testing

This guide covers testing standard tripwires including AWS credentials, MySQL dumps, and Windows process monitors. General tripwire testing validates Horizon3 infrastructure components without impacting your customer environment.

Using the Test Tripwire Button

The easiest way to test your tripwires is through the NodeZero Portal interface:

1. **Navigate to Tripwire Details**: Go to the specific tripwire you want to test in the NodeZero Portal
2. **Click Test Tripwire**: At the bottom of the tripwire details page, click the `Test Tripwire` button
3. **Monitor for Alerts**: The system will simulate an attack against the tripwire and generate a test alert
4. **Verify Results**: Check that alerts are properly delivered through all configured channels



Infrastructure Testing Only

General tripwire testing using the `Test Tripwire` button **tests only the Horizon3 infrastructure and components**. This includes alert generation, delivery mechanisms, and integration endpoints. It does not generate actual activity or events in your environment.

Why Test Tripwires?

This built-in testing feature is the recommended method for validating tripwire functionality, because it:

- Safely tests Horizon3 infrastructure without impacting your environment
- Automatically generates test alerts through the proper detection pipeline
- Validates alert delivery and integration endpoints
- Doesn't require access to target systems or generate customer environment activity

4.5.3 AD Tripwires Testing

This guide covers testing Active Directory-specific tripwires including domain user scraping detection, Kerberoasting, and AS-REP roasting. AD tripwire testing generates real events in your customer environment and requires careful coordination.

Pre-Testing Requirements

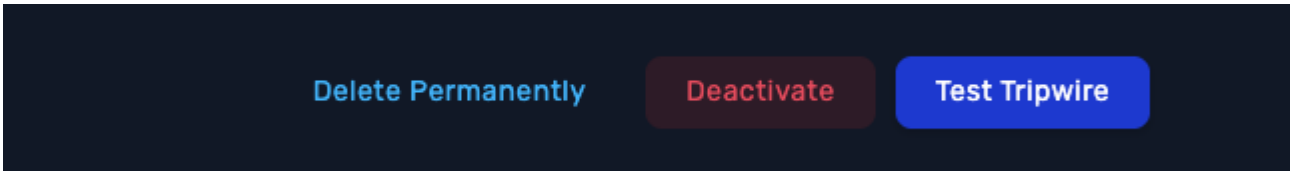
Before testing AD Tripwires, ensure:

- AD Tripwires domain policy is properly deployed
- Tripwire accounts are created and configured
- Event Collector is running on Domain Controllers
- AD Agent service is running on the NodeZero Runner
- Security team is notified of testing activities

Using the Test Tripwire Button

The recommended way to test your AD Tripwires is through the NodeZero Portal interface:

1. **Navigate to Tripwire Details**: Go to the specific AD tripwire you want to test in the NodeZero Portal
2. **Click Test Tripwire**: At the bottom of the tripwire details page, click the `Test Tripwire` button
3. **Wait for AD Agent Processing**: The system sends a message to the AD Agent to perform the test
4. **Allow Processing Time**: Due to event collection intervals and processing delays, expect **up to 20-30 minutes** before receiving notifications
5. **Check Portal for Updates**: Detection may appear in the Portal before alerts are delivered, so check the Portal periodically
6. **Monitor for Alerts**: Check for alerts through all configured channels after the delay period



End-to-End Testing with Customer Environment Impact

AD Tripwires testing is **end-to-end testing that generates actual activity and log events in your customer environment**. Unlike general tripwire testing which only tests Horizon3 infrastructure, AD Tripwire testing:

- **Generates real AD events** on your Domain Controllers
- **Creates actual log entries** in Windows Security Event logs
- **Triggers real authentication attempts** against tripwire accounts
- **May activate security monitoring** systems in your environment

Coordinate with your IT and security teams before testing, as this will create genuine activity that may be detected by other security tools.

Processing and Display Delays

AD Tripwires testing involves multiple processing stages that can affect timing:

- **Detection Processing**: The actual detection may occur within 10 minutes
- **Portal Updates**: It may take additional time for the detection to populate and display in the Portal
- **Alert Delivery**: Email and webhook notifications may arrive separately from Portal updates

Plan for up to 20-30 minutes for the complete testing cycle, including Portal display updates.

This built-in testing feature:

- Performs end-to-end testing of the complete AD Tripwires detection pipeline
- Generates actual AD events and log entries in your customer environment
- Automatically creates realistic attack scenarios through the AD Agent
- Validates the complete detection pipeline including event collection and processing
- Tests alert delivery through all configured integration points

Event Collection Testing

VERIFY EVENT COLLECTOR FUNCTIONALITY

1. **Check Scheduled Task**

```
# Verify Event Collector scheduled task is running
Get-ScheduledTask -TaskName "IoA Collector" | Get-ScheduledTaskInfo
```

2. **Monitor Event Forwarding**

```
# Check recent security events
Get-WinEvent -FilterHashtable @{LogName='Security'; ID=4768,4769,4771; StartTime=(Get-Date).AddHours(-1)}
```

3. **Validate Network Connectivity**

```
# Test connectivity to Horizon3.ai collection endpoints
Test-NetConnection -ComputerName tripwires.horizon3.ai -Port 443
```

Alert Delivery Testing

For detailed alert delivery testing procedures including webhook and email testing, see the General Tripwires Testing Guide.

Note that AD Tripwires alerts may take 20-30 minutes to be delivered due to the event processing pipeline, so allow adequate time when testing alert delivery mechanisms.

Timing Considerations

AD Tripwires testing requires patience due to the distributed nature of Active Directory and multiple processing stages:

- **Event Collection Intervals**: Domain Controllers may batch events before forwarding
- **AD Agent Processing**: The AD Agent processes events in scheduled intervals
- **Detection Processing**: Backend systems analyze events and generate detections
- **Portal Updates**: Detections must be processed and displayed in the user interface
- **Alert Processing**: Final alert generation and delivery to external systems

Expected Timeline:

- 0-5 minutes: Test action is initiated via AD Agent
- 5-15 minutes: Events are generated and collected on Domain Controllers
- 10-20 minutes: Events are processed and tripwire activity is detected
- 15-25 minutes: Alerts appear in the NodeZero Portal and are delivered to configured endpoints

> **Monitoring Strategy**
>
> While detection may happen faster, plan for the full 20-30 minute window. Check the Portal periodically rather than waiting only for external alerts.

AD-Specific Safety Measures

In addition to general safety measures, AD Tripwires testing requires:

- **Coordinate with IT and Security Teams**: AD testing generates real events that may trigger other security monitoring systems
- **Notify SOC/Security Teams**: Inform security operations centers about planned testing to avoid false alarm responses
- **Monitor Domain Controller Resources**: Watch for performance impact on Domain Controllers during testing
- **Plan for Real Events**: Remember that AD testing creates actual log entries that will appear in security logs

4.6 Settings

4.6.1 Grant access to NodeZero Tripwires

As an Org Admin, you can grant any team member with access to NodeZero the ability to fully manage NodeZero Tripwires, regardless of their role. When a team member is given full access to Tripwires, they will be able to:

- View the **Tripwires** tab.
- Manage all deployed tripwires.
- Receive email notifications for all tripwire alerts.

If the team member also has the appropriate roles to run pentests, they will additionally be able to enable the tripwire option in pentest configurations and templates.

Granting Full Access to Tripwires

To grant full access to Tripwires for a team member, follow these steps:

1. Navigate to the **Settings** page and select **User Management**.
2. Locate the team member you wish to grant access to and click the three dots next to their name.
3. Click **Edit User**.
4. Toggle on **Full Access to Tripwires**.
5. Click **Edit** to save the changes.

Once granted, the team member will have full management capabilities for tripwires and will receive all relevant alert communications.

4.6.2 Splunk Cloud Connection

Integrations

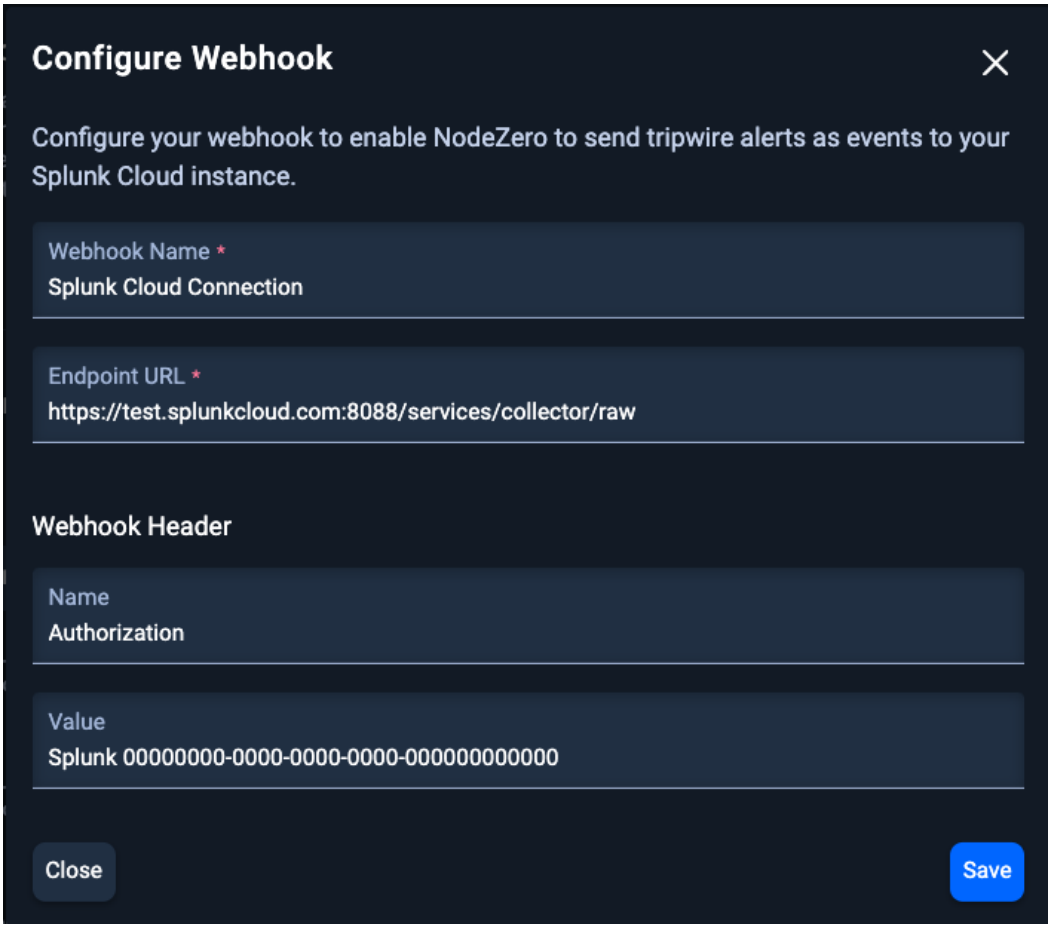
Only one instance of Splunk Cloud can be connected at a time.

Org Admins can integrate tripwire alerts with Splunk Cloud using webhook connections. Once configured, NodeZero will send all tripwire alerts to the connection.

Configuring a Webhook

To connect a new vendor to receive tripwire alerts via a webhook:

1. Navigate to the **Settings** page and go to **Integrations**.
2. Click **Create Webhook**.
3. Fill out the configuration details in the modal and save.



TESTING A WEBHOOK CONFIGURATION

To test a webhook configuration:

1. In the **Settings** page, go to **Integrations**.
2. Click the carrot (dropdown arrow) next to the webhook configuration you want to test.
3. Click **Test Webhook**.

UPDATING A WEBHOOK CONFIGURATION

To update a webhook configuration:

1. In the **Settings** page, go to **Integrations**.
2. Click the carrot next to the webhook configuration you wish to update.
3. Click the three dots and select **Edit Webhook**.
4. Make your changes in the modal and save.

DELETING A WEBHOOK CONFIGURATION

To delete a webhook configuration:

1. In the **Settings** page, go to **Integrations**.
2. Click the carrot next to the webhook configuration you wish to delete.
3. Click the three dots and select **Delete Webhook**.
4. Confirm the deletion from the modal.

Viewing a Sample Payload

Tripwire alerts may vary depending on the type of tripwire. A sample event payload is provided, showing all possible fields with sample values that could be sent.

To view the sample payload:

1. In the **Settings** page, go to **Integrations**.
2. Click the carrot next to the webhook configuration.
3. Click the three dots and select **View Sample Payload**.

Viewing Webhook Events

All webhook events are logged in the **Integrations** page under the **Event Log** section. The log includes all events that NodeZero attempted to send and their status.

- **SENT** indicates the event was successfully delivered.
- **FAIL** indicates the event failed to send.

SETTING UP A HTTP EVENT COLLECTOR (HEC) IN SPLUNK CLOUD

1. **Enable HEC**:
2. In Splunk Web, go to **Settings > Data Inputs > HTTP Event Collector**.
3. Click **Global Settings** and toggle **All Tokens** to **Enabled**.
4. **Create a New HEC Token**:
5. Click **New Token** and configure your token (Name, Source Type, Index, etc.).
6. Click **Next**, then **Review**, and **Submit**.
7. **Use the HEC Token**:
8. Use the token you created to configure the webhook as described above.

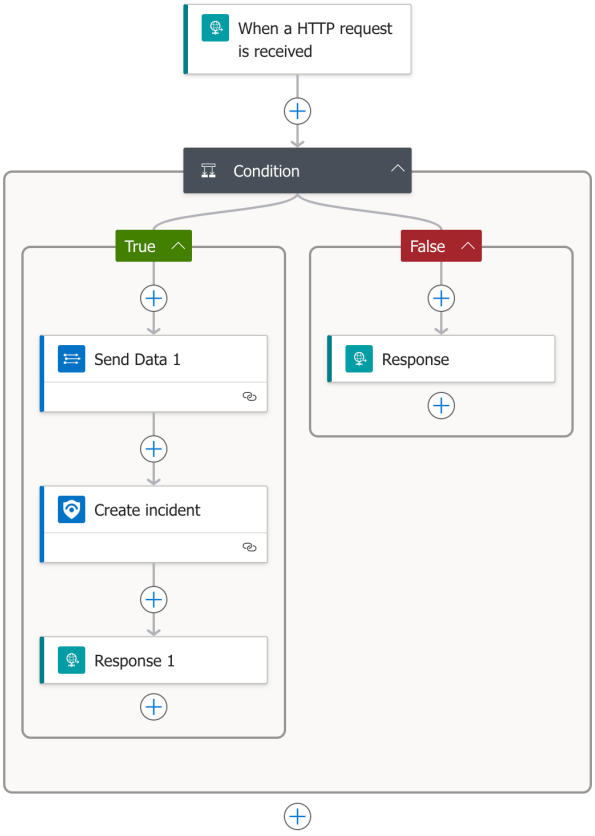
For more detailed information about Splunk HEC, refer to Splunk's own HTTP Event Collector documentation.

4.6.3 Guide: Building a Logic App in Azure to Automate Conditional Actions

Integrations

Only one instance of Microsoft Sentinel can be connected at a time.

This guide will show you how to create a Logic App in Azure that responds to incoming HTTP requests from NodeZero's tripwire alerts and sends the data to Microsoft Sentinel as well as triggering different actions based on specific conditions. Follow these steps to create a similar flow as seen below.



Create a New Logic App

Navigate to your `Azure Portal`

1. LOG IN TO AZURE PORTAL

- Search for `Logic Apps` in the search bar.
- Click on `Create Logic App` and fill in the necessary details such as Subscription, Resource Group, and Region.
- Once creation is complete click `Go to resource`.
- Select `Logic App Designer` under Development tools in the menu on the left of the screen.

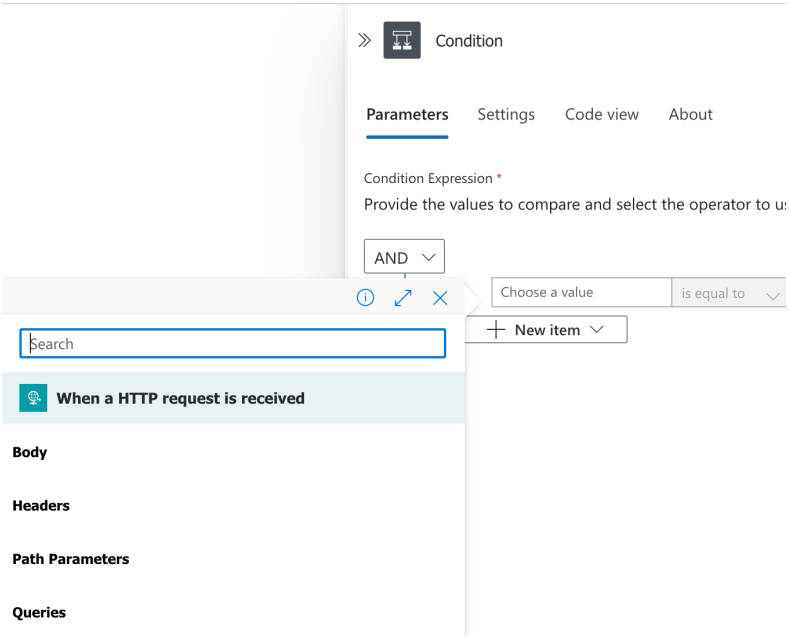
2. ADD THE TRIGGER

- Choose the Trigger

- Click `Add a trigger` and search for `When an HTTP request is received` as the trigger. This will allow your Logic App to listen for HTTP requests and act accordingly.
- When prompted, leave the Request Body JSON Schema empty for now.

3. DEFINE CONDITIONS

- Add a Condition to Check for a `x-api-key` Header.
- After the `HTTP request trigger`, click on the `+` symbol and choose `Add an action`.
- Search for `Control` and select `Condition` to add a conditional logic action.
- Set up the condition to evaluate the `x-api-key header` in the HTTP request. To do this:
- Add a new parameter by selecting the `Headers object` from the dynamic content.



- Ensure you add `x-api-key` in the right field.
- Click `Save` from the menu bar.

- If using a stored Parameter create that now.
- Click `Parameters` from the menu shown above and choose `Create Parameter`.
- Name: `ApiKey`
- Type: `String`
- Default value:
- Click `Save` from the menu bar.
- Click `Code view` from the menu.
- Set the condition to check if the `x-api-key` value matches a stored parameter (e.g., using an Azure Key Vault or a Logic App parameter as outlined above). In this example we are using a parameter named ApiKey defined in the logic app. In the Code view the section we are interested in will look like this and should be near the top.

```
"type": "If",
"expression": {
  "and": [
    {
      "equals": [
        "@triggerOutputs()?['headers']",
        ""
      ]
    }
  ]
},
```

- You need to add the `x-api-key` value and matcher like below.

```
"type": "If",
"expression": {
  "and": [
    {
      "equals": [
        "@triggerOutputs()?['headers']['x-api-key']",
        "@parameters('ApiKey')"
      ]
    }
  ]
},
```

4. DEFINE THE TRUE BRANCH ACTIONS

- Click `Save` from the menu bar and then click `Designer` from the same menu bar.
- Add `Send Data` Action (for True Condition)
- If the condition is met (True), click on the `Add an action` button inside the True branch.
- Search for `Azure Log Analytics Data Collector` and select `Send Data`.
- Fill in the required fields, Workspace ID and Workspace Key.
- Choose the required parameters, below we are choosing the body which contains the information as well as a custom log name.

- Add a `Response Action`.
- Click `+` New step again and search for `Response`.
- Set up the response that should be sent back after successfully creating the incident.
- Customize the Status Code (e.g., 200 for success) and any optional details like a response message.

5. DEFINE THE FALSE BRANCH ACTIONS

- Add a `Response Action` (for False Condition)
- In the False branch, click `Add an action`.
- Search for `Response` and configure it to return a different Status Code (e.g., 400 or 404, depending on your use case).
- Optionally add a message indicating why the condition failed.

6. SAVE AND CONFIGURE IN THE NODEZERO PORTAL

- Save Your Logic App
- Click `Save` at the top of the Logic App designer.
- Get the HTTP Request URL
- After saving, return to the `When an HTTP request is received` trigger and copy the generated `HTTP POST URL`.
- In the NodeZero Portal, navigate to `Settings` -> `Integrations` -> `Tripwires Webhooks`.
- Click `+` for `Microsoft Sentinel | Azure`.
- Name the connection
- Enter the Endpoint URL
- Add the Header
- Name: `x-api-key`
- Value:
- Click `Save`

- You can now test your Webhook.

  Summary

  You have now created a Logic App that receives HTTP requests, evaluates a condition, and performs different actions based on whether the condition is true or false. The true branch sends data to Sentinel, and sends a response, while the false branch simply sends a different response.

  From here you can expand on the actions with Sentinel automation and carry out further actions such as opening incidents, sending emails, etc.

4.7 AD Tripwires

4.7.1 **AD Tripwires** - Overview

AD Tripwires provides advanced detection capabilities for common Active Directory attack techniques by deploying decoy accounts and monitoring for suspicious authentication activities. This system creates honeypot-style traps that detect attackers attempting to enumerate domain users, perform credential-based attacks, and exploit Kerberos vulnerabilities.

Recommendation: After provisioning, we encourage disguising tripwire accounts with non-privileged, commonly populated attributes so they blend in with typical users. See Disguising Tripwires (Post-Provision Hardening).

Detection Capabilities

AD Tripwires monitors and detects the following attack patterns:

DOMAIN USER SCRAPING DETECTION

- **Mechanism**: Decoy credential exposure in user account descriptions
- **Detection**: Monitors for access attempts to accounts with exposed credentials
- **Attack Vector**: Detects attackers who scrape user descriptions to find embedded passwords or credential hints

KERBEROASTING DETECTION

- **Mechanism**: Service Principal Name (SPN) configured on decoy accounts
- **Detection**: Monitors for TGS (Ticket Granting Service) requests against honeypot service accounts
- **Attack Vector**: Detects attempts to request Kerberos service tickets for offline password cracking

AS-REP ROASTING DETECTION

- **Mechanism**: Pre-authentication disabled on specific decoy accounts
- **Detection**: Monitors for AS-REP (Authentication Server Response) requests against vulnerable accounts
- **Attack Vector**: Detects attempts to request authentication responses for accounts without pre-authentication

Components

```
graph TB
    domain_policy("Domain Policy")
    ad_agent("`AD Agent`")
    event_collector("`Event Collector`")
```

AD DOMAIN POLICY

An Active Directory domain policy which enables additional Kerberos logging events is required for detecting Kerberoasting and AS-REP Roasting attacks against **AD Tripwires**. This policy specifically enables enhanced audit logging for authentication events and creates a scheduled task on Domain Controllers to run the `Event Collector`.

Note: The AD Tripwires domain policy does **not** enable "Audit account management" events. Only authentication-related events required for attack detection are enabled.

The policy enables Windows Security Events 4625, 4768, 4769, 4771, and 4776 to provide the necessary audit trail for attack detection. Note that some of these events can generate significant log volume in busy environments.

Event scope and interactions

AD Tripwires looks for specific event types associated with each tripwire (e.g., TGS requests for Kerberoastable accounts, AS-REP responses for AS-REP Roastable accounts, failed logons for exposed credentials). However, the system evaluates any Windows Security event that involves a tripwire account and may generate tripwire events depending on your domain's audit configuration. Because these accounts should never be used legitimately, any interaction with a tripwire account is treated as potentially malicious.

For detailed information about specific events and their logging impact, see Windows Security Events.

PROVISIONING PROCESS

What it does:

- Creates and manages decoy user accounts with built-in vulnerabilities
- Sets up the monitoring infrastructure required for detection
- Configures the necessary permissions for secure event collection
- Provides easy removal and cleanup when needed

Tripwire Accounts (3 total)

- **Exposed Credential Account**: Appears to contain credentials that attackers might try to use
- **Kerberoastable Account**: Attracts attackers attempting Kerberos-based attacks
- **AS-REP Roastable Account**: Targets a specific type of Kerberos vulnerability exploitation

Note: Tripwire accounts are configured with long, complex, random passwords that are extremely unlikely to be cracked, even with dedicated hardware. These accounts are not used for legitimate logins and will not generate events for successful authentication. Only failed login attempts or suspicious activity will trigger detection events.

System Account

- `AD Agent` **Service Account**: Used by the `AD Agent` to securely access Active Directory and SYSVOL (not a decoy account)

AD AGENT

The `AD Agent` is a monitoring service that runs on your selected NodeZero host to provide centralized threat detection and alerting. It continuously monitors your tripwire accounts, analyzes potential threats, and delivers immediate alerts through your Horizon3 Portal.

Key Capabilities:

- **Monitors Activity**: Continuously watches for suspicious activity targeting your tripwire accounts
- **Sends Alerts**: Delivers immediate notifications when threats are detected
- **Provides Context**: Gives you detailed information about attack methods and timelines
- **Integrates Seamlessly**: Works directly with your Horizon3 Portal for unified security visibility

EVENT COLLECTOR

The `Event Collector` is a PowerShell component that runs as a scheduled task on Domain Controllers to gather security events related to AD Tripwire activities. It collects and caches IoA (Indicators of Attack) events for retrieval by the `AD Agent` for analysis and threat detection.

Key Functions:

- **Event Harvesting**: Collects security events from Domain Controller logs related to tripwire account activities
- **Data Filtering**: Focuses on authentication events that indicate potential attacks on decoy accounts
- **IoA Event Caching**: Stores Indicators of Attack events for secure access by the `AD Agent`
- **Secure Transfer**: Enables the `AD Agent` to retrieve cached events for centralized processing
- **Automated Execution**: Runs on a scheduled basis to ensure continuous monitoring coverage

For a detailed explanation of the detection process, see How It Works.

Benefits

- **Early Detection**: Identifies attacks in progress before significant damage occurs
- **Low False Positives**: Decoy accounts should never be accessed by legitimate users
- **Comprehensive Coverage**: Detects multiple attack vectors within a single deployment
- **Minimal Infrastructure Impact**: Lightweight monitoring with negligible performance overhead
- **Real-time Alerting**: Immediate notification of suspicious activities
- **Attack Attribution**: Provides detailed information about attacker techniques and timelines

Next Steps

To get started with **AD Tripwires**:

1. **Check Requirements**: Ensure you have Domain Admin privileges and access to your NodeZero Runner
2. **Set Up Accounts**: Follow the Getting Started guide to create your tripwire accounts and agents
3. **Enable Monitoring**: Use the Configuring Domain Policy guide to activate detection
4. **Verify Operation**: Test the system to ensure everything is working correctly
5. **Disguise Tripwires (Recommended)**: After provisioning, apply non-privileged persona attributes so decoy accounts blend in. See Disguising Tripwires (Post-Provision Hardening)

For help with issues or advanced settings, see the Troubleshooting guide.

4.7.2 AD Tripwires – System Requirements

PowerShell Requirements

Version	TLS 1.2+ Support	Status
PowerShell 7+	Enabled by default	✓ Tested and supported
PowerShell 5.1	Requires manual configuration	⚠ Not tested

Recommendation: Use PowerShell 7+ for guaranteed TLS 1.2+ support.

Security Requirements

- **TLS 1.2+** is required for all communications with AD Tripwires.

Performance Recommendations

- **DC Security Log Size**: Configure to 5 GB or less for optimal Event Collector performance.

Server Operating Systems

Operating System	Version	US English	Non-US Locales	Default PowerShell	Notes
Windows Server 2025	All editions	Supported	Not Tested	5.1	Native support
Windows Server 2022	All editions	Supported	Not Tested	5.1	Native support
Windows Server 2019	All editions	Supported	Not Tested	5.1	Native support
Windows Server 2016	All editions	Supported	Not Tested	5.1	Native support
Windows Server 2012 R2	All editions	Supported*	Not Tested	4.0	*Requires PS 5.0+ or PS 7+ installation
Windows Server 2012	All editions	Supported*	Not Tested	3.0	*Requires PS 5.0+ or PS 7+ installation
Windows Server 2008 R2	SP1	Supported*	Not Tested	2.0	*Requires PS 5.1 or PS 7+ installation
Windows Server 2008	All	Unsupported	Unsupported	2.0	Cannot run PS 5.0+; PS 7 not supported

Client Operating Systems

Operating System	Version	US English	Non-US Locales	Default PowerShell	Notes
Windows 11	All versions	Supported	Not Tested	5.1	Native support
Windows 10	1607+	Supported	Not Tested	5.1	Native support
Windows 10	Pre-1607	Supported*	Not Tested	5.0	*Requires PS 5.1 or PS 7+ installation
Windows 8.1	All	Supported*	Not Tested	4.0	*Requires PS 5.1 or PS 7+ installation
Windows 8	All	Supported*	Not Tested	3.0	*Requires PS 5.1 or PS 7+ installation
Windows 7	SP1	Supported*	Not Tested	2.0	*Requires PS 5.1 or PS 7+ installation
Windows Vista	All	Unsupported	Unsupported	2.0	Cannot run PS 5.0+; PS 7 not supported

Support Legend

- **Supported**: Tested and confirmed working with AD Tripwires
- **Not Tested**: Not formally tested but should work; use at your own discretion
- **Unsupported**: Does not meet minimum PowerShell requirements
- **Supported***: Requires manual installation of PowerShell 5.1+ or PowerShell 7+

Notes

1. All modern Windows Server and Client versions (2016/Win10 1607+) include PowerShell 5.1 by default
2. Older operating systems can be supported by installing PowerShell 5.1 or PowerShell 7+
3. PowerShell 7+ can be installed side-by-side on any supported Windows version
4. Non-US localized versions have not been formally tested but should function correctly
5. Core/Nano server installations are supported if they meet PowerShell requirements

4.7.3 Guides

AD Tripwires – Getting Started

PREVENT SIMULTANEOUS PENTESTS

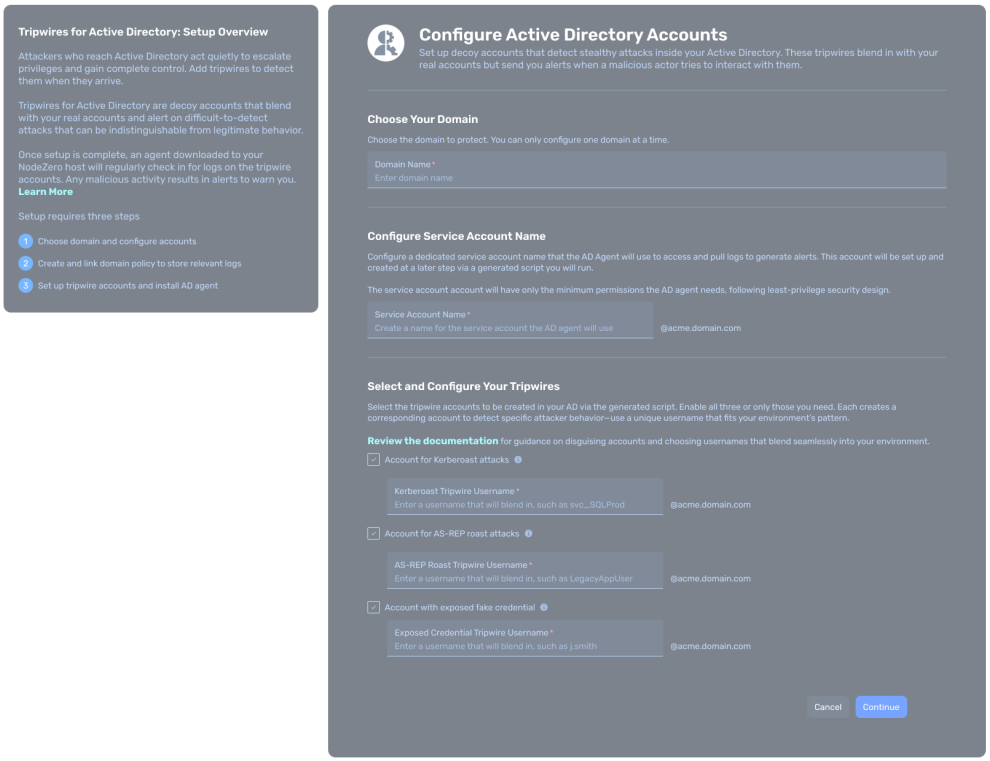
> **NodeZero pentest consideration**
>
> If a NodeZero pentest is running while you set up AD Tripwires, this might cause false-positive tripwire triggers.

Let ongoing pentests complete, or cancel ongoing pentests, before proceeding with AD Tripwire setup – specifically, before running the PowerShell command to configure accounts.

If you opt to cancel pentests, you can start new pentests after AD Tripwire accounts are properly configured.

INITIAL SETUP

1. Configure Active Directory Accounts

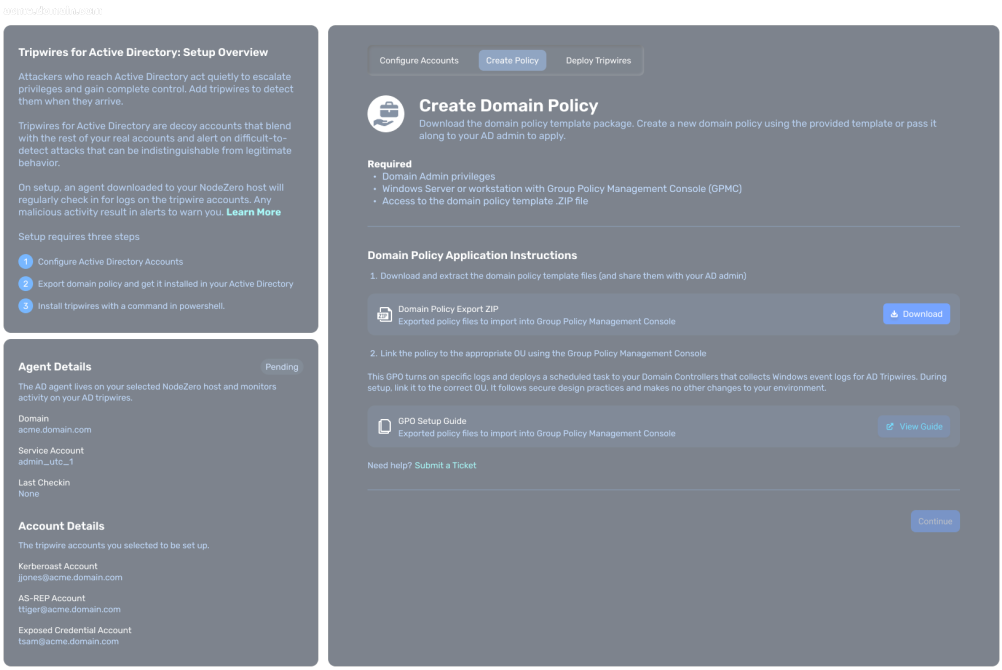


- Specify the Active Directory domain to configure for tripwires
- Select AD tripwire account types
- Specify user names for each selected AD tripwire account type
- Specify a user name for a new service account that will be used by the `AD Agent` to audit tripwire events

Kerberoastable account SPN requirement

The Kerberoastable tripwire account must always have at least one Service Principal Name (SPN). You may change the SPN value(s) to match your environment, but removing all SPNs will cause the account to fail the periodic health checks.

2. Create Domain Policy



The domain policy enables additional security event logging required for **AD Tripwires** detection across all domain controllers. Some events are enabled by default in Windows, while others are specifically enabled by the IoA Domain Policy.

Event Coverage:

- **Events enabled by default in Windows:**
- 1100: Event Log service has shut down
  Detects attempts to disable or tamper with event logging.
- 1101: Audit events dropped by transport
  Indicates possible loss of audit data due to system or network issues.
- 1102: Audit log cleared
  Alerts when the security log is cleared, which may indicate attacker log cleanup.
- 1104: Security log full
  Warns when the security log is full and may stop recording new events.
- **Events enabled by the IoA Domain Policy:**
- 4625: Failed logon attempt (Kerberos or NTLM)
  Detects failed login attempts to tripwire accounts, such as password spraying or brute force attacks.
- 4768: Kerberos authentication ticket (TGT) was requested
  Detects AS-REP roasting attempts against tripwire accounts with pre-authentication disabled.
- 4769: Kerberos service ticket (TGS) was requested
  Detects Kerberoasting attempts by monitoring TGS requests for tripwire service accounts.
- 4771: Kerberos pre-authentication failed
  Identifies failed Kerberos authentication attempts, including brute force or password guessing against tripwire accounts.
- 4776: NTLM authentication failed (NTLM credential validation failure)
  Detects failed NTLM authentication attempts to tripwire accounts, such as brute force or password guessing using NTLM.

Requirements:

- Domain policy template ZIP file (downloaded from the Portal)
- Domain Admin privileges
- Windows Server or workstation with Group Policy Management Console (GPMC) installed

Quick Setup Overview:

1. Download the domain policy template ZIP file from the Portal
2. Create and configure a new Group Policy Object (GPO)
3. Import the provided policy template
4. Link the GPO to the Domain Controllers OU

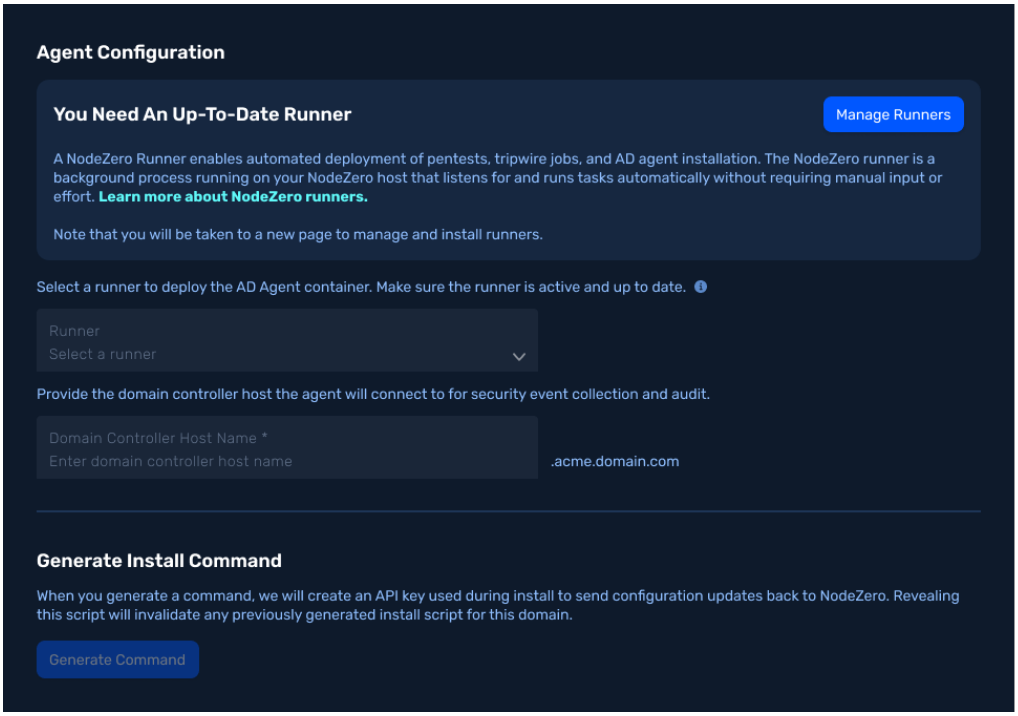
Detailed Instructions: For complete step-by-step guidance with screenshots, see the Configuring Domain Policy guide

3. Deploy Tripwires and AD Agent

- Specify the hostname of the domain controller the `AD Agent` will access to audit tripwire events
- Specify how frequently the `AD Agent` will audit tripwire event logs
- Specify which Runner should be used to install the `AD Agent`
- If no eligible Runners are available, visit the Runners page to update your desired Runner

Provision AD Accounts & Event Logging

Error shown when no eligible NodeZero Runner is available for agent deployment.



Permission Requirements

The setup process requires specific permissions to provision the **AD Tripwires** infrastructure. The executing account must have:

Active Directory Permissions:

- **User Account Management**: Create, delete, and modify user accounts
- **Service Principal Name (SPN) Management**: Set SPNs on user accounts for Kerberoastable tripwires
- **User Attribute Modification**: Modify user properties including:
- Account descriptions (for exposed credential tripwires)
- Pre-authentication settings (for AS-REP Roastable tripwires)
- Account control flags

SYSVOL Permissions:

- **Directory Creation**: Create the EventAnalytics directory in `SYSVOL`
- **Permission Management**: Set appropriate permissions on the created directory
- **Full Control**: Complete access to modify SYSVOL directory structure

Implementation Options:

1. **Domain Admin Account** (Recommended): Includes all necessary permissions
2. **Custom Permissions**:
3. Add account to "Account Operators" group
4. Grant "Full Control" on SYSVOL share
5. Assign "Write All Properties" permission on User objects in AD

Setup Requirements

1. Requirements:
a. Windows server or workstation joined to the target domain
b. Account with Domain Admin privileges or specific permissions listed above
2. Generate command to run the provisioning process, which will:
a. Configure tripwire event processing
b. Provision a service account to enable H3 to access tripwire event data
c. Provision all tripwire accounts
3. Open PowerShell as Administrator
4. Paste and run the command in the Administrator PowerShell session
5. Confirm all tasks were completed successfully via Portal

Domain Mismatch

In a multi-domain environment, you must ensure that the provided command is executed on a machine joined to the target domain.

Verify AD Agent Installation

1. `AD Agent` installation will be automatically triggered after AD accounts are successfully provisioned and the provisioning process reports back to Portal.
2. The `AD Agent` should become active within 5-10 minutes

> **Post-provision hardening**
>
> After the utility provisions tripwire accounts, optionally apply non-privileged persona attributes so accounts blend in with real users. See: Disguising Tripwires (Post-Provision Hardening).

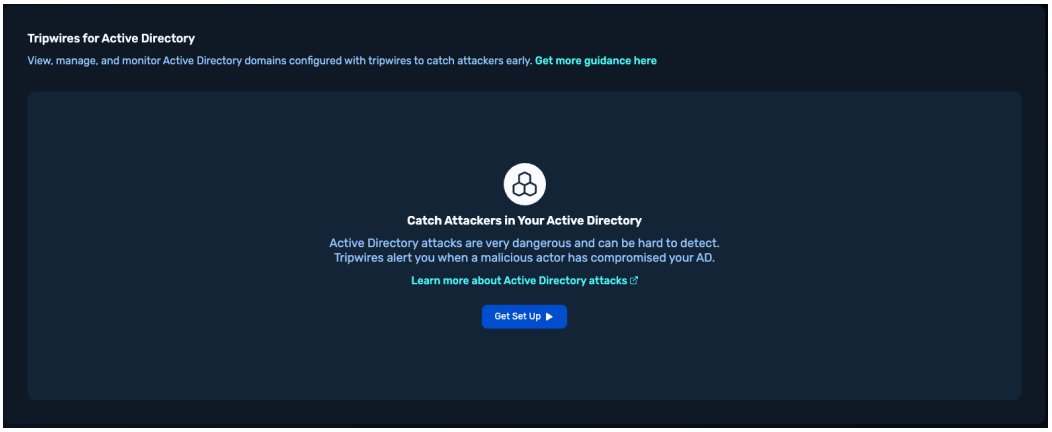
AD DOMAIN MANAGEMENT

The **AD Domain Management** section in the NodeZero Portal allows you to view, add, and manage Active Directory domains configured for tripwire monitoring. This is where you can review the status of each domain, check agent connectivity, and initiate management actions.

The AD Domain Management page when no domains have been added.

Adding a New AD Domain

When no domains have been added yet, you'll see an empty state card introducing the feature and a call to action:



The empty state for AD Domain Management, with a "Get Set Up" button.

To add a new Active Directory domain for tripwire monitoring:

1. Click the **Get Set Up** or **Add Domain** button.
2. In the modal that appears, enter the required details:
   - **Domain Name**
   - **Domain Controller Hostname**
   - **Service Account Information**
3. Review the information and follow the prompts to complete the setup.

The Add Domain modal, where you provide domain and account details.

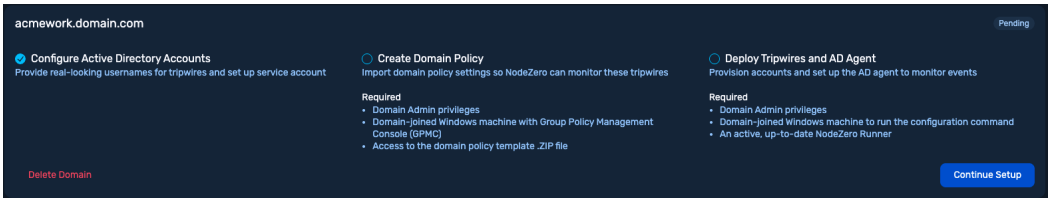
Once the domain is added, it will appear in the list of configured domains for monitoring and management.

Viewing and Managing Existing Domains

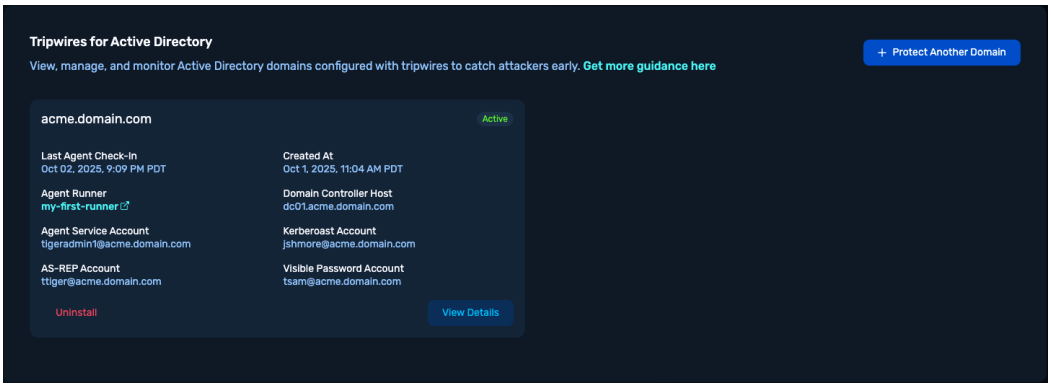
Once domains are added, you will see a list of all configured AD domains, along with their status and key details:

- **Domain Name**
- **Agent Status** (Active/Inactive)
- **Last Communication**
- **Actions** (View, Edit, Remove)

During setup, a domain will show a status of "Pending"



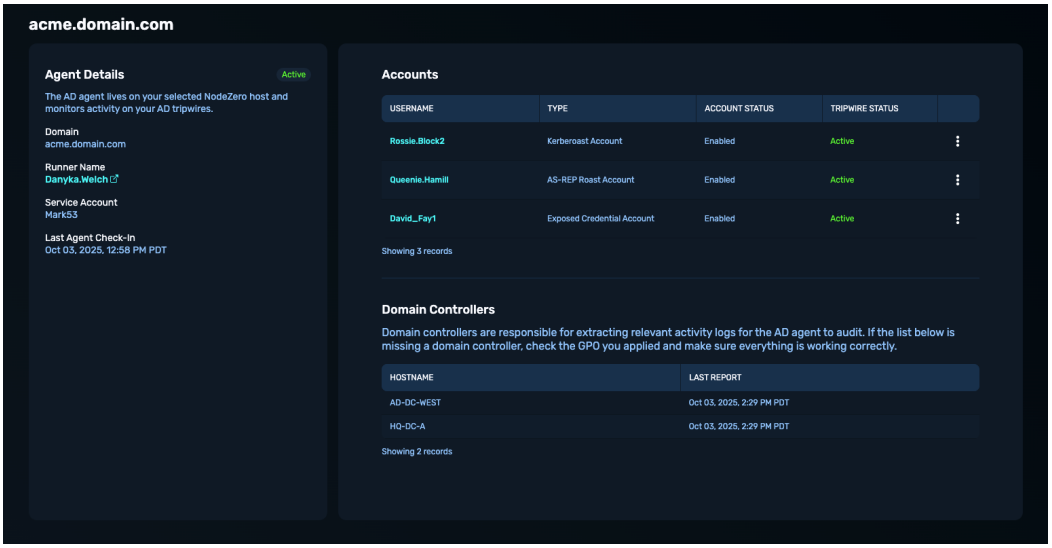
A fully configured domain will have a status of "Active"



Domain Details

Clicking on a domain name brings up detailed information, including:

- Domain configuration summary
- Linked tripwire accounts
- Agent installation and status
- Troubleshooting and management actions



ADDING ADDITIONAL DOMAINS

The "Protect Another Domain" button allows you to add and configure additional Active Directory domains for tripwire monitoring after the initial setup is complete.



AD Tripwires - Configuring Domain Policy

AD Domain Policy

An Active Directory domain policy which enables additional Kerberos logging events is required for detecting Kerberoasting and AS-REP Roasting attacks against **AD Tripwires**. This policy specifically enables enhanced audit logging for authentication events and creates a scheduled task on Domain Controllers to run the `Event Collector`.

Note: The AD Tripwires domain policy does **not** enable "Audit account management" events. Only authentication-related events required for attack detection are enabled.

The policy enables Windows Security Events 4625, 4768, 4769, 4771, and 4776 to provide the necessary audit trail for attack detection. Note that some of these events can generate significant log volume in busy environments.

DOWNLOAD DOMAIN POLICY TEMPLATE PACKAGE

1. During the **AD Tripwires** setup in the Horizon3 Portal, you will be provided a domain policy package.
2. Locate the **Download & Configure Domain Policy** card
3. Click the **Download** button to obtain the domain policy template ZIP file
4. Save the ZIP file to a location accessible from your domain controller or management workstation

PREREQUISITES

Before proceeding with domain policy configuration, ensure you have:

1. **Domain policy template ZIP file** (downloaded from the Horizon3 Portal)
2. **Domain Admin privileges** on the target Active Directory domain
3. **Windows Server or workstation** with Group Policy Management Console (GPMC) installed
4. Access to a domain controller or management workstation joined to the target domain

EXTRACT DOMAIN POLICY TEMPLATE

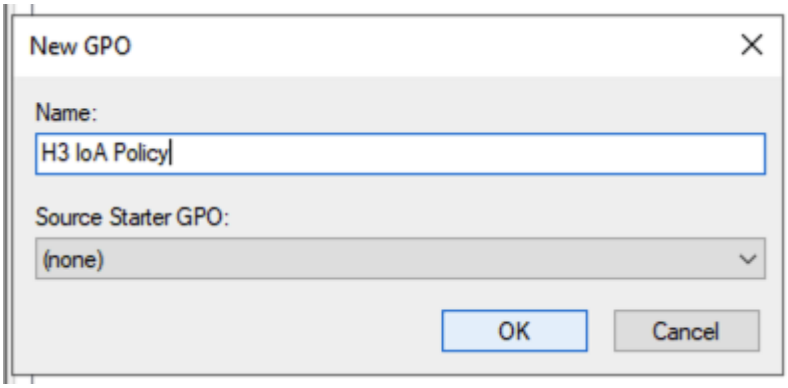
1. Navigate to the location where you downloaded the domain policy template ZIP file
2. Right-click on the ZIP file and select **Extract All**
3. Choose a temporary directory for extraction (e.g., `C:\Temp\H3_Policy_Template`)
4. Click **Extract** to extract the domain policy template files
5. Verify that the extracted folder contains the necessary Group Policy backup files including:

- `gpreport.xml`
- `manifest.xml`
- `Backup.xml`
- Additional policy configuration files

CREATE NEW GROUP POLICY OBJECT

1. Open Group Policy Management Console by pressing `Win`+`R`, typing `gpmc.msc`, and pressing `Enter`
2. In the left pane, expand your target domain
3. Right-click on **Group Policy Objects** and select **New** from the context menu

4. In the **New GPO** dialog, enter a meaningful name for the policy (e.g., "H3 IoA Policy")



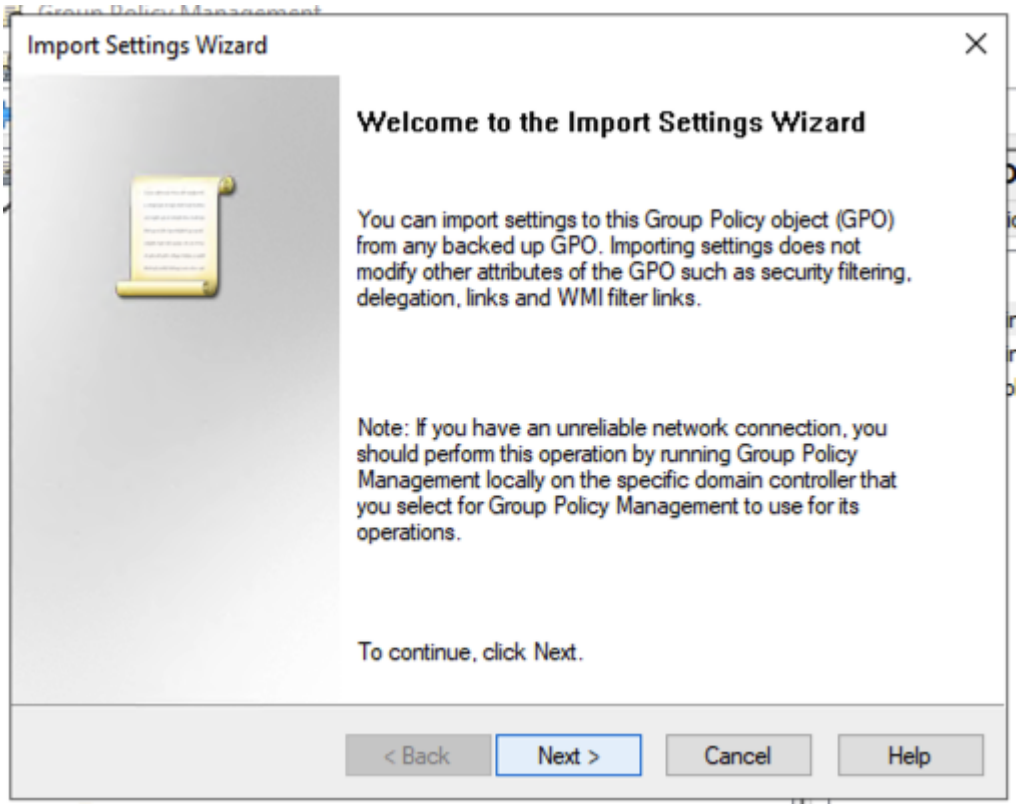
5. Click **OK** to create the new Group Policy Object

> **GPO Status**
>
> The newly created GPO is not yet linked to any organizational unit, so it will not affect any computers in your domain until you complete the linking process.

IMPORT DOMAIN POLICY SETTINGS

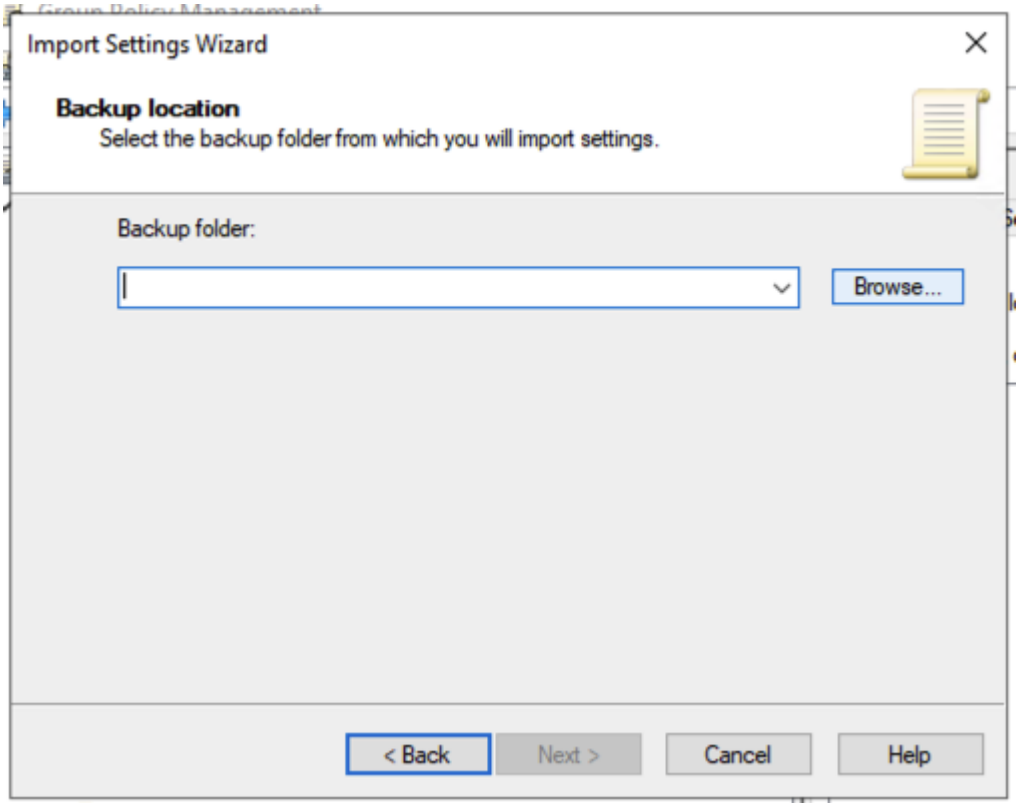
1. In the Group Policy Management Console, locate the newly created GPO under **Group Policy Objects**
2. Right-click on the GPO and select **Import Settings** from the context menu

3. The **Import Settings Wizard** will open. Click **Next** to begin the import process



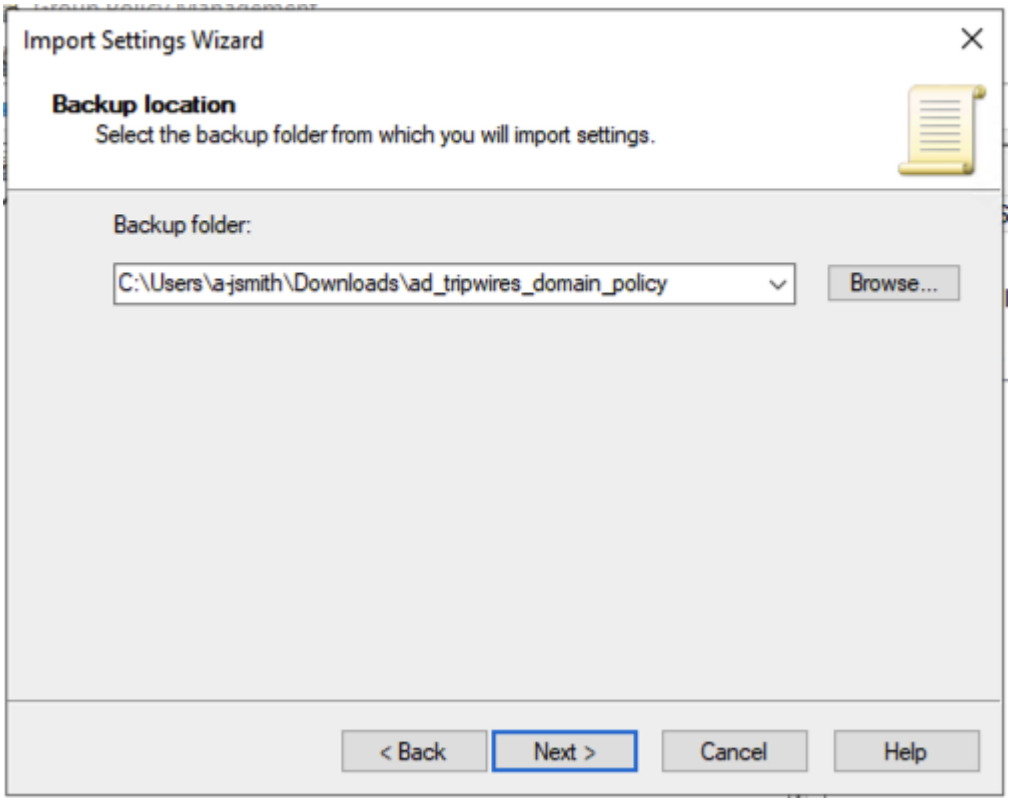
4. On the **Backup GPO** screen, click **Next** to continue

5. On the **Backup Location** screen, click **Browse** to select the folder containing the extracted domain policy template



6. Navigate to and select the folder where you extracted the domain policy template files

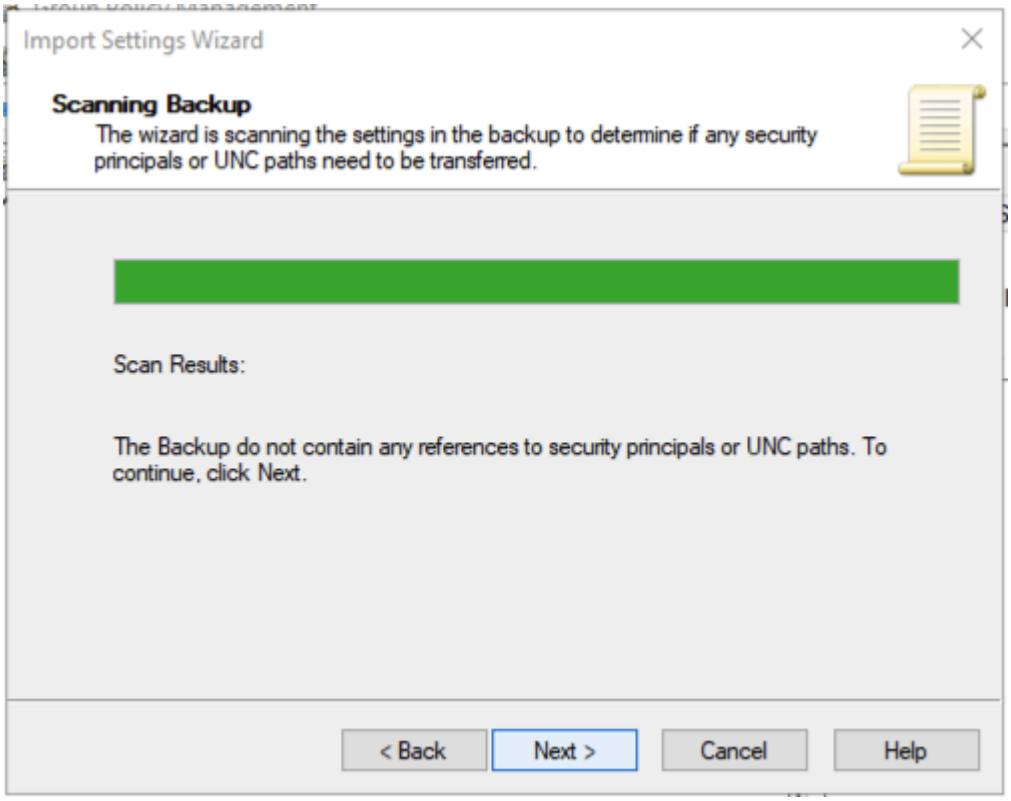
7. Select the appropriate GPO backup from the list and click **Next**



> **Troubleshooting Import**
>
> If you don't see any GPO backups listed, verify that you selected the correct folder containing the manifest files (`gpreport.xml`, `manifest.xml`, etc.).

8. Review the source GPO information

9. Choose how to handle security principals and click **Next**
10. Review the import summary and click **Next** to proceed with the import



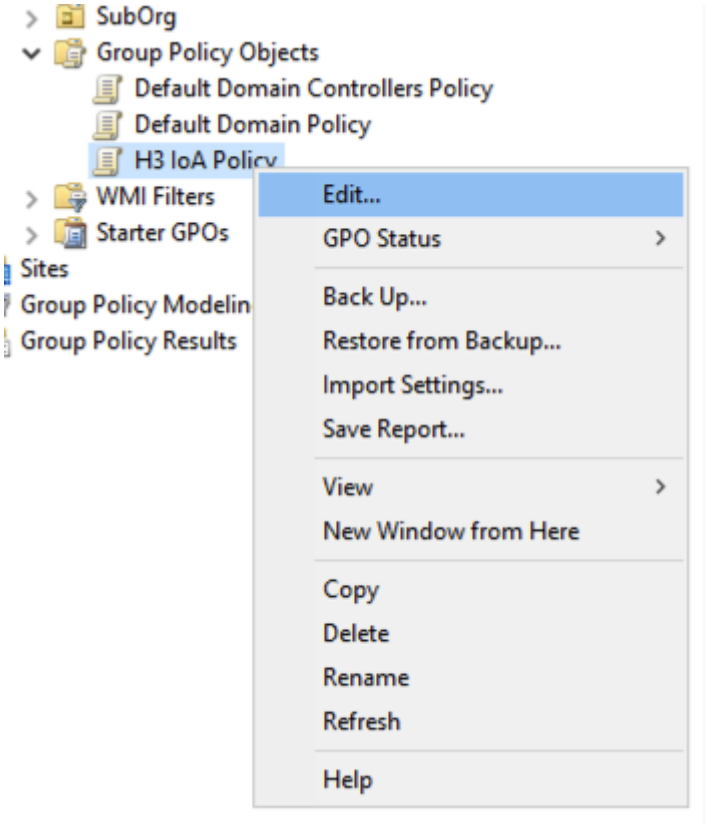
11. Wait for the import process to complete, then click **Finish**

Import Settings Wizard
Completing the Import Settings Wizard
You have successfully completed the Import Settings Wizard. The wizard will import settings from the following GPO.
Summary:
Backup location:
C:\Users\a-jsmith\Downloads\ad_tripwires_domain_policy
Backup GPO:
Name: H3 IoA Policy Template
Time Stamp: 6/6/2025 5:52:47 PM
Description:
To close this wizard and import settings, click Finish.
< Back
Finish
Cancel
Help

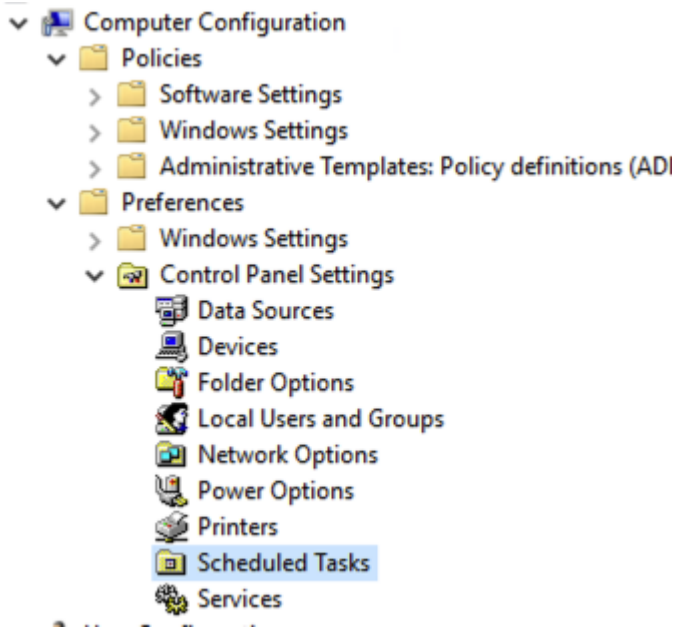
Import
Import progress:
Status:
GPO: H3 IoA Policy...Succeeded
OK
Cancel

REVIEW IMPORTED POLICY SETTINGS

1. In the Group Policy Management Console, right-click on the imported GPO and select **Edit**

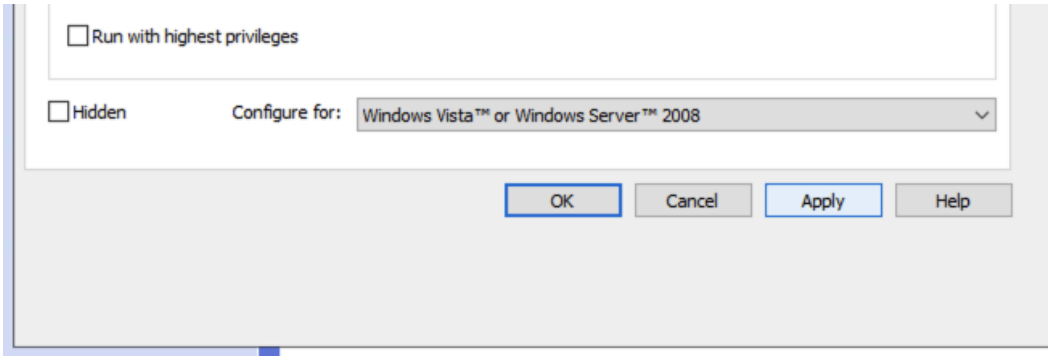


2. The **Group Policy Management Editor** will open. Navigate to **Computer Configuration → Preferences → Control Panel Settings → Scheduled Tasks**



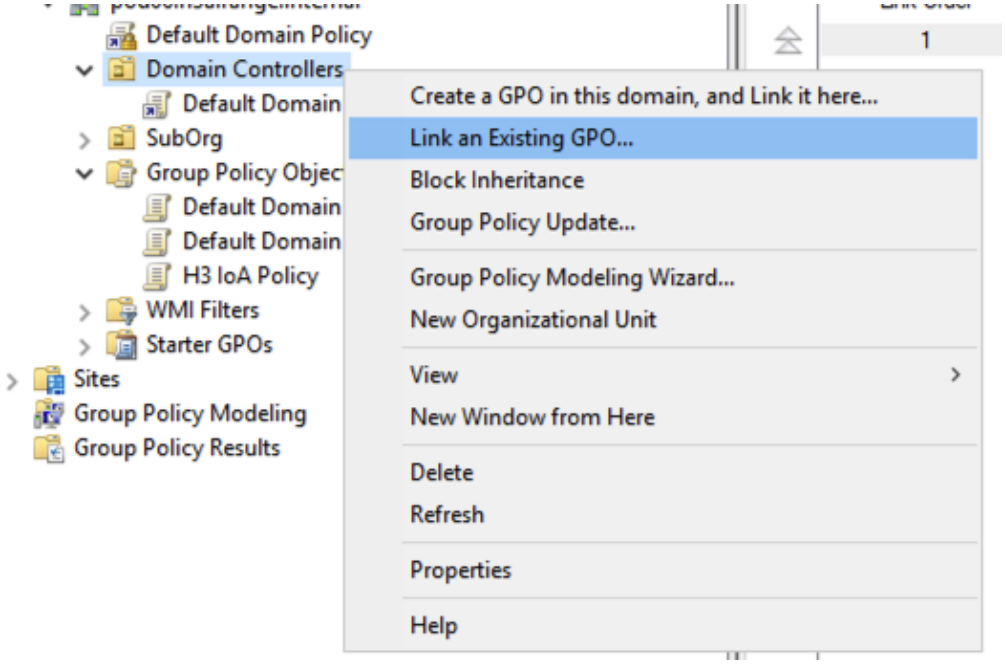
3. Verify that the imported policy contains the **IoA Collector** scheduled task configuration

4. Review the task settings to ensure they match your environment requirements
5. Close the **Group Policy Management Editor** when review is complete



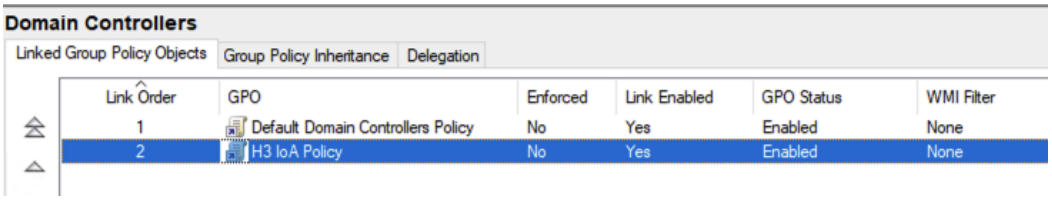
LINK GROUP POLICY OBJECT

1. In the **Group Policy Management Console**, navigate to the **Domain Controllers** organizational unit (or the appropriate OU where you want to apply the policy)
2. Right-click on the **Domain Controllers** OU and select **Link an Existing GPO**



3. In the **Select GPO** dialog, choose the H3 IoA Policy that you created and imported

4. Click **OK** to link the policy to the organizational unit
5. Verify that the policy appears in the **Linked Group Policy Objects** list for the OU



APPLY GROUP POLICY AND VERIFY

1. **Wait for group policy replication** to all domain controllers

   Typical Replication Timeframes

   Small domains (1-10 DCs): 15 minutes to 1 hour

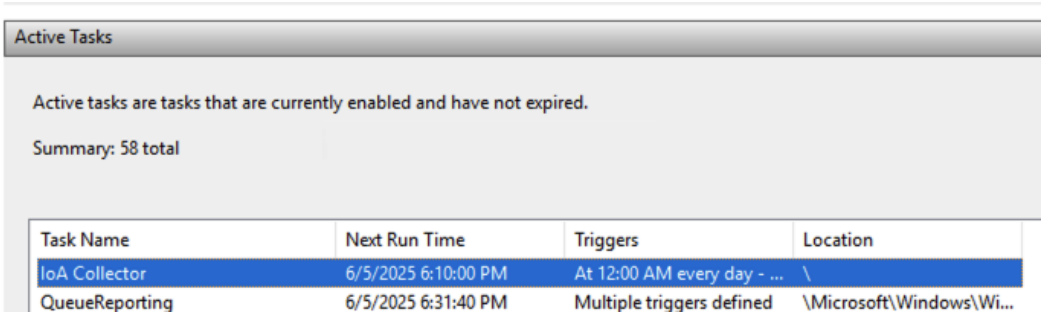
   Medium enterprises (10-50 DCs): 1-4 hours

   Large enterprises (50+ DCs): 2-8 hours

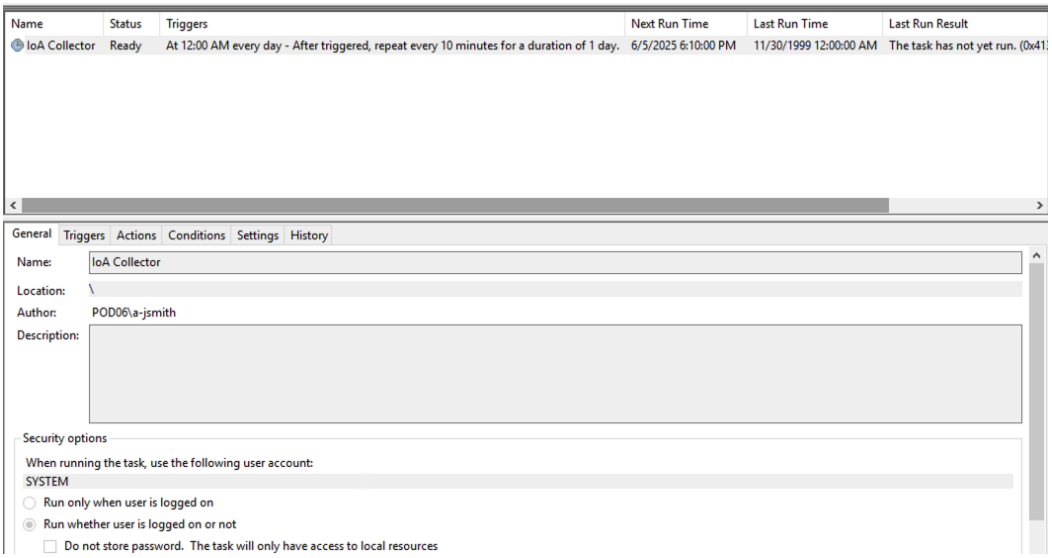
   Very large/global enterprises: 8-24 hours

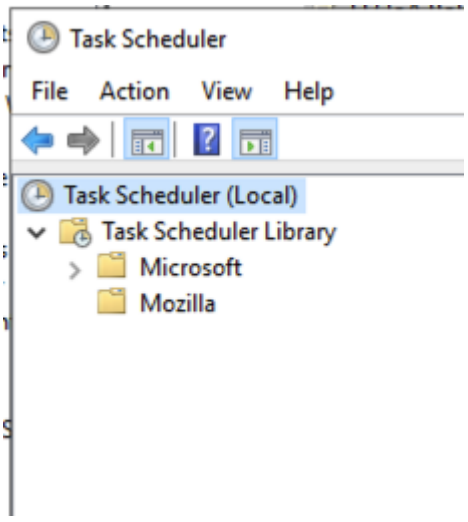
You can also run `gpupdate /force` on individual domain controllers to force an immediate Group Policy refresh.

2. **Verify scheduled task deployment** on domain controllers:
3. Open Windows Task Scheduler by pressing `Win`+`R`, typing `taskschd.msc`, and pressing `Enter`
4. Navigate to **Task Scheduler Library** in the left pane
5. Verify that the **IoA Collector** task appears in the task list



6. Right-click on the **IoA Collector** task and select **Properties** to review the task configuration





7. **Monitor task execution** to ensure the scheduled task runs successfully according to its configured schedule

NEXT STEPS

Once the domain policy has been successfully configured and applied:

1. **Provision AD accounts and event logging** using the provisioning process (see Getting Started guide)
2. **Verify `AD Agent` installation** and connectivity
3. **Test tripwire functionality** by performing controlled activities that should trigger alerts

For troubleshooting any issues during setup or operation, see the Troubleshooting guide.

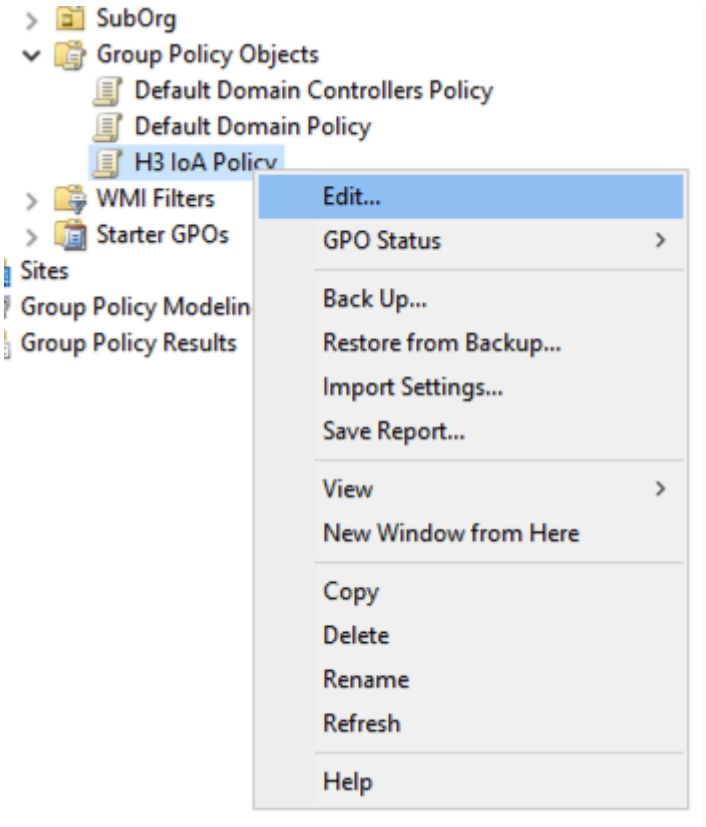
> **Important**
>
> The domain policy configuration is just one component of the **AD Tripwires** setup. You must also complete the AD account provisioning and agent installation steps for full functionality.

AD Tripwires - Removing Domain Policy

Note: Removing the Domain Policy does not fully remove AD Tripwires. Additional steps are required to clean up tripwire accounts, agent configuration, and related infrastructure.

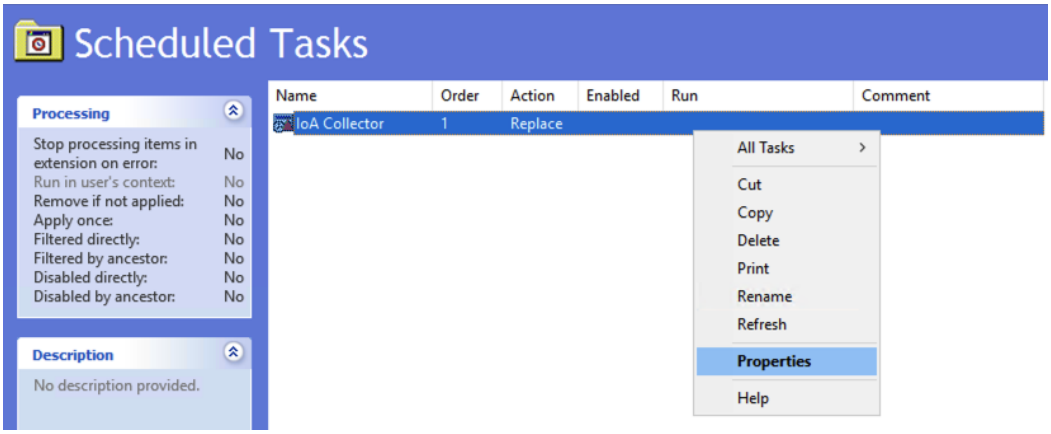
EDIT EXISTING POLICY TO CONFIGURE REMOVAL OF SCHEDULED TASK

1. Open Group Policy Management Console (`gpmc.msc`)
2. Locate the existing `H3 IoA Policy` object inside the `Group Policy Objects` container.
3. Right click on the policy object and click the **Edit** option from the context menu.

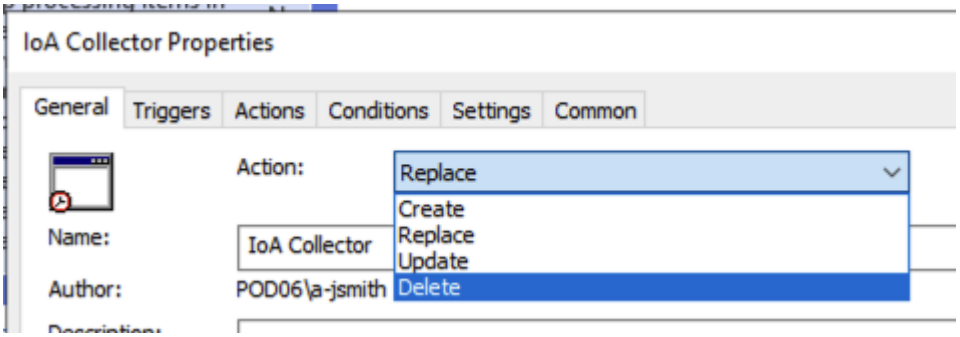


4. Once the `Group Policy Management Editor` opens, use the sidebar to navigate to `Computer Configuration` -> `Preferences` -> `Control Panel Settings` -> `Scheduled Tasks`

5. Right click on the scheduled task in the list. Click **Properties** from the context menu.



6. Once the Scheduled Task Properties Dialog window opens, navigate to the `General` tab if it isn't there already.
7. Change the `Action` dropdown from `Replace` to `Delete`.



8. Click **Apply** and then **OK**.
9. Close the `Group Policy Management Editor` window.

10. **ATTENTION:** Wait for group policy to replicate to all domain controllers

    Typical Timeframes

    Small domains (1-10 DCs): 15 minutes to 1 hour

    Medium enterprises (10-50 DCs): 1-4 hours

    Large enterprises (50+ DCs): 2-8 hours

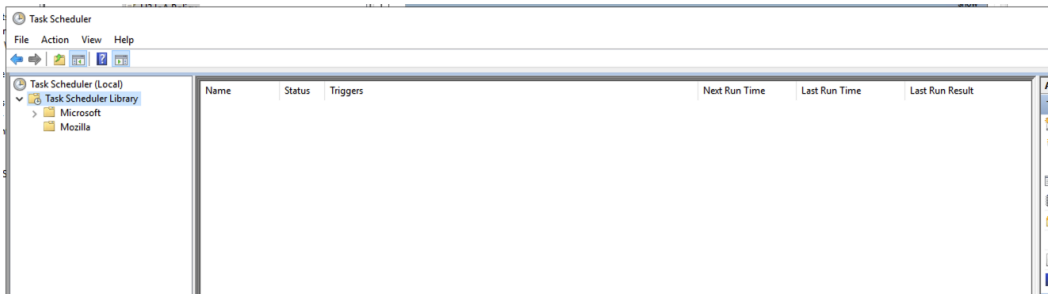
    Very large/global enterprises: 8-24 hours

    You can also run `gpupdate /force` on individual domain controllers to force an immediate Group Policy refresh.

1. Spot check domain controllers to verify that scheduled task has been removed

a. Open Windows Task Scheduler (`taskschd.msc`)

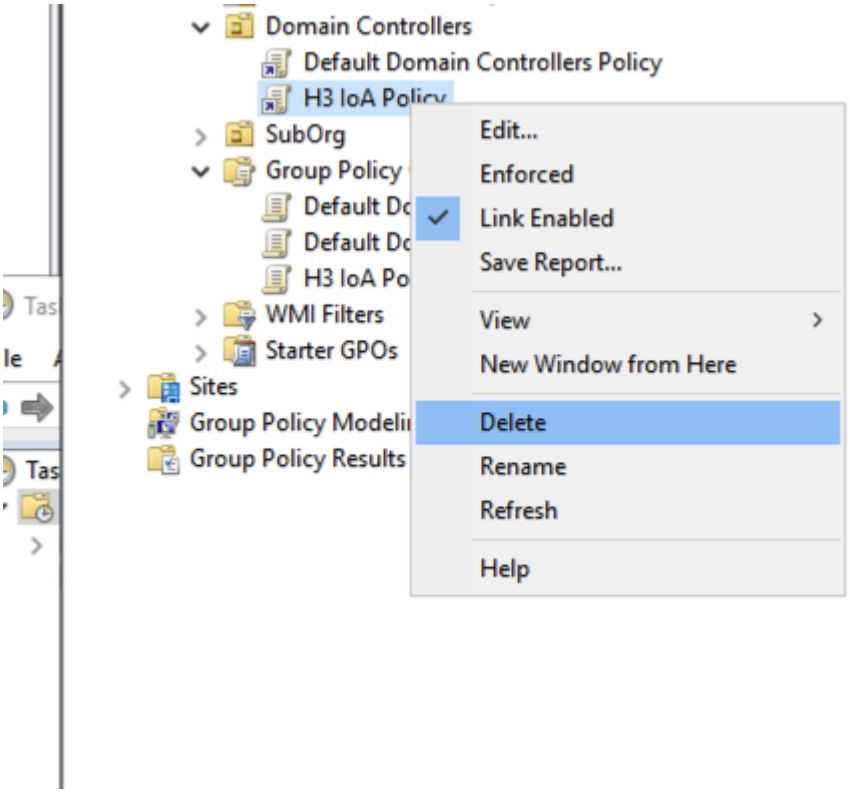
b. Verify that the `IoA Collector` task does not appear in the Task Scheduler Library.



REMOVE GROUP POLICY OBJECT

Once the GPO has replicated and removed the scheduled task from all domain controllers, the group policy itself can be unlinked and removed.

1. Open Group Policy Management Console (`gpmc.msc`)
2. Locate the `H3 IoA Policy` link under the `Domain Controllers` OU and right click on the link and select **Delete** from the context menu.



3. Locate the `H3 IoA Policy` object under the `Group Policy Objects` container. Right click on the policy object and select **Delete** from the context menu.

Group Policy Objects
Default Domain Controllers Policy
Default Domain Policy
WMI Filter
Sites
Edit...
GPO Status
Back Up...
Restore from Backup...
Import Settings...
Save Report...
View
New Window from Here
Copy
Delete
Rename
Refresh
Help

AD Tripwires — Disguising Tripwires

This guide describes optional, post-provision steps to help tripwire accounts created by the `AD Utility` to blend in with real users – without affecting detection. You will set only non-privileged, commonly populated attributes, so that tripwires maintain authenticity while remaining safe.

WHO THIS IS FOR

- **Audience**: AD admins performing manual touch-ups after the `AD Utility` provisions tripwire accounts.
- **Goal**: Apply common, low-risk attributes so that tripwires resemble typical user accounts, while preserving **AD Tripwires** behavior.

> **Enhanced realism**
>
> The utility already configures tripwire characteristics and generates natural logon artifacts. This guide focuses on "persona" details seen on real users.

WHEN TO USE THIS

- After `AD Utility` finishes creating tripwire accounts.
- Before broad visibility (e.g., before blue team reviews or attacker simulations).

PRINCIPLES

In general, set only non-privileged, commonly populated attributes. Match your org's naming, UPN, OU, and group conventions.

Password Changes

If required, you may change a tripwire user account password; it should not cause issues or pose any security risk as long as the new password meets or exceeds the strength of the existing 20-character complex password.

> **Critical: AD Tripwire service account password**
>
> Do not manually change the AD Tripwire service account password. It is managed by the utility/agent.

Maintain Critical Decoy Settings

Do not alter required decoy metadata, and do not disable required protocol settings. Such changes would cause related tripwire accounts to fail health checks, reducing detection coverage. Specific settings to leave unchanged:

- Kerberoastable: Do not remove all SPNs from the account.
- AS-REP roastable: Do not change the account's pre-authentication requirement, configured by the utility.
- Exposed credential: Do not change utility-set decoy metadata on the account.

> **If you're unsure about a change**
>
> If you're unsure whether modifying a specific attribute is safe, or whether it could impact tripwire detection or health checks: Contact Horizon3 Support through the NodeZero Portal, or contact your account team, before making the change.

RECOMMENDED MODIFICATIONS

- **Identity and contact**
- `displayName`, `givenName`, `sn`, `initials`
- `userPrincipalName` (UPN suffix must match org standard)
- `mail` (only if all users have one)
- `telephoneNumber`, `mobile`
- `physicalDeliveryOfficeName` (Office), `company`
- `title`, `department`
- **Org graph**
- `manager` (DN of a real manager used by peers)
- **Description**
- Exposed credential tripwire: Do not modify the description; preserve the utility-set decoy text.
- Other tripwire accounts: You may append normal descriptors if desired, but preserve any utility-set decoy metadata.
- **Group membership**
- Keep default `Domain Users` membership.
- Optionally add to low-priv distribution or departmental groups common to peers (no admin/sensitive groups).
- **SPNs (Kerberoastable tripwire only)**
- `servicePrincipalName` with a plausible entry (e.g., `HTTP/<service-host>.<domain>`) consistent with your environment. You may modify the SPN value(s) to match your naming standards, but the Kerberoastable account **must always have at least one SPN** configured; removing all SPNs disables Kerberoast detection for this account.

OPTIONAL MODIFICATIONS

- **Email system attributes**
- `proxyAddresses`, `mailNickname`
- **Profile/home settings**
- `homeDirectory`, `homeDrive`, `profilePath`, `scriptPath`
- **Office/geo metadata**
- `streetAddress`, `l` (city), `st` (state), `postalCode`, `c` / `co` (country), `countryCode`
- **Logon restrictions**
- `userWorkstations`, `logonHours` (only where broadly enforced)

PROHIBITED MODIFICATIONS

- **Account naming/identifier**
- Do not rename tripwire accounts after initial provisioning. The AD Agent relies on the original identifier for health checks and detections.
- **AD Tripwire service account credentials**
- Do not manually change the service account password; it is managed by the utility/agent.
- **Privilege-bearing or unusual account control flags**
- `PasswordNeverExpires`, `SmartcardRequired`, `AccountNotDelegated`, `TrustedForDelegation`, `UseDESKeyOnly`, etc.
- Exception: AS-REP decoy requires `DoesNotRequirePreAuth` (already handled by the utility).
- **Mailbox attributes** if no mailbox will exist (inconsistent state can look suspicious).

OU PLACEMENT

- Place accounts in the same OU as typical users for your org (not just `CN=Users`) if that matches real provisioning practices.

- Avoid special “service” or “quarantine” OUs unless that’s normal for the persona you’re mimicking.

STEP-BY-STEP PROCEDURE

1) **Identify the target persona for each tripwire**

- `displayName`, title, department, office, phone, UPN, manager DN, baseline groups, OU.

2) **Set identity and contact attributes**

```
# Identity and contact (example placeholders)
Set-ADUser -Identity <user> \
  -GivenName "<GivenName>" \
  -Surname "<Surname>" \
  -DisplayName "<Display Name>" \
  -Title "<Job Title>" \
  -Department "<Department>" \
  -Company "<Company>" \
  -Office "<Office>" \
  -EmailAddress "<user>@<your-domain>" \
  -OfficePhone "<phone>"
```

3) **Align UPN and manager**

```
# UPN and manager
Set-ADUser -Identity <user> \
  -UserPrincipalName "<user>@<your-domain>" \
  -Manager "<manager distinguishedName>"
```

4) **Add to baseline low-priv groups**

```
# Example distribution/department group
Add-ADGroupMember -Identity "<distribution-or-dept-group>" -Members <user>
```

5) **(Kerberoast tripwire only) Add plausible SPN**

```
# Add SPN consistent with your environment
Set-ADUser -Identity <service-account> -ServicePrincipalNames @{Add="HTTP/<service-host>.<domain>"}
```

6) **(Optional) Set environment-dependent attributes**

```
# Only if your org uses these widely
Set-ADUser -Identity <user> -Add @{
  proxyAddresses = "SMTP:<user>@<your-domain>"
} -ErrorAction SilentlyContinue
```

7) **Move to standard user OU (if applicable)**

```
# Move user to the OU where typical users reside
Get-ADUser <user> | Move-ADObject -TargetPath "<OU path for typical users>"
```

VALIDATION CHECKLIST

- User appears in the same OU as peers.
- UPN matches org suffix and naming convention.
- Persona attributes (title, department, office) look consistent with similar users.
- Group membership limited to baseline/non-priv groups.
- For Kerberoast tripwire, SPN looks plausible and consistent.
- No privileged account control flags were added.

QUICK AUDIT COMMANDS

```
# Review key attributes
Get-ADUser <user> -Properties DisplayName,Title,Department,Office,UserPrincipalName,Manager,mail,proxyAddresses | Format-List

# Review group memberships
Get-ADPrincipalGroupMembership <user> | Select-Object Name | Sort-Object Name

# Review OU location
(Get-ADUser <user>).DistinguishedName
```

NOTES ON AD TRIPWIRES BEHAVIOR

- The utility triggers natural updates to logon metadata (e.g., `lastLogon` / `lastLogonTimestamp`) via legitimate authentications; these values update automatically as part of normal operations.
- For AS-REP tripwire, the utility sets `DoesNotRequirePreAuth` as designed; do not undo this.

SEE ALSO

- Overview
- Getting Started
- Configuring Domain Policy
- Troubleshooting

Updating AD Tripwires System

We continuously improve the AD Tripwires System with new features, security enhancements, and bug fixes. This guide explains how to keep your system up to date.

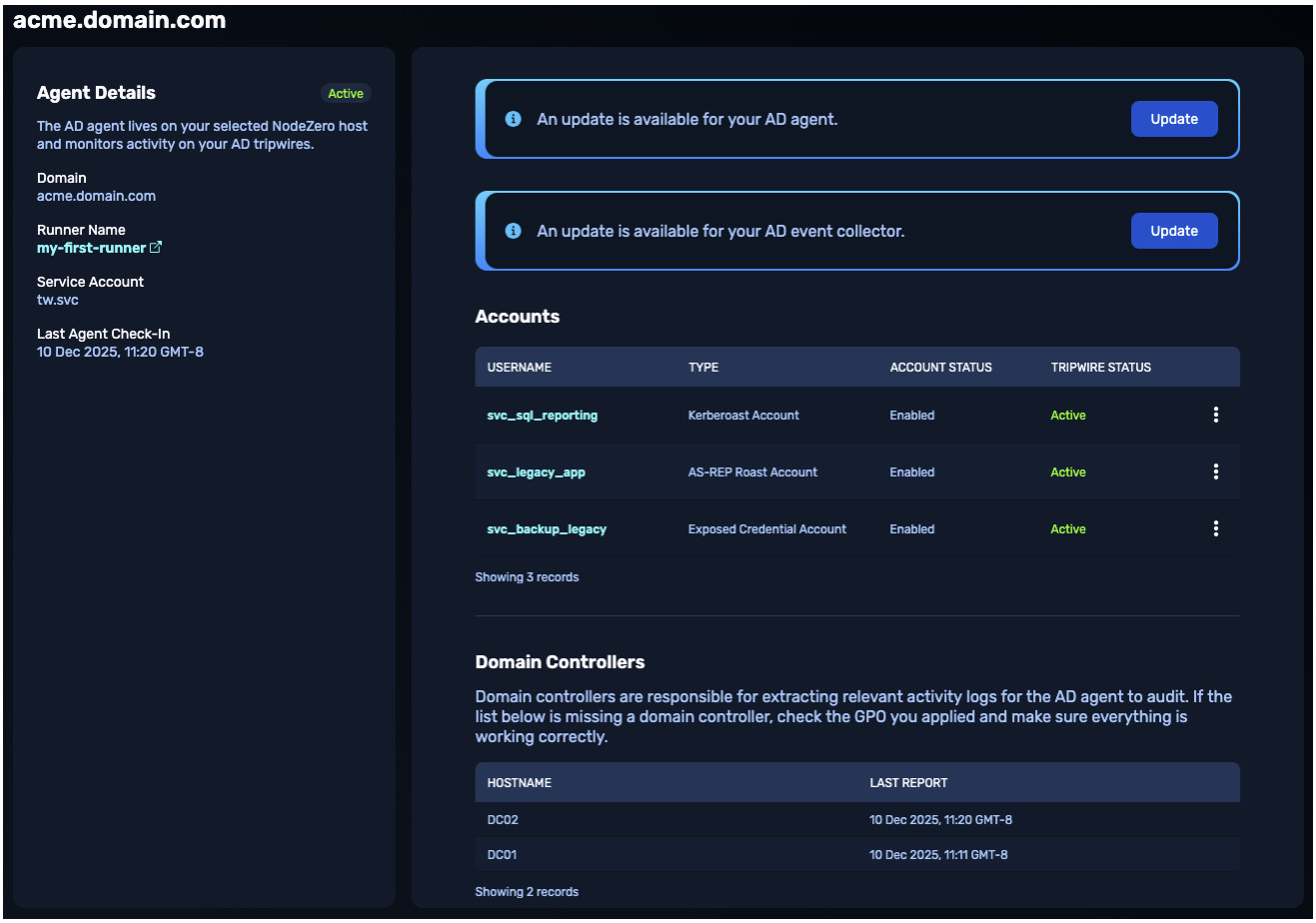
UPDATE NOTIFICATIONS

You'll be notified when updates are available through:

- Email notifications sent to all users with Tripwires access
- A banner displayed on the AD Tripwires domain details page

UPDATE STEPS

1. Navigate to the AD Tripwires domain details page
2. Click on the `Update` button for the tripwire component requiring an update



AD Event Collector Update

- Copy and run the command provided on the update page with domain admin credentials on any domain-joined Windows host with internet connectivity
- You will receive confirmation from the command in your terminal or Powershell window

AD Agent Update

- Click the Update button to trigger the update process
- You will be taken to the NodeZero Runners page; here you can view the progress of the `install-ad-agent` command

3. Update banners will disappear when updated components have been detected by NodeZero. This may take up to 20 minutes.

KEEP YOUR NODEZERO RUNNER UP TO DATE

The AD Tripwires System requires an up-to-date NodeZero Runner to function properly. To update your Runner:

1. Navigate to the Runners page in your NodeZero console
2. Locate your Runner in the list
3. Click the dropdown menu next to the Runner
4. Select "Upgrade Runner"

TROUBLESHOOTING

If you encounter issues during the update process, see the AD Tripwires Troubleshooting

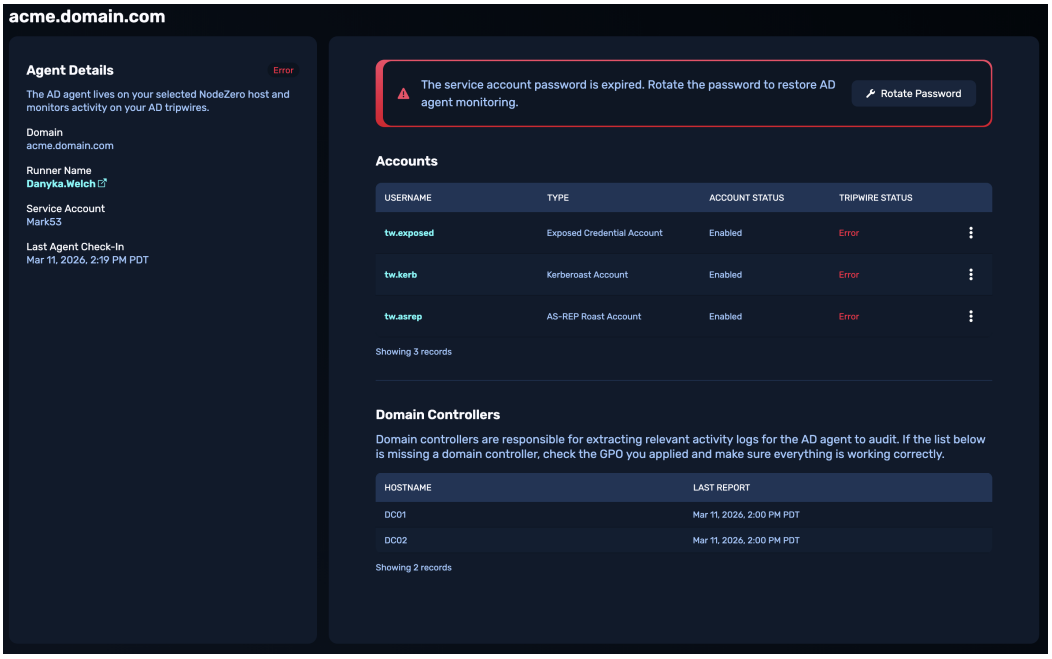
AD Tripwires – Rotating the Service Account Password

The AD Tripwires service account password might expire or be scheduled for rotation by your domain's password policy. When this happens, the AD Agent loses the ability to communicate with your domain, and monitoring is suspended until the passwords are rotated and the AD Agent is updated with the new credentials.

WHEN TO ROTATE

You will see the following error on the AD Tripwires domain index:

```
The service account password is expired. Rotate the password to restore AD agent monitoring.
```



Error indicator for password rotation

STEPS TO ROTATE

> **Runner must be active**
>
> Ensure that the NodeZero Runner used during installation is active before proceeding. It will be used to cache the new credentials on the NodeZero host and restart the AD Agent.

1. On the AD Tripwires domain index, click the **Rotate Password** button for the affected domain
2. Copy the PowerShell command provided
3. Run the command in your AD environment with domain admin credentials on any domain-joined Windows host with internet connectivity — this will rotate all AD Tripwires passwords in your domain
4. Once the passwords are rotated, the Runner will automatically cache the new service account credentials and restart the AD Agent
5. The error will clear once the AD Agent reconnects — this may take up to 5 minutes

4.7.4 **AD Tripwires** - How It Works

AD Tripwires provides early attack detection in your Active Directory environment using strategically placed decoy accounts. This document addresses key architectural aspects that security teams typically evaluate.

Core Concept

AD Tripwires deploys three types of decoy accounts that attract common Active Directory attack techniques:

```
graph LR
    attacker@{ shape: circle, label: "Attacker"}
    attacker-->A[Exposed Credential Account]
    attacker-->B[Kerberoastable Account]
    attacker-->C[AS-REP Roastable Account]
    A --> detection[Attack Detection]
    B --> detection
    C --> detection
    detection --> alerts[Security Alerts]
```

Why This Works: Legitimate users and applications have no reason to interact with these accounts. Any activity indicates reconnaissance or active compromise. Tripwire accounts are configured with long, complex, random passwords that are extremely unlikely to be cracked, even with dedicated hardware. These accounts are not used for legitimate logins and will not generate events for successful authentication—only failed attempts or suspicious activity are detected.

Security Architecture

MINIMAL PRIVILEGED ACCESS

Service Account Permissions:

- Read-only access to `SYSVOL` EventAnalytics directory
- No privileged access to Active Directory objects
- Cannot modify domain configuration or user accounts

AD Agent Isolation:

- Runs on existing NodeZero Runner (no additional infrastructure)
- Outbound HTTPS only (no inbound network access required)
- Processes only attack metadata, never actual credentials

DATA PROTECTION

What Gets Monitored:

- Authentication attempts against tripwire accounts
- Kerberos ticket requests for decoy service accounts
- Account enumeration activities

What Stays Private:

- No access to legitimate user credentials
- No monitoring of production user activities
- No collection of business data or communications

DEPLOYMENT FOOTPRINT

Domain Controllers:

- Single scheduled task (runs every 10 minutes)
- Reads existing Windows Security logs
- Writes minimal event metadata to `SYSVOL`

Active Directory Changes:

- 3 decoy user accounts (customer defined usernames)
- 1 service account for monitoring access
- 1 directory in `SYSVOL` for event and config storage

How Detection Works

EVENT COLLECTION PROCESS

```
graph LR

    logs@{ shape: rectangle, label: "Windows\nSecurity Logs"}
    logs --> task@{ shape: rectangle, label: "Scheduled\nTask"}
    task --> scan@{ shape: rectangle, label: "Scan for\nTripwire Events"}
    scan --> events@{ shape: rectangle, label: "Write\nEvent Metadata"}
    events --> sysvol@{ shape: rectangle, label: "EventAnalytics\nDirectory"}
    sysvol --> agent@{ shape: rectangle, label: "AD\nAgent"}
    agent --> portal@{ shape: rectangle, label: "Horizon3\nportal"}
```

Key Points:

- Uses existing Windows logging (no additional log sources)
- Only processes events related to tripwire accounts
- Event data contains timestamps and attack indicators, not sensitive information

WINDOWS SECURITY EVENTS

AD Tripwires monitors account-specific security events that are enabled by the IoA Domain Policy.

> **Event scope and interactions**
>
> While specific events are associated with each tripwire type (e.g., TGS requests for Kerberoastable accounts, AS-REP responses for AS-REP Roastable accounts, failed logons for exposed credentials), the system evaluates any Windows Security event that involves a tripwire account. Depending on your domain's audit configuration, these interactions may generate tripwire events. Because tripwire accounts should never be used legitimately, any interaction is treated as potentially malicious.

Event Coverage:

- **4625: Failed logon attempt (Kerberos or NTLM)**
- **Detection Use**: Detects failed login attempts to tripwire accounts, such as password spraying or brute force attacks.
- **Logging Impact**: Can be frequent in large environments; filter for tripwire accounts to reduce noise.
- Microsoft Docs
- **4768: Kerberos authentication ticket (TGT) was requested**
- **Detection Use**: Detects AS-REP roasting attempts against tripwire accounts with pre-authentication disabled.
- **Logging Impact**: Can increase log volume; filter for tripwire accounts.
- Microsoft Docs
- **4769: Kerberos service ticket (TGS) was requested**
- **Detection Use**: Detects Kerberoasting attempts by monitoring TGS requests for tripwire service accounts.
- **Logging Impact**: Can increase log volume; filter for tripwire accounts.
- Microsoft Docs
- **4771: Kerberos pre-authentication failed**
- **Detection Use**: Identifies failed Kerberos authentication attempts, including brute force or password guessing against tripwire accounts.
- **Logging Impact**: Moderate; filter for tripwire accounts.
- Microsoft Docs
- **4776: NTLM authentication failed (NTLM credential validation failure)**
- **Detection Use**: Detects failed NTLM authentication attempts to tripwire accounts, such as brute force or password guessing using NTLM.
- **Logging Impact**: Moderate; filter for tripwire accounts.
- Microsoft Docs

Important Considerations: - Events 4768 and 4769 can significantly increase log volume in busy environments - May impact log storage and SIEM performance - Consider log retention policies and filtering strategies when enabling these events - The Event Collector only processes events related to tripwire accounts, reducing the data volume sent to the AD Agent

ATTACK DETECTION TYPES

Attack Type	What Triggers Detection	Why It Matters
Credential Harvesting	Login attempts to exposed credential accounts	Indicates password spraying or credential stuffing attacks
Kerberoasting	TGS requests for tripwire service accounts	Detects attempts to extract service account hashes for offline cracking
AS-REP Roasting	Authentication requests to pre-auth disabled accounts	Identifies reconnaissance for accounts with weak Kerberos configuration

Network and Infrastructure Requirements

CONNECTIVITY

- **Outbound Only**: `AD Agent` requires HTTPS access to Horizon3 platform
- **No Inbound Access**: No firewall changes or external access to your domain
- **Standard SYSVOL**: Uses existing domain controller `SYSVOL` share

PERFORMANCE IMPACT

- **Minimal Resource Usage**: Event collection runs for seconds every 10 minutes

- **Low Network Overhead**: Only attack metadata transmitted (typically < 1KB per event)
- **No Production Impact**: Monitoring operates independently of business applications

Operational Considerations

MAINTENANCE REQUIREMENTS

- **Self-Contained**: No ongoing configuration changes required
- **Automatic Updates**: Event collection logic updates via standard Group Policy
- **Standard AD Tools**: Managed using existing Active Directory administration tools

AUDIT AND COMPLIANCE

- **Transparent Operation**: All components visible in standard AD management consoles
- **Audit Trail**: All system activities logged in Windows Event Logs
- **Removable**: Complete removal possible through standard AD administrative procedures

FALSE POSITIVE PREVENTION

- **Targeted Monitoring**: Only monitors explicitly created tripwire accounts
- **No Production User Impact**: Legitimate user activities never trigger alerts
- **Clear Account Identification**: Tripwire accounts use customer-defined usernames and are tracked via Portal

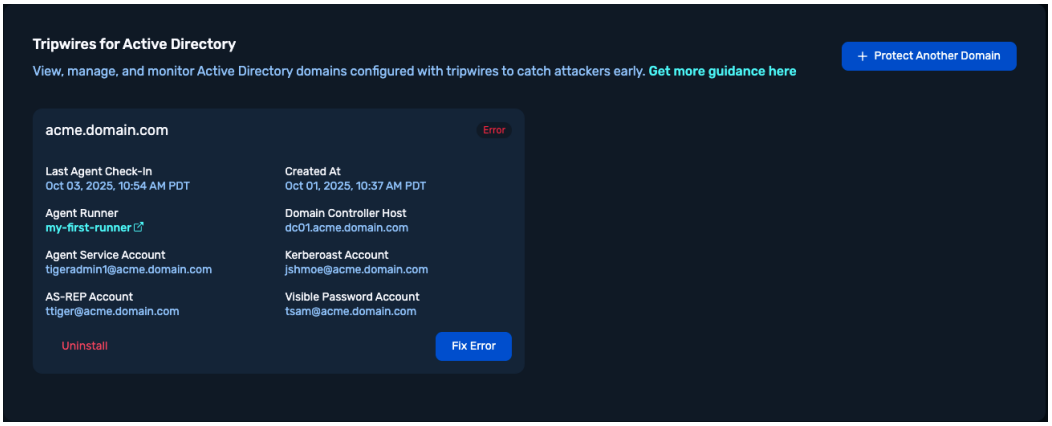
4.7.5 **AD Tripwires** – Troubleshooting

This guide helps resolve common issues encountered during **AD Tripwires** setup and operation.

> **Event scope and interactions**
>
> Although detection focuses on specific event types for each tripwire, the system evaluates any Windows Security event that involves a tripwire account. Depending on your domain's audit configuration, these interactions may generate tripwire events. Since tripwire accounts should never be used legitimately, any interaction is considered potentially malicious and may appear in the Portal.

Error States

If your AD Tripwire deployment encounters an error, the UI will display error states to help you diagnose and resolve problems.



Domain index showing an error state for a domain.

Error when a tripwire account already exists in Active Directory.

Setup Issues

DOMAIN POLICY IMPORT PROBLEMS

Problem: GPO import fails or settings don't appear correctly after import.

Solution:

1. Verify the extracted folder contains all required files (`gpreport.xml` , `manifest.xml` , `Backup.xml`)
2. Ensure you're selecting the correct folder path during import
3. Check that you have Domain Admin privileges
4. Try creating a new GPO and importing again

Problem: Scheduled task doesn't appear after GPO application.

Solution:

1. Force Group Policy update: Run `gpupdate /force` on domain controllers
2. Check GPO link: Verify the policy is linked to Domain Controllers OU
3. Verify task scheduler: Open `taskschd.msc` and look for "IoA Collector" task
4. Check event logs for Group Policy errors in Windows Event Viewer

AD AGENT CONNECTION ISSUES

Problem: `AD Agent` fails to start or connect to domain controllers.

Solution:

1. Verify service account permissions are correctly configured
2. Check network connectivity between NodeZero Runner and domain controller
3. Ensure the specified domain controller hostname is accessible
4. Verify the service account has necessary permissions to access `SYSVOL`

Problem: No events appearing in Horizon3 Portal.

Solution:

1. Verify tripwire accounts are created and configured correctly
2. Check that the scheduled task is running on domain controllers
3. Test connectivity to the EventAnalytics directory in `SYSVOL`
4. Review `AD Agent` logs on the NodeZero Runner

ACCOUNT PROVISIONING ISSUES

Problem: Failure to create tripwire accounts during provisioning.

Solution:

1. Ensure the account used for provisioning has the required permissions. See the Permission Requirements section in the Getting Started guide for detailed information
2. Verify the domain is accessible from the machine running the utility
3. Check for naming conflicts with existing accounts
4. Review Active Directory permissions and domain policies that might block account creation

Verification Steps

CONFIRM GPO IS APPLIED

1. Open Group Policy Management Console (`gpmc.msc`)
2. Navigate to Domain Controllers OU
3. Verify "H3 IoA Policy" appears in the linked GPOs list
4. Check GPO status shows as "Enabled"

VERIFY SCHEDULED TASK

1. On each domain controller, open Task Scheduler (`taskschd.msc`)
2. Look for "IoA Collector" task in Task Scheduler Library
3. Verify task is enabled and shows recent successful runs
4. Check task history for any errors

CONFIRM TRIPWIRE ACCOUNTS

1. Open Active Directory Users and Computers (`dsa.msc`)
2. Locate the created tripwire accounts
3. Verify account properties match expected configurations:
4. Exposed credential accounts have descriptions with embedded credentials
5. Kerberoastable accounts have Service Principal Names (SPNs) configured
6. AS-REP Roastable accounts have "Do not require Kerberos preauthentication" enabled

TEST AD AGENT STATUS

1. In Horizon3 Portal, navigate to **AD Tripwires** management
2. Verify `AD Agent` shows as "Active" status
3. Check last communication timestamp is recent
4. Review any error messages or warnings displayed

Common Error Messages

"ACCESS DENIED" DURING PROVISIONING

- **Cause**: Insufficient privileges for Active Directory operations or `SYSVOL` access

- **Solution**: Ensure the account has the following minimum permissions:
- **Active Directory**: Create/modify user accounts, set SPNs, and modify user properties
- **SYSVOL Access**: Full control permissions to create the EventAnalytics directory and set permissions
- **Alternative**: Use Domain Admin account which includes all necessary permissions

"DOMAIN CONTROLLER NOT ACCESSIBLE"

- **Cause**: Network connectivity or DNS resolution issues
- **Solution**: Verify hostname, check network connectivity, ensure DNS resolution works

"SYSVOL PATH NOT FOUND"

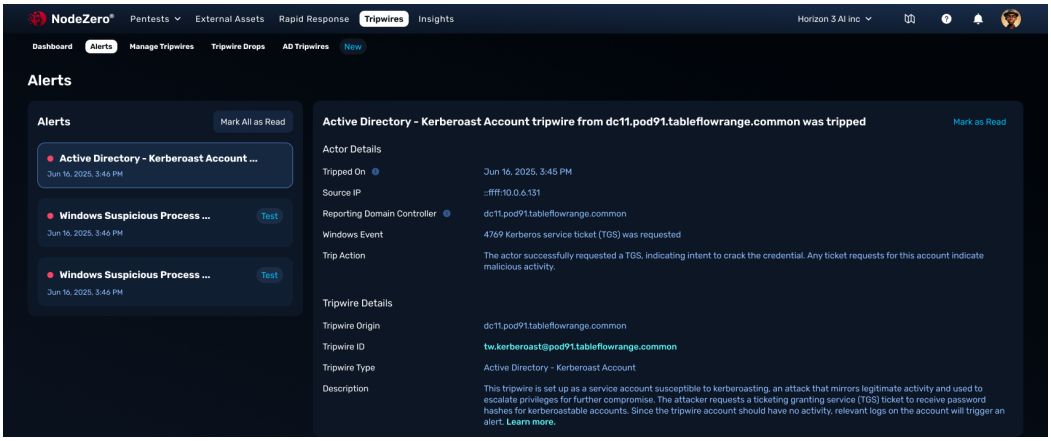
- **Cause**: Incorrect domain controller configuration or `SYSVOL` permissions
- **Solution**: Verify `SYSVOL` share is accessible and service account has read permissions

403 FORBIDDEN ERROR RUNNING UPDATE COMMAND

Cause: An outdated command was used to update the AD Tripwire **Solution**: - To obtain a new command, refresh the page in the NodeZero Portal - Generate a new command by clicking the "Generate Command" button - Rerun the command

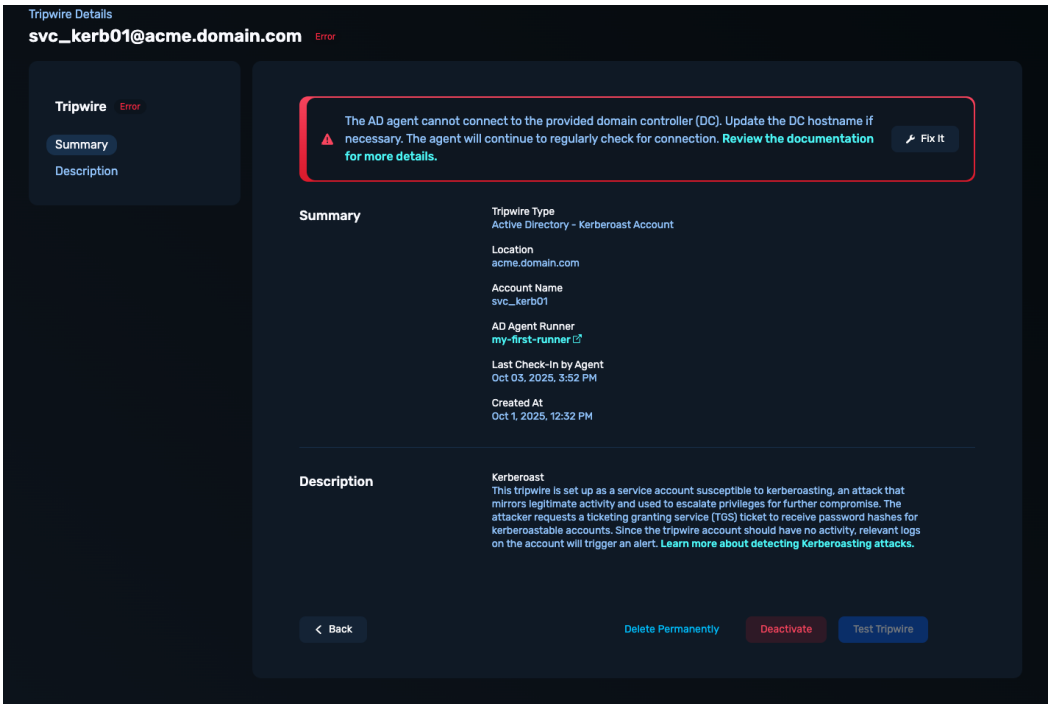
Example Alert and Tripwire Account Status

The Portal provides visual feedback for tripwire alerts and account status.



Example of tripwire alerts in the NodeZero Portal.

Tripwire account details and status.



Error state for a tripwire account.

Getting Additional Help

If you continue experiencing issues:

1. **Check Portal Logs**: Review any error messages in the Horizon3 Portal under **AD Tripwires** status
2. **Event Viewer**: Check Windows Event Logs on domain controllers for related errors
3. **Contact Support**: Provide detailed error messages and steps taken when contacting Horizon3 support

Running Windows Tools

1. Press `Win`+`R`
2. Type tool name in the `Open` text box
3. Click `OK` or press `Enter`

TOOL NAMES

- Group Policy Management Console: `gpmc.msc`
- Windows Task Scheduler: `taskschd.msc`
- Active Directory Users and Computers: `dsa.msc`

5. Insights

5.1 NodeZero Insights

This is an add-on feature, providing organization-wide visibility into your security posture – which helps you better prioritize, prove impact, and deliver strategic executive reports. This streamlined dashboard gives you aggregate and trend data from all tests you run with NodeZero autonomous pentesting. With this information, security leaders can prioritize mitigation efforts, prove impact of discovered weaknesses and exploits, and summarize a report to their executive stakeholders.

You can think of Insights as an aggregation of all your organization's Campaigns and their contained perspectives. As with campaigns, Insights deliver real value when you templatize and schedule your key pentests so that you can gather and monitor trends over time.

Features

- **Visualize Attack Surface Evolution**: Identify systemic blind spots and validate control effectiveness to ensure measurable progress.
- **Monitor Security Progress**: Track critical metrics, such as mean-time-to-remediation (MTTR), to quantify improvements over time.
- **Analyze Weakness Trends**: Deep dive into vulnerability types and severities, prioritizing fixes for maximum impact.
- **Proactively Identify Risks**: Address organization-wide issues like outdated security controls and misconfigurations to prevent future threats.

Screenshots

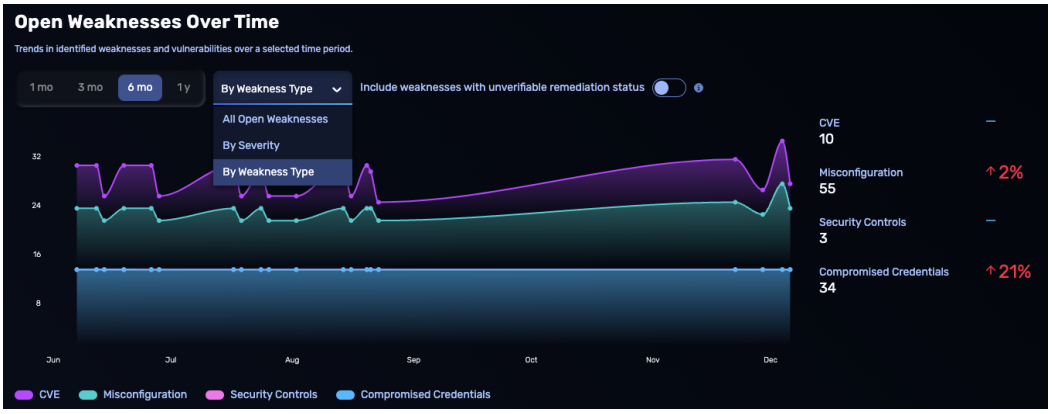
OPEN WEAKNESSES OVER TIME

This section shows the un-mitigated weaknesses across the entire organization as reported from NodeZero pentests. Each data point on the graph shows the total number of open weaknesses reported from all the pentest operations that were completed on that day. Users can visualize the data over **1 month**, **3 month**, **6 months**, and **1 year** periods. The list at the right of the graph summarizes counts of **Open Weaknesses**, **Assets Discovered**, and **Assets at Risk** as of today as well as the percentage change over the time period selected. An “Asset at Risk” is any unique asset that has at least one open weakness associated with it. Trend percentages are shown in red for metrics that are increasing but should be decreasing to improve the risk posture of the organization.



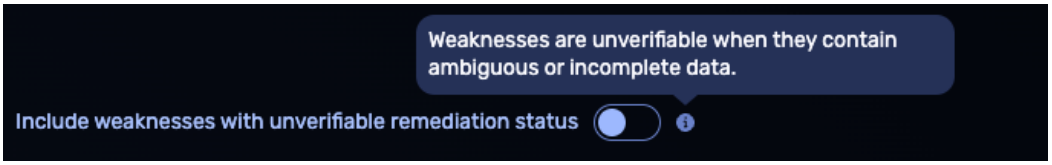
Users can see the open weaknesses separated by severity by changing the chart type in the selector. This graph shows the relative contribution of each weakness severity type to the overall open weakness trend along with the associated trends across the selected time period.

Users can also see the open weaknesses separated by type. This graph shows the relative contribution of each weakness type to the overall open weakness trend along with the associated trends across the selected time period.



UNVERIFIABLE WEAKNESSES

The majority of weakness types that are discovered and reported with NodeZero can be reported as mitigated once the user has taken some mitigation action and has run either a 1-click-verify operation or any incremental pentest operation. For the verifiable weakness categories, Insights will report open weaknesses as closed once the mitigation action is confirmed and tested. For the unverifiable categories, open weaknesses will continue to be reported as open in the Insights charts. Users have the option to include unverifiable open weaknesses in the Insights dashboard sections for sake of completeness when reporting trends.



ATTACK PATHS OVER TIME

This chart shows the open attack paths (or impacts) as reported as of each pentest operation over time. Users here can choose to visualize the data across **1 month**, **3 month**, **6 month**, and **1 year** time periods. The list to the right of the graph shows the top 5 most severe open attack path types with the most severe type listed at the top along with the associated trends. Trend percentages are shown in red for metrics that are increasing but should be decreasing to improve the risk posture of the organization.

REMEDIATION SUMMARY

In this section, users track distributions of key metrics that indicate the overall open weakness backlog and the remediation velocity across all weakness severities through mean-time-to-remediation (MTTR). **Open Weaknesses by Severity** shows the distribution of current open weaknesses by severity with the weaknesses that are in the un-verifiable category separated into their own histogram. **Open Weaknesses by Age** shows the distribution of the current open weakness backlog by how long the weakness has been open. For example, there could be weaknesses that were discovered within the last day or between the last day and 6 months ago. **Mean Time to Remediation** shows the average number of days across each severity type the organization took to remediate open weaknesses. This is visualized with a graph showing historical MTTR over a selectable time range. The current MTTR for each severity level is shown to the right.

CISA KEV FILTER

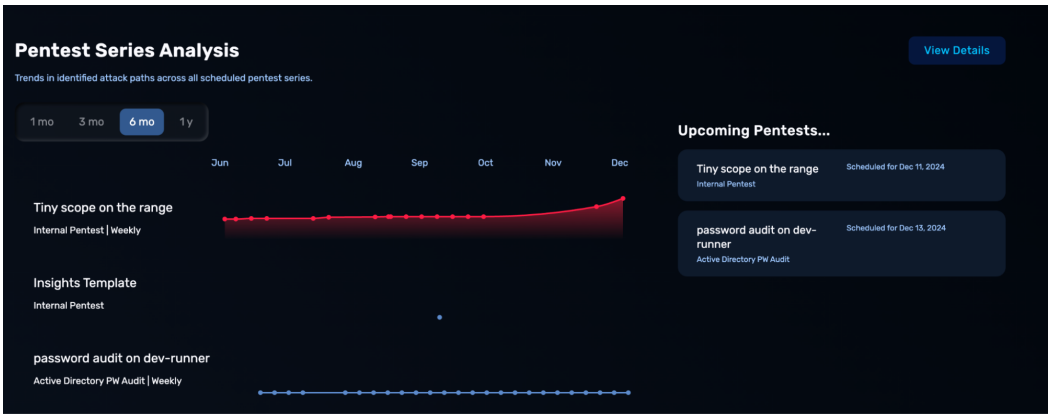
Users can track their remediation efforts for CISA Known Exploited Vulnerabilities (KEV) using the toggle switch near the top of this section.



When enabled, the MTTR graph and Open Weakness charts will reflect CISA KEV weaknesses only. This allows organizations to prioritize remediation efforts towards known exploits.

PENTEST SERIES ANALYSIS

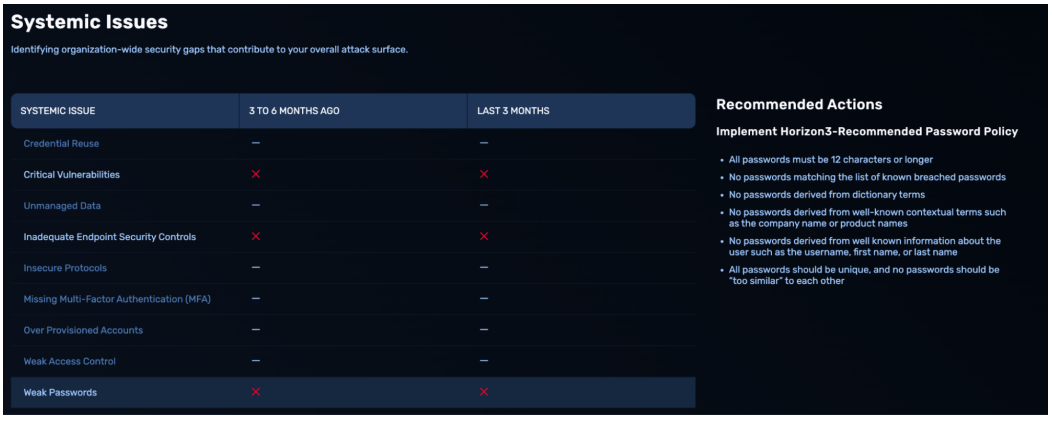
This section shows an overview of the active scheduled pentest templates the organization has configured along with the result attack paths after completion of each pentest operation in the series. Users can select **1 month**, **3 month**, **6 month**, and **1 year** time periods and each data point shows the number of open attack paths reported. Users can also see a list of upcoming scheduled pentest for each template.



To see further details on the numbers of each data point and information on other metrics like credentials compromised, users can click on the “View Details” button and then “View Trend” for a specific pentest series:

SYSTEMIC ISSUES

This section shows a 6-month trend of organization-wide security gaps that contribute to your overall attack surface. An “x” for each row denotes that the respective systemic issue is still open in the organization’s attack surface, while a “-” means that the issue was not detected. With two 3-month time periods that comprise the last 6 months, organizations can see a trend of whether company-wide policy changes and initiatives are working in closing these gaps in their risk posture. Users can click on each open issue to see remediation guidance as well.



6. Rapid Response

6.1 Rapid Response

NodeZero's Rapid Response feature helps you stay on top of critical, late-breaking N-day vulnerabilities and outrun attackers. The Horizon3.ai Attack Team proactively assesses your environment's assets for susceptibility to new vulnerabilities. Rapid Response then alerts you (via email and in the NodeZero Portal) if any of your assets are found to be exploitable, and provides targeted tests to quickly check assets' exploitability and remediation.

Not available on all license types

Rapid Response is available with certain license categories, or as an add-on feature. Contact your Horizon3.ai Account, Customer Success, or Support representative to add Rapid Response to NodeZero.

6.1.1 What Rapid Response Tells You

To help defenders identify and respond to real threats as soon as possible after their discovery, Rapid Response distills vulnerability noise into actionable signal. Rapid Response offers personalized responses to these questions:

- Which emerging vulnerabilities pose real exploitability risk?
- Am I exposed to that risk, and where?
- What do I need to do next, and how urgent is it?
- Did my fix work, and can I prove it?

These answers can dramatically cut down attackers' window of exploitation when a new vulnerability comes to light.

6.1.2 How Rapid Response Works

As a personalized, context-aware risk engine, Rapid Response maps internal and external exposures based on your historical penetration-test data. It sends alerts to subscribed members of your organization, and categorizes your assets with precise statuses (Exploitable, Potentially Relevant, Mitigated, Not Exploitable, Regressed), which update as you act. Embedded ticketing workflows turn each Rapid Response into a prioritized to-do list. Proof artifacts close the loop. Alerts resume if a vulnerability ever regresses.

The feature is backed by Horizon3.ai's team of attack engineers, who continually monitor and research new vulnerabilities. Using our knowledge of exploitation patterns – prioritizing internet-facing and easily leveraged vulnerabilities – we sift out risks with proven real-world exploitability and severe business impact. Then we develop production-safe tests to validate these risks.

Rapid Response routinely ships tests and alerts ahead of the CISA KEV catalog – the U.S. Cybersecurity and Infrastructure Security Agency’s industry-standard reference list of Known Exploited Vulnerabilities. This helps defenders focus their capacity on the most exploitable threats, including threats not yet publicly disclosed.

6.1.3 Rapid Response Components

Specific capabilities include:

- Asset-level vulnerability mapping, with scannable summaries linking to rich details.
- Clear indications of assets **not** at risk.
- Specific guidance about which tests and fixes to prioritize.
- Targeted tests to efficiently verify your remediations.
- Dashboards to track, validate, and demonstrate your progress.
- For opted-in environments, proactive asset checks the moment we get a working exploit – notifying you instantly if we confirm you have assets exposed.

Follow the links below to learn how to use each of these capabilities.

Notifications

Receive alerts about N-day vulnerabilities when they are found to affect your assets.

Portal View

Navigate the NodeZero Portal's overview of Rapid Response advisories and vulnerabilities.

Rapid Response Details

View metadata including relevant assets, references, response/mitigation timelines, and test availability.

Asset Details

View metadata about each asset's vulnerabilities, with remediation options and progress.

Targeted Tests

Assess your environment for the given vulnerability during Rapid Response testing.

Managing Rapid Response

Settings to opt in/out of features like proactive testing, with usage tips.

6.2 Notifications

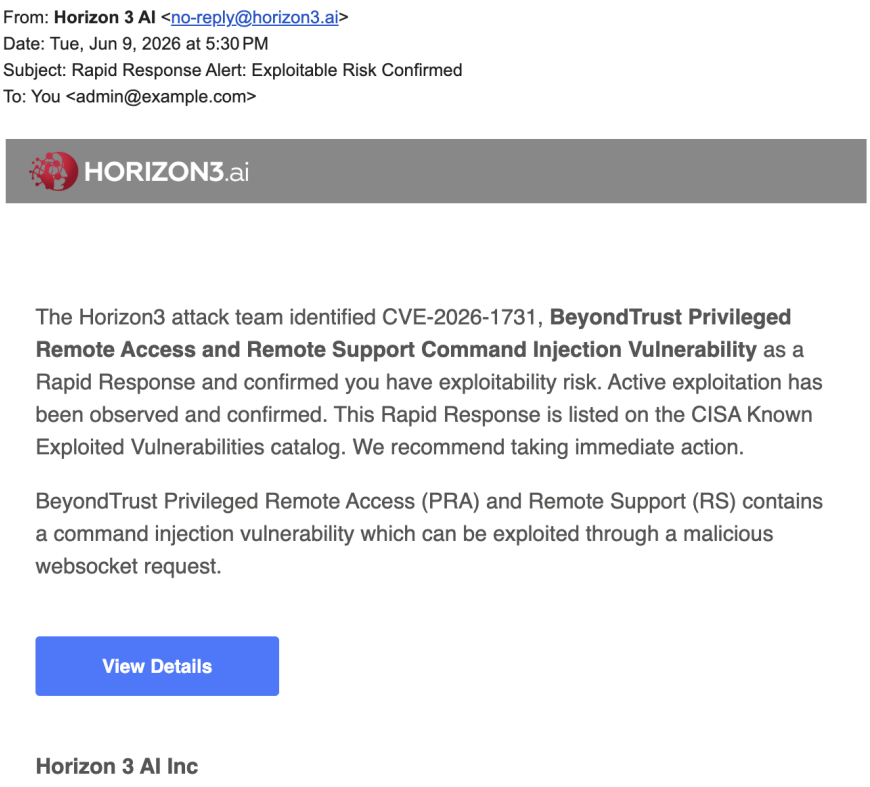
6.2.1 Rapid Response Notifications

With Rapid Response enabled, the Horizon3 AI Attack Team proactively assesses your assets for susceptibility to newly discovered N-day vulnerabilities, and sends email notifications to alert you if any of your assets are found to be exploitable. By default, all Org Admins and Users are subscribed to receive these notifications, but Read-only accounts are not subscribed.

There are two types of notifications:

- Confirmed exploitability – these are personalized alerts that you'll receive if Horizon3.ai finds your environment's assets to be at risk. Each email links to a Rapid Response Details page with rich data about the discovered vulnerability and guidelines on remediating it.
- A new Rapid Response test is available – this is a general email sent to all Rapid Response customers when Horizon3.ai publishes a new targeted test (which will also display the test as **Available** in the Rapid Response Portal view).

Here is an example of a confirmed-exploitability notification, including a button that links to Rapid Response details:



Here is an example of a new-test-available notification, including a button that links to Rapid Response details:

From: **Horizon 3 AI** <no-reply@horizon3.ai>
Date: Thu, Jun 18, 2026 at 2:11 PM
Subject: Horizon3.ai - New Rapid Response Test Available for Oracle PeopleSoft PeopleTools Remote Code Execution Vulnerability
To: Your <admin@example.com>

HORIZON3.ai

The Horizon3 attack team has released a Rapid Response test for CVE-2026-35273, **Oracle PeopleSoft PeopleTools Remote Code Execution Vulnerability**. Active exploitation has been observed and confirmed. This Rapid Response is listed on the CISA Known Exploited Vulnerabilities catalog.

No exposure to this Rapid Response has been identified based on known assets. Run this Rapid Response test on additional targets as necessary.

Oracle PeopleSoft PeopleTools contains a critical vulnerability in the Updates Environment Management component.

View Details

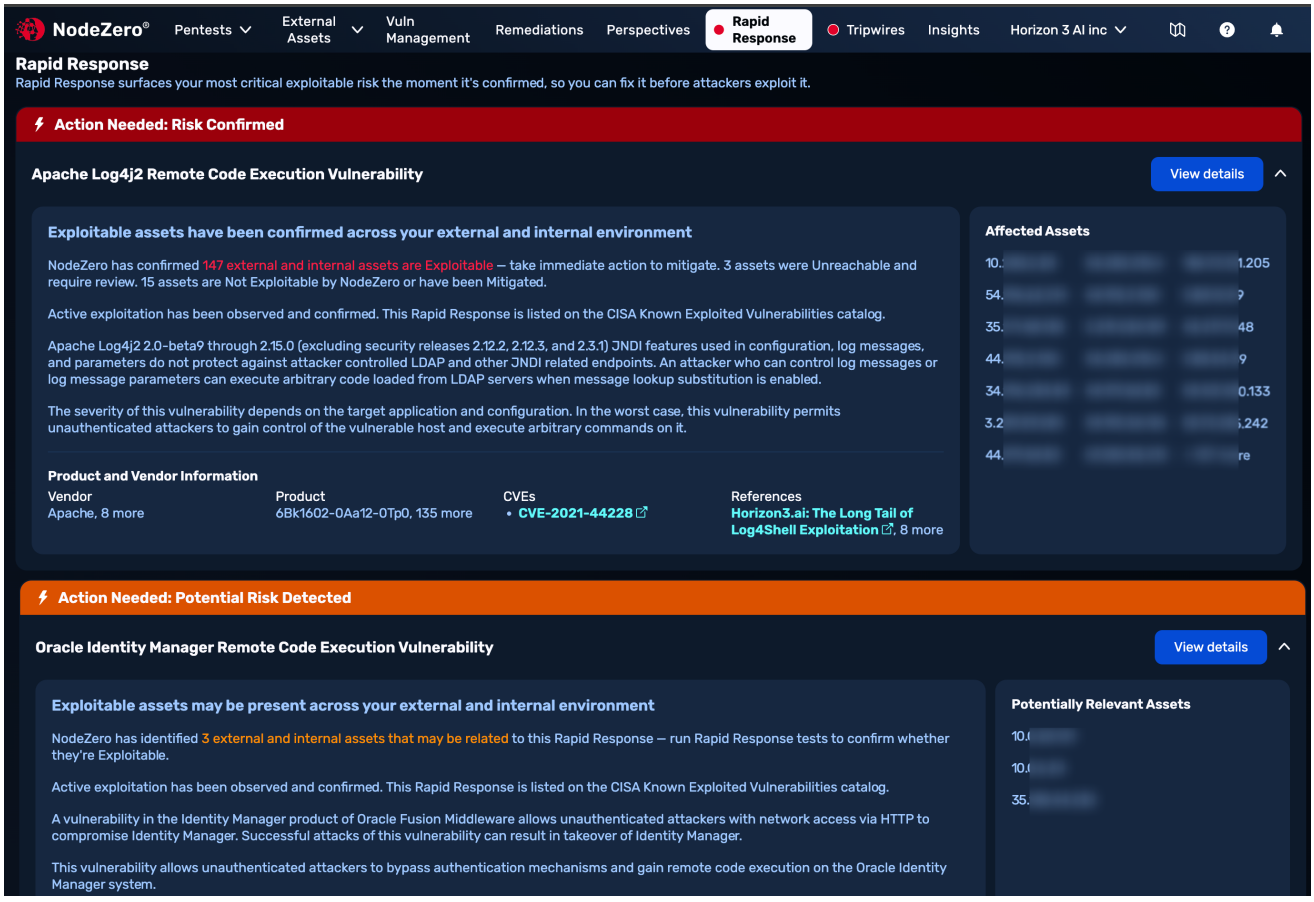
Horizon 3 AI Inc

6.3 Portal View

6.3.1 Rapid Response Portal View

At the top of the NodeZero Portal, select the **Rapid Response** tab to display all current Rapid Response advisories for your environment in a scrollable, paginated feed. As shown below, cards for individual advisories appear in prioritized order:

- **Action Needed: Risk Confirmed** red advisories (where present) appear at the top. These highlight vulnerabilities that (as assessed by Horizon3.ai) place your environment's specific assets at risk of exploitation, based on past NodeZero pentest results.
- **Action Needed: Potential Risk Detected** orange advisories (where present) follow all **Risk Confirmed** cards. These highlight vulnerabilities whose fingerprint matches some of your assets' attributes. They prompt you to run a targeted test (where available) to assess your actual risk.
- **No Exploitability Risk Detected By NodeZero** If you see only a green advisory, this is a welcome indication that there is no current known exposure to any existing Rapid Response vulnerabilities.



A green all-clear advisory appears on a shallow card:

Summary Contents

In this summary feed, each advisory generally displays the following information:

- A description of the vulnerability, of how attackers can exploit it, and of the potential impact of an exploit.
- A vendor and product affected (from broader lists that you can display on this advisory's Details page).
- CVE (Common Vulnerabilities and Exposures) or H3 (Horizon3.ai-discovered weakness) identifiers.
- Most-relevant external reference.
- A list of your affected assets, by IP address.

Rapid Response Details

Click any advisory's **View details** button to open a Rapid Response Details page with personalized context about how to eliminate your risk.

Additional Vulnerabilities

Below your prioritized feed is a **Rapid Response Vulnerabilities** table that complements it. This compact, paginated table lists Rapid Response advisories that do **not** (in Horizon3.ai's assessment) appear to affect your environment's assets. In other words, these advisories are not in a Risk Confirmed, Potential Risk Detected, or Regressed state.

This table provides a search field at its upper right, and you can sort the display by most columns. Along with Vendor, Product, and historical columns, you can click into the:

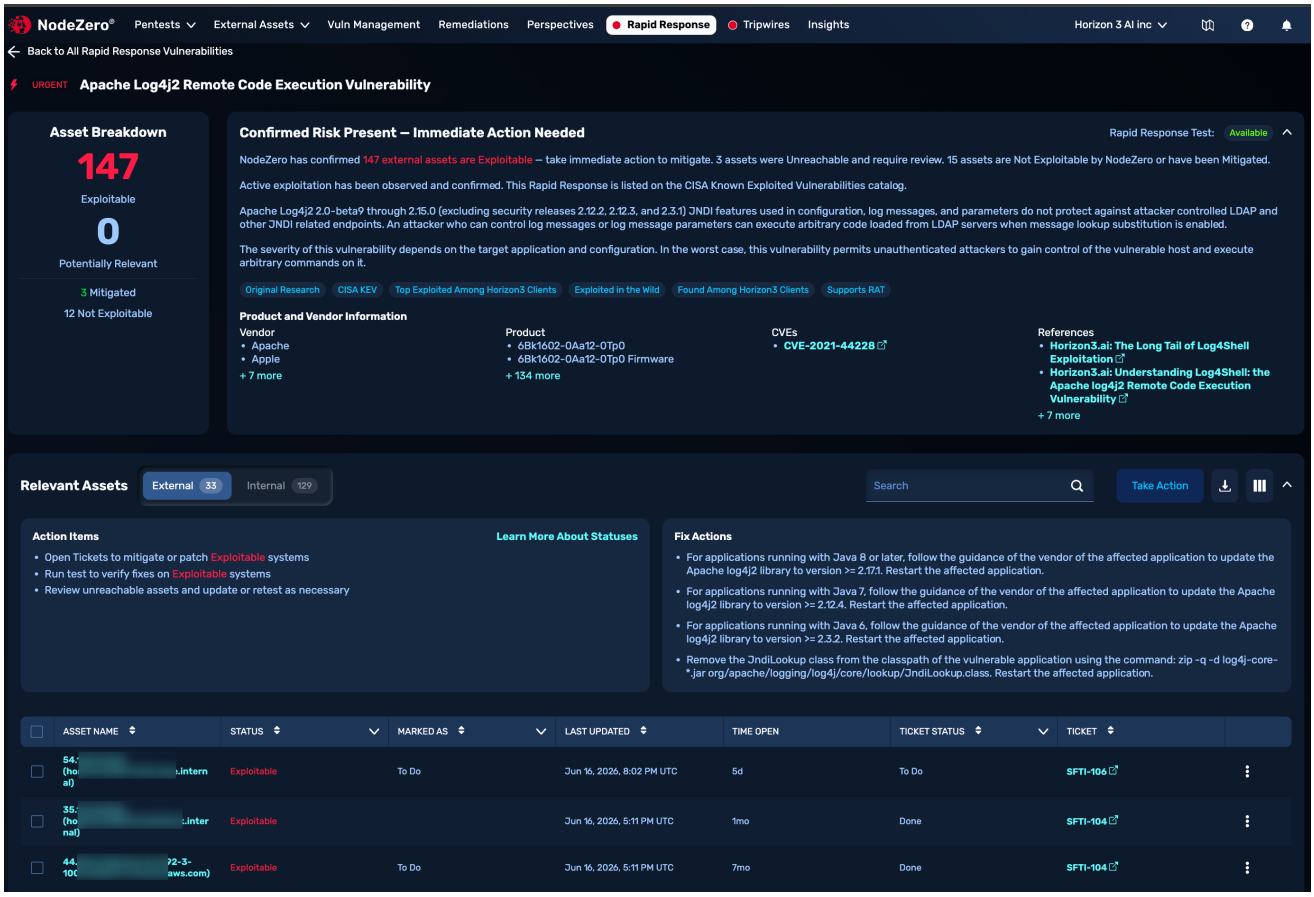
- **NAME** column to open the corresponding Rapid Response Details page.
- **WEAKNESS ID** column to open a CVE Catalog (nvd.nist.gov) page in a new tab. (H3 weakness IDs are also displayed in this column, but are not linked or clickable.)
- **TEST AVAILABILITY** column, where any **Available** link opens a test configuration modal from which you can launch the related Rapid Response test.

Rapid Response Vulnerabilities
Search
NAME
WEAKNESS ID
VENDOR
PRODUCT
ADVISORY PUBLISHED
TEST AVAILABILITY
TEST ADDED
Splunk Enterprise PostgreSQL Sidecar Service Arbitrary File Write Vulnerability
CVE-2026-20253
Splunk
Splunk Enterprise
Jun 10, 2026
Available
Jun 15, 2026
Oracle PeopleSoft PeopleTools Remote Code Execution Vulnerability
CVE-2026-35273
Oracle
Peoplesoft Enterprise Peopletools
Jun 7, 2026
Available
Jun 12, 2026
Ivanti Sentry Pre-Auth OS Command Injection Vulnerability
CVE-2026-10520
Ivanti
Mobileiron Sentry
Jun 8, 2026
Available
Jun 10, 2026
Palo Alto Networks PAN-OS GlobalProtect Authentication Bypass Vulnerability
CVE-2026-0257
Palo Alto Networks
Globalprotect, 2 more
May 13, 2026
Available
Jun 5, 2026
Gitea Container Registry Authentication Bypass Vulnerability
CVE-2026-27771
Gitea
Gitea
Available
May 28, 2026
Starlette Host-Header Authentication Bypass affecting LiteLLM, vLLM,
CVE-2026-48710, 1 more
Berrial, 2 more
Litellm, 2 more
May 21, 2026
Available
May 27, 2026
Showing 1 - 25 of 37, Page 1 of 2

6.4 Rapid Response Details

6.4.1 Rapid Response Details

From the NodeZero Portal's summary feed of Rapid Response advisories, click any advisory's **View details** button to view extensive metadata about the vulnerability or issue, along with personalized guidance and progress metrics on eliminating your risk. This Details page includes multiple panels, which we describe below.



Asset Breakdown

This panel lists the number of assets that Horizon3.ai has found to be **Exploitable**, **Potentially Relevant**, **Mitigated**, **Regressed**, or **Not Exploitable**.

Description

An expanded version of the description card in the summary feed, this panel provides links to access all affected vendors and products, all relevant CVE (Common Vulnerabilities and Exposures) or H3 weakness identifiers, and all relevant external references that Horizon3.ai has identified.

This panel might include any combination of status tags that summarize public and Horizon3.ai original research into the vulnerability.

The upper right indicates whether a Rapid Response Test is currently **Available** or **Unavailable**.

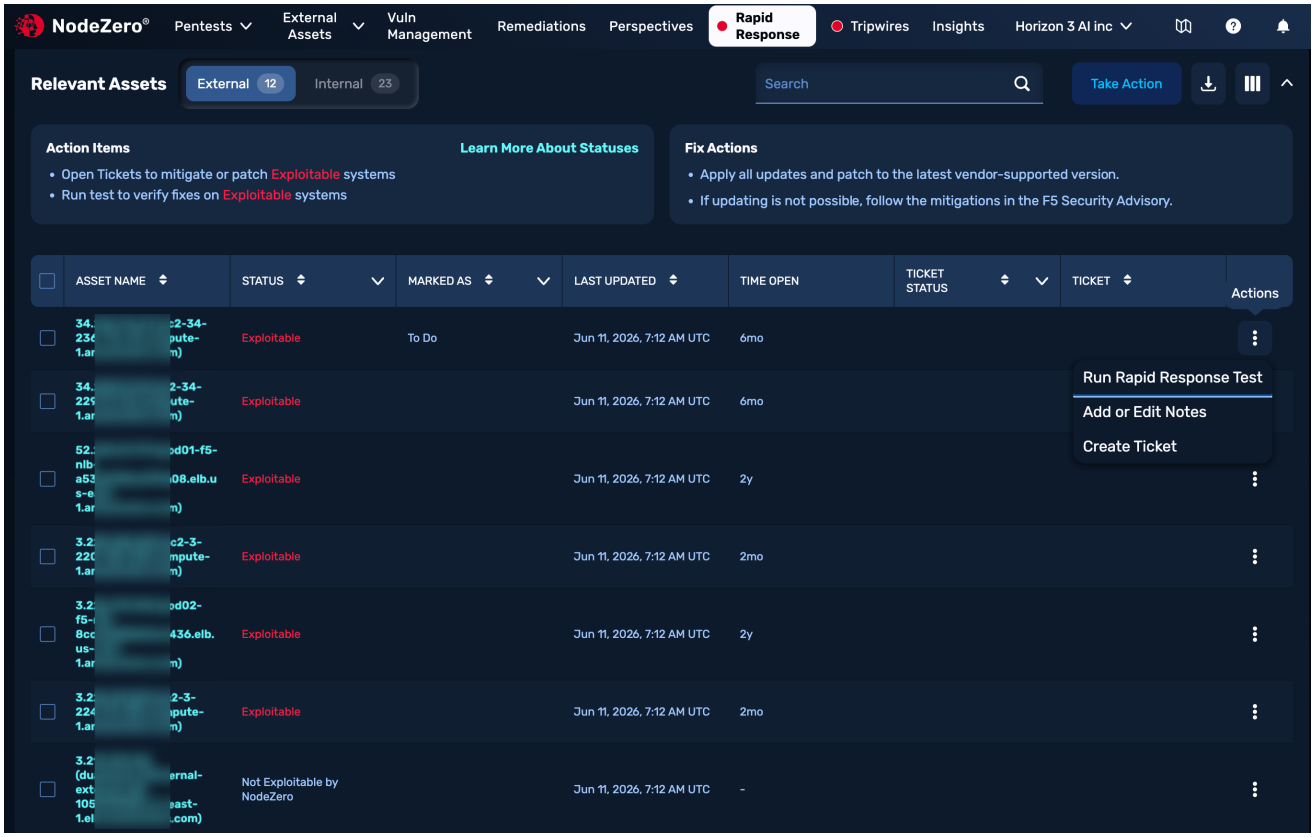
Relevant Assets

This paginated table provides a single interface to manage and audit all your assets & statuses. Along the top, you can filter the display using **External** versus **Internal** buttons, a search bar, and column toggles.

As with other NodeZero tables, most columns offer sorting and/or filtering controls. Each **ASSET NAME** entry links to an Asset Details page.

Each **TICKET** entry, where present, links to a ticket you've created in a ticketing system, like Jira, that you've integrated and synced via Vulnerability Management Hub. Adjacent columns show the status of the relevant asset and of any ticket.

You can use the check boxes at left to select multiple assets before taking a single action (such as creating one test or ticket for the group), using the Take Action control above the table.



ASSET STATUS

This column shows the current status of each potentially affected asset. This status will change as you mitigate risks, and as you or NodeZero conduct further testing.

Ideally, you want to see statuses trending toward green or gray **Not Vulnerable**, **Not Exploitable**, or **Mitigated**. (Click **Learn More About Statuses** above the table to see descriptions and recommended actions for each status.)

TAKE ACTION

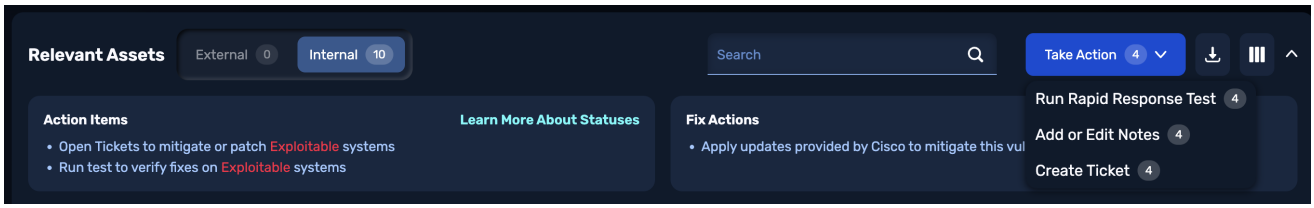
Above the Relevant Assets table, the **Action Items** and **Fix Actions** panels display personalized guidance about how to address your most exploitable assets.

The **Take Action** control at the upper right acts in one of two ways:

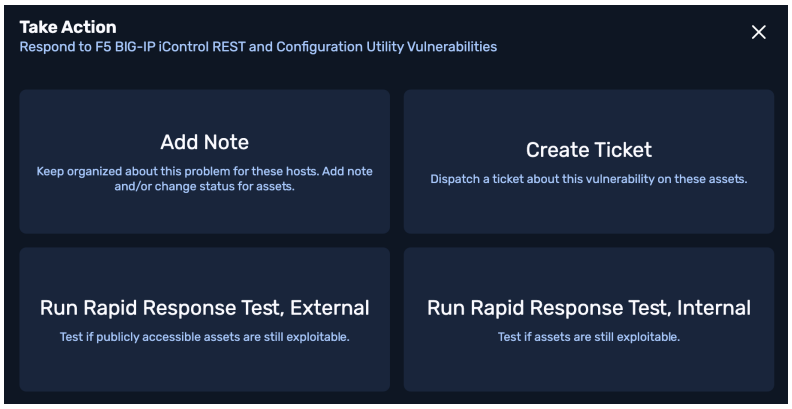
If you selected individual assets in the Relevant Assets table, you'll see a drop-down that provides options scoped to that group of assets: You can run a targeted internal or external Rapid Response test to prove exploitability or

successful mitigation, or add a status note within NodeZero, or create an actionable ticket within an integrated ticketing system (indicated in Relevant Assets above).

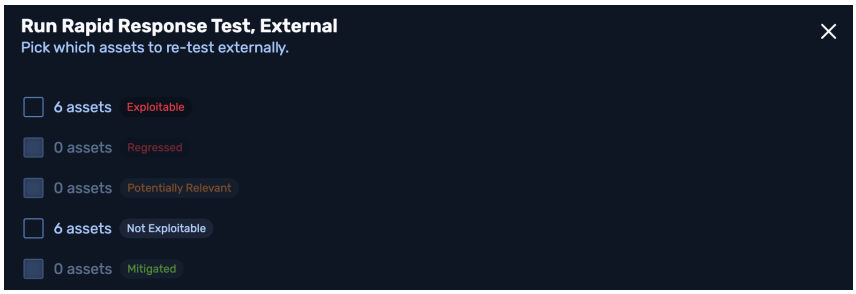
If you have not integrated a ticketing system, the **Create Ticket** button here downloads a CSV (comma-separated value) file, containing relevant details to relay to your team to fix the issue(s).



If you did **not** select individual assets in the Relevant Assets table, clicking this button opens the modal shown below.



In this case, you have the option to take action against **all** assets in a given status. You can select one or more available statuses using check boxes.



Mitigation Progress

This waterfall chart, shown below, traces the flow of relevant NodeZero pentests over time. Colors indicate Exploitable versus Mitigated status, and here again, you ideally want to see tests trend from red to green.

The trace begins when assets are found to be exploitable, and runs through their mitigated status. Any regressions extend the trace. Click a test link below the chart to see detailed test results.

Events and Actions Timeline

This paginated list, shown at the bottom of the preceding screenshot, shows significant events in assessing the vulnerability: tests that your organization ran against relevant assets; tests by which the Horizon3.ai Attack Team discovered exploitable/not exploitable assets; and disclosure to vendors and public databases.

Entries include a timestamp, and (where relevant) the user who ran the test, the number of assets evaluated, and a summary of the results.

Relevant Tests

In the **Relevant Rapid Response Tests** paginated table, shown below, you can click through to individual Rapid Response (targeted) Test results pages. This table includes both internal and external tests, tracked against relevant Rapid Response advisories. The table provides multiple search, filtering, and sorting options.



Status Tags

On the description card at the top of an Asset Details page, you might see any combination of status indicators below the description text.

If a vulnerability is actively being assessed by the Horizon3.ai Attack Team, you'll see an `In Progress` tag will follow the title. This status progresses as follows:

- During its assessment, Horizon3.ai determines whether the vulnerability is truly exploitable by analyzing publicly available proofs of concept, or by reverse-engineering the vulnerability to come up with a proof of concept.
- Based on its assessment, Horizon3.ai will alert any Rapid Response subscribers found to be running exploitable assets on the internet, and we will add the vulnerability to NodeZero for testing.
- If Horizon3.ai deems a vulnerability to be of low attacker value, we will not add the vulnerability to NodeZero.
- Once Horizon3.ai's assessment is complete, we will remove the `In Progress` status tag.

Additional tags might be present:

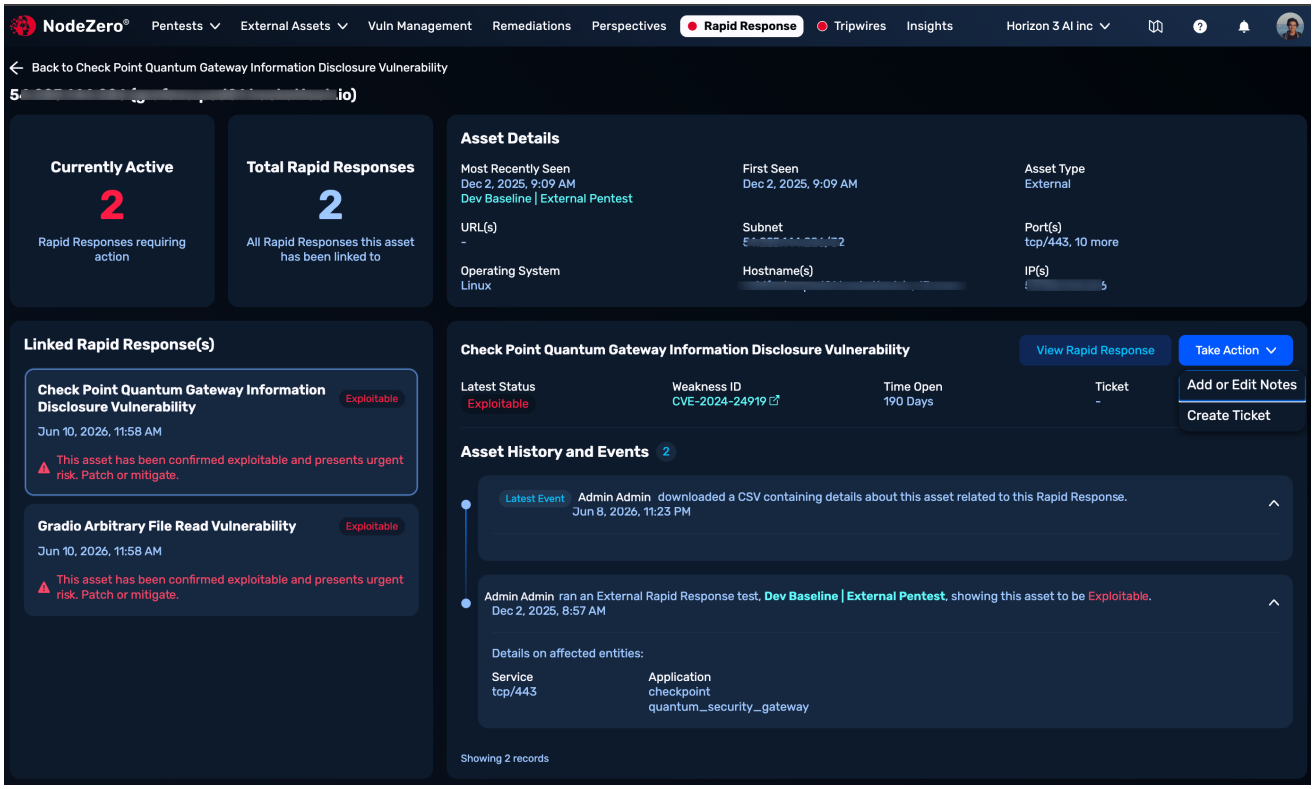
- `CISA KEV`: Displayed if the vulnerability is on the CISA Known Exploited Vulnerabilities list
- `Exploited in the Wild`: Displayed if there are reports that the vulnerability has been exploited in the wild, prior to potentially making it to the CISA KEV list. This tag will not be shown if the vulnerability has been added to CISA KEV list.
- `Found Among Horizon3.ai Clients`: Displayed if any Horizon3.ai clients were found to have a vulnerable application or device in their network.
- `Reversed by Horizon3`: Displayed when Horizon3 reverse engineered the vulnerability to create a proof of concept.
- `Discovered by Horizon3`: Displayed when the vulnerability was discovered by Horizon3.ai as a zero-day.
- `Original Research`: Displayed when Horizon3.ai performed and published original analysis related to the vulnerability, outside of discovering the vulnerability or reversing it.
- `Top Exploited`: Displayed when many Horizon3.ai clients have been found to have the vulnerable application or device.
- `Supports RAT`: Supports tests using the Horizon3.ai remote access tool, which enables deeper post-exploitation analysis of what an attacker could do after compromise, and supports dropping Tripwires.

6.5 Asset Details

6.5.1 Rapid Response Asset Details

From a Rapid Response Details page's Relevant Assets table, click any asset of interest to see metadata about this specific asset, as shown below.

The top-left panel displays a count of Rapid Response advisories linked to this asset. The top-right panel displays relevant asset metadata and identifiers, with a link to the most recent test run against this asset.



Linked Rapid Response(s)

This lower-left panel displays this asset's exposure status against associated Rapid Response advisories. Click a tile to give it focus – this will change the right-hand panels to display metadata about how each advisory affects this asset.

Rapid Response Summary

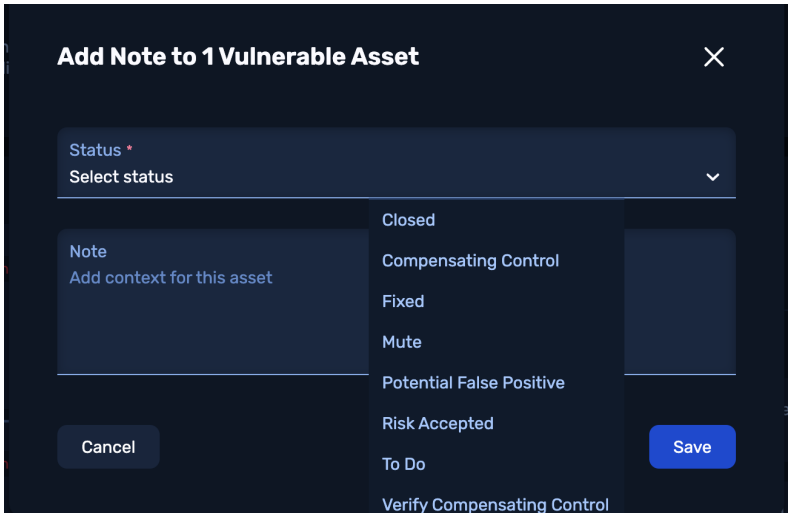
For the selected advisory, this right-hand panel shows this asset's latest vulnerability status, the weakness ID (linked CVE or Horizon3.ai-discovered H3), the time since the weakness was discovered, and any linked ticket in your ticketing system.

Click the **View Rapid Response** button to see the Rapid Response Details page for this advisory.

Take Action

This drop-down, shown above, provides two options:

- **Add or Edit Notes** Opens the modal shown below, where you can assign a NodeZero-native status, and enter or update free-text notes.
- **Create Ticket** writes an actionable ticket for you in a ticketing system, like Jira, that you've integrated and synced via Vulnerability Management Hub. If you have not integrated a ticketing system, this instead downloads a CSV (comma-separated value) file, containing relevant details to relay to your team to fix the issue(s).



Asset History and Events

At the lower right, this paginated timeline tracks the assessment progress of this asset. Entries might include:

- Your organization's completed Rapid Response or other NodeZero tests that confirmed this asset's Exploitable (or other) status.
- The Horizon3.ai Attack Team's completed tests that identified the asset's exploitability.
- Your organization's ticket creations, status annotations, and CSV downloads related to this asset.

Cards include a timestamp, the actor who performed the action, and (where applicable) a link to the test results or ticket and a summary of the service and application tested.

6.6 Targeted Tests

6.6.1 Running a Rapid Response Test

In many cases, Rapid Response details pages and tables will provide a link to run a Rapid Response test. This is a targeted test during which NodeZero will check your assets for that vulnerability, and will attempt to exploit it, as a benign assessment of its severity. NodeZero will not look for, nor report on, any other vulnerabilities during a Rapid Response test.

For example, to get started from a Relevant Assets table, you can select check boxes beside the assets you want to test, then select **Take Action > Run Rapid Response Test**.

Take Action drop-down opened from Rapid Response Details page, badged with "4" selected assets, and showing options labeled "Run Rapid Response Test," "Add or Edit Notes," and "Create Ticket"

This will expose the test configuration panels shown in the screenshot below, and described in the next few sections.

Rapid Response Selection

This first section on the Rapid Response Test page, shown below, confirms the Rapid Response advisory you've chosen to test. It displays a subset of the metadata from the corresponding Asset Details page, including current status tags. Click the teal link to navigate to the full page.

Test Name

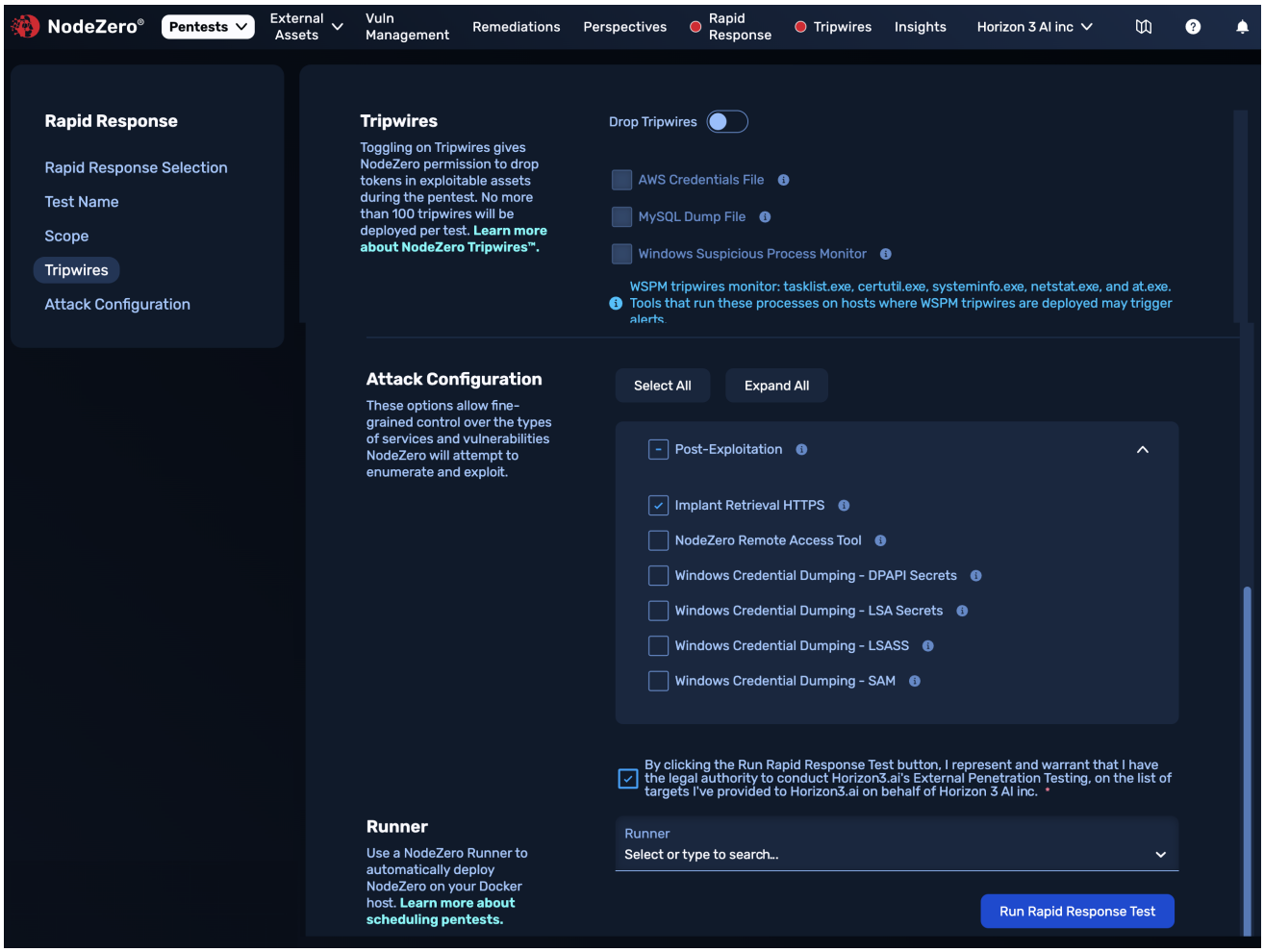
This section (shown above) confirms whether you'll run an Internal or External pentest. Options include override the default test name and (for External tests) enabling **Get IP before starting pentest** if you need to allowlist IP addresses.

Scope

This section will vary for External (shown above) versus Internal tests. Here, you have the option to expand or modify the test's scope, compared to the group of assets you selected before opening this config modal. You can add a single IP or a block of IPs.

Additional Controls

The screenshot below is a composite of remaining controls available for both Internal and External tests, described in the next few sections.



TRIPWIRES

Toggling on **Drop Tripwires** here exposes check boxes to enable individual types of Tripwires.

ATTACK CONFIGURATION

To navigate the granular options in this section, see details for Internal or External pentests.

ACKNOWLEDGMENT

On External tests, you'll need to select the check box acknowledging that you have authority to test against the enumerated targets.

RUNNER

On Internal tests, you'll have the option to select a Runner. Runners must be preconfigured before they will be available in this drop-down.

Run the Test!

The final step is to click the **Run Rapid Response Test** button. If you're running an Internal test without a Runner, you'll need to copy and paste the curl one-liner onto your Docker host. External tests are launched automatically from Horizon3.ai's cloud and require no further action.

You've run a Rapid Response Test

NodeZero sends an email once the test completes.

Test Results

Completed tests are tabulated in each Rapid Response Details page's Relevant Tests section.

There, click a link in the **Name** column and you'll see the same kind of results page, and reports download options, as for other pentests.

Alternative Tests

For some advisories and assets, even if a Rapid Response Test is not available, you'll still be able to test for the vulnerability within a NodeZero Internal or External pentest.

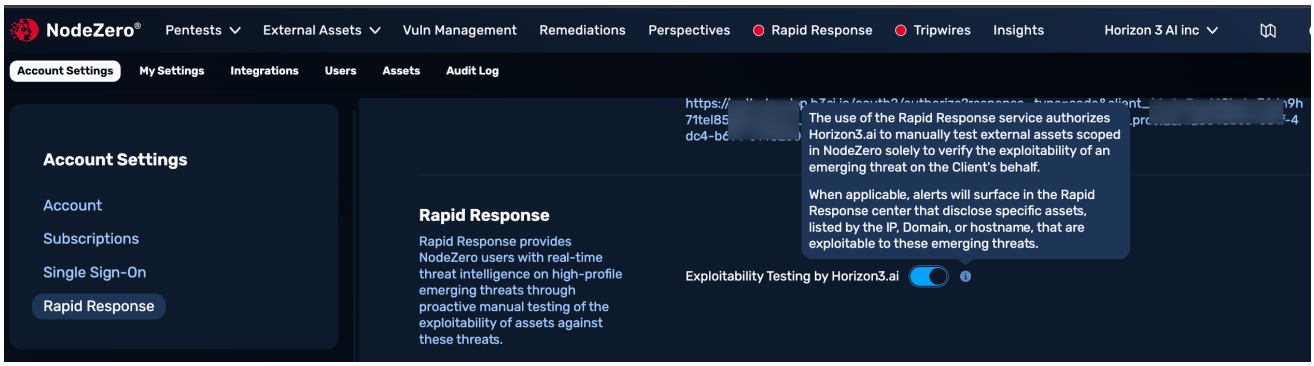
6.7 Managing Rapid Response

6.7.1 Managing Rapid Response

Rapid Response permissions are governed by NodeZero overall access role. Org Admins have the permissions described below, and automatically receive Rapid Response email Notifications. Users receive these emails if enabled by an Admin.

Accessing Rapid Response Settings

Org Admins can control Rapid Response proactive testing for their organization. From your profile at the NodeZero Portal's upper-right corner, select **Settings**. From the resulting **Account Settings** page, scroll down to the **Rapid Response** section. Here is a composite screenshot lining up this section with its left navigation link:



Testing Opt In/Out

Org admins can opt their org in or out of proactive exploitability testing, as shown above. Upon opt-out, you'll see a confirmation banner, with a button to **Opt in** again.

This is a trade-off: By opting out, you gain stricter control over whether Horizon3.ai manually touches your internet-facing assets. However, you lose key Rapid Response benefits – Horizon3.ai testing of external assets as soon as we have a working exploit, along with our ability to immediately provide you with personalized notifications as soon as we confirm your exposure.

Operationally, this forces your team to do more of the first-triage work yourselves, rather than having Horizon3.ai eliminate that step.

User Notifications

At **Settings > Users > Edit User > Rapid Response**, Org Admins can grant individual Users full permissions to receive email notifications. Toggle this off to opt a User out of notifications.

Frequently Asked Questions

Why am I not seeing Rapid Response in the NodeZero Portal?

Rapid Response is available on certain account plans, or as an add-on. Contact your Horizon3.ai representative to add the feature to your account.

Why is it important to frequently? run External Asset Discovery

Refreshing External Asset Discovery often is important in order to nurture Rapid Response with an accurate assets database to monitor for exposure. Within the NodeZero Portal's Rapid Response section, you might see reminders to re-run out-of-date asset discovery.

7. CLI

7.1 NodeZero CLI

Horizon3.ai provides a Command Line Interface (CLI) tool called `h3-cli`, which offers a convenient way to interact with the Horizon3.ai Application Programming Interface (API). The API gives programmatic access to a selection of features that are also available through the Horizon3.ai Portal.

The CLI allows users to manage and automate key aspects of pentesting without relying on the Portal. You can configure and launch assessments, schedule recurring tests, and retrieve results, all from your terminal or as part of a script.

This makes the CLI an excellent choice for security teams that want to integrate pentesting into existing workflows, CI/CD pipelines, or other automated processes.

7.1.1 When to use

- You want to schedule and run pentests without using the Portal
- You need to integrate pentesting into CI/CD pipelines or workflows
- You need a way to securely manage credentials for injection
- You are managing multiple environments and want to automate setup
- You need advanced scheduling options

7.1.2 Usage

Ready to use `h3-cli`? Choose from the options below:

Getting Started

Step-by-step instructions to set up and begin using the h3-cli tool.

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

7.1.3 Customization

Want more control? Check out the NodeZero MCP Server, or create your own tools and workflows using the Horizon3.ai GraphQL API.

7.2 Getting Started

7.2.1 Getting Started with CLI

`h3-cli` is a convenient CLI (command-line interface) for accessing the Horizon3.ai API. The Horizon3.ai API provides programmatic access to a subset of functionality available through the Horizon3.ai Portal.

At a high level, the API allows you to:

- Monitor the status of a pentest while it is running
- Retrieve a pentest report after it's complete
- Manage multiple environments and want to automate setup
- Configure advanced scheduling options

The API can be used for a variety of use cases such as scheduling periodic assessments of your environment or kicking off a pentest as part of a continuous integration build pipeline.

Next step

1 **Set Up the CLI**

Additional info about how to deploy, schedule pentests, monitor, and parse results.

2 **Upgrade the CLI**

Additional info about how to deploy, schedule pentests, monitor, and parse results.

3 **Using the CLI**

Additional info about how to deploy, schedule pentests, monitor, and parse results.

7.2.2 Installation and Initial Setup

The steps below will get you up and running quickly with h3-cli. These instructions were tested on Linux machines, and generally should work on any POSIX-compliant system with bash support.

It is assumed you already have an account with Horizon3.ai. If not, sign up at https://portal.horizon3ai.com/.

Create an API key

An API key is required to access the H3 API. You can create one in the Portal under the User -> Account Settings menu.

When creating an API key you must assign it a role that controls its permissions. The available roles are:

- **User:** Basic read/write permissions. The API key can run pentests and read results.
- **Read-only:** The API key can read pentest results, but cannot run pentests.
- **NodeZero Runner:** A specialized, heavily restricted role designed specifically for NodeZero Runners.

We recommend the **User** role if you're testing out h3-cli and want to experiment with all its features. After that, you may want to use more restrictive permissions, based on your use case. For example, if you only want to use h3-cli to set up a NodeZero Runner, we recommend using the **NodeZero Runner** role.

You can easily manage multiple API keys within the same h3-cli install. Learn more here.

Keep your API key secure, as anyone with your API key can access your H3 account. Think of an API key as a username + password rolled into one. Anyone with the API key can access your account from anywhere. h3-cli will store your API key under the `$HOME/.h3` directory. This directory is created during installation and configured with permissions such that only you can read or write to it.

Install h3-cli

Download and run the installation script:

```
curl https://downloads.horizon3ai.com/utilities/cli/easy_install.sh | bash -s <your-api-key-here>
```

This will install the CLI in a new directory, `h3-cli`. The new directory is created within the directory where the above command is run.

The install script will install dependencies (`jq`) and create your `default` h3-cli profile under the `$HOME/.h3` directory. Your API key is stored in your h3-cli profile. The directory and profile permissions are restricted so that no other users (besides yourself) can read or write to it.

The install script will ask you to edit your shell profile (for example `$HOME/.bash_profile`, `$HOME/.bash_login`, `$HOME/.profile`, depending on your operating system) to set the following environment variables:

- `H3_CLI_HOME`: this environment variable is set to the full path of the h3-cli directory. It is used by h3-cli to locate itself and its supporting files.
- `PATH`: this environment variable specifies the directories to be searched to find a shell command. You can add `$H3_CLI_HOME/bin` to your `PATH` to find the `h3` command.

After updating your shell profile, you will have to re-login or restart your shell session to pick up the profile changes. You can then verify that you can invoke `h3` by running it from the command prompt:

```
h3
```

If everything's installed correctly, you should see the h3-cli help text.

Next step

2 Upgrade the CLI

Additional info about how to deploy, schedule pentests, monitor, and parse results.

3 Using the CLI

Additional info about how to deploy, schedule pentests, monitor, and parse results.

7.2.3 Upgrading the CLI

We release new features, bug fixes, and other updates for the h3-cli on monthly boundaries. Upgrade your installation using one of the methods below.

CLI Command | **Run Install Script** | **Download Files** | **OVA Host Utility**

From the CLI, enter: `h3 upgrade`

Support for this command started with the June, 2023 release.

If you get an `ERROR: unrecognized command: "upgrade"`, then you are on an earlier version of h3-cli that does not support the upgrade command. Use one of the adjacent options to upgrade h3-cli.

Run this command from h3-cli's parent directory (i.e., the directory that contains the `h3-cli/` directory):

```
curl -L https://downloads.horizon3ai.com/utilities/cli/easy_install.sh | bash
```

Previously downloaded the repo as a zip file? Then re-download the zip file and unzip it to the same location (in other words, replace your existing h3-cli installation with the new zip).

Using the OVA host virtual machine? If you update the host using the Host Utility script, this will also update the h3-cli version and any Runners on the host.

Checking the current version

As of June, 2023, you can view your current version of h3-cli via:

```
h3 version
```

Updating the source location

Prior to July, 2025, h3-cli pulled CLI (and Runner) updates from `github.com`. As of July, 2025, **new** CLI deployments and new Runner installations now pull from `downloads.horizon3ai.com`.

If you have older installs and want to switch their source (for future updates) to `downloads.horizon3ai.com`, log into your NodeZero host and run `h3 upgrade ng`. This will pull future updates from the new location. It will also remove the requirement for network access to `github.com`.

Next step

3 Using the CLI

Additional info about how to deploy, schedule pentests, monitor, and parse results.

7.2.4 Using the CLI

This page lists CLI commands for common use cases and tasks.

Verify connectivity with the API

Run the following command to verify connectivity with the API.

```
h3 hello-world
```

You should see the response:

```
{
  "data": {
    "hello": "world!"
  }
}
```

If you are getting an error response, please contact H3 via the chat icon in the Horizon3.ai Portal.

Query the list of pentests in your account

The command below will return the list of pentests in your account, most recent first.

```
h3 pentests
```

To filter for pentests that match a given search term, pass the search term as a parameter:

```
h3 pentests sample
```

Query a specific pentest from your account

To query the most recent pentest in your account:

```
h3 pentest
```

To query any pentest in your account, pass the `op_id` of the pentest as a parameter:

```
h3 pentest your-op-id-here
```

Several h3-cli commands will use the most recent pentest as the default, unless an `op_id` is passed as a parameter.

The terms "op" and "pentest" are often used interchangeably.

Run a pentest

Running a pentest requires specifying an op template. An op template specifies a full pentest configuration, which includes scope, attack parameters, and other (optional) configuration.

Horizon3.ai provides new users with a default op template named `Default 1 - Recommended`. This template is always up-to-date with our latest attack parameters and recommended configuration. The default template does not define a scope, in which case NodeZero will use Intelligent Scope - NodeZero's host subnet will provide the initial scope, and it will expand organically during the pentest as more hosts and subnets are discovered. For more information on Intelligent Scope and other deployment options, visit our product documentation.

For experienced users, custom op template(s) may be created via the Horizon3.ai Portal. To create a custom op template, walk through the Run a Pentest modal until you see the option to customize the pentest configuration. The op template can be created without actually running the pentest.

To provision a pentest using the default op template and Intelligent Scope:

```
h3 run-pentest
```

The JSON response contains the details for the newly created pentest. You can verify the pentest is provisioning by checking your Horizon3.ai Portal, or by running `h3 pentest`.

There are several ways to specify additional parameters when creating pentests. For more information see additional examples here.

WAIT! YOU'RE NOT DONE!

For internal pentests (which are the default), additional steps are required before the pentest will begin running. See the next section about downloading and running NodeZero in order to complete the initiation of your pentest.

If you are running an external pentest, NodeZero is launched for you automatically in the H3 cloud as part of `h3 run-pentest`, in which case there are no additional steps on your end to initiate the pentest.

Run NodeZero

The following step applies to internal pentests only; for external pentests, NodeZero is launched for you automatically in the H3 cloud.

After creating an internal pentest, you then have to run our NodeZero container on a NodeZero Host inside your network. This is done by running the NodeZero Launch Script on your NodeZero Host.

To run the NodeZero Launch Script for your most recently created pentest:

```
h3 run-nodezero
```

YOUR PENTEST HAS BEEN LAUNCHED! Assuming all commands ran without error, then you have successfully created and launched your pentest. You should see output from the NodeZero Launch Script being logged to the console. The script will first verify your system is compatible with NodeZero before downloading and running it. When the pentest is complete, NodeZero will automatically shut itself down.

NodeZero runs as a container. You can view it using `docker ps`, or `sudo podman ps` if you are using Podman. The container name will be of the form `n0-xxxx`.

Download pentest reports

After your pentest has finished, use the following command to download a ZIP file containing all PDF and CSV reports for the most recently created pentest:

```
h3 pentest-reports
```

The above command will download the ZIP file to `pentest-reports-{op_id}.zip` in the current directory.

Download individual reports

You can identify and download individual reports using the Horizon3.ai GraphQL API.

Authentication & h3-cli profiles

Authentication happens seamlessly and automatically when you invoke the `h3` command. There's nothing explicit you need to do to authenticate. This section documents the underlying mechanics.

h3-cli reads your `H3_API_KEY` from your h3-cli profile (under `$HOME/.h3`) to authenticate to the Horizon3.ai API and establish a (temporary) session. The session token (a JWT) is cached under `$HOME/.h3`. The session token expires after 1 hour, at which point h3-cli will automatically re-authenticate and re-establish a session.

You can explicitly authenticate using the following command:

```
h3 auth
```

The above command will output the session token (and also cache it under `$HOME/.h3`). If you already have an established (non-expired) session token, `h3 auth` will continue to use that session token rather than re-authenticate.

If you want to force h3-cli to re-authenticate, use the `force` option:

```
h3 auth force
```

H3-CLI PROFILES

You can manage multiple h3-cli authentication profiles under the same `$HOME/.h3` directory. Each h3-cli profile has its own API key.

When you first install h3-cli it will automatically create an initial profile named `default` with the API key you provided to `install.sh`.

If you wish to create another profile with a different API key, use the following command:

```
h3 save-profile my-profile {api-key}
```

This will create a profile named `my-profile` under `$HOME/.h3` for the given `{api_key}`. To activate the profile in your current shell session, use the following command (note the leading dot `.`):

```
. h3 profile my-profile
```

You can verify the currently active profile using `h3 profile`, and view details about its API key using `h3 whoami`:

```
h3 profile
h3 whoami
```

You can save multiple API keys under different h3-cli profiles and switch between them as needed using the command above. For example, to switch back to the `default` profile:

```
. h3 profile default
```

To view the list of h3-cli profiles under your `$HOME/.h3` directory:

```
h3 profiles
```

You can delete a profile from your `$HOME/.h3` directory using:

```
h3 delete-profile {name}
```

This will remove the profile named `{name}` and its API key from your `$HOME/.h3` directory on the local machine. Note that it will NOT revoke the API key; it only deletes it from the local machine. You can revoke the API key from the Portal.

Running pentests with h3-cli

This section contains additional examples for running pentests using h3-cli.

The simplest way to provision a pentest is to use the default op template and Intelligent Scope:

```
h3 run-pentest
```

To provision a pentest AND launch NodeZero on the local machine (for internal pentests only):

```
h3 run-pentest-and-nodezero
```

Note that this only applies to internal pentests. For external pentests, NodeZero is launched for you automatically in the H3 cloud as part of `h3 run-pentest`.

If you happen to run `h3 run-pentest-and-nodezero` for an external pentest, it will simply skip the part where it downloads and runs NodeZero, since that is handled automatically in the H3 cloud.

To run a pentest using a custom op template, specify it as a parameter to schedule_op_template.graphql:

```
h3 run-pentest '{"op_template_name":"your-op-template-here"}'
```

To run a pentest using the default op template but assign it a name of your choosing, use the optional `op_name` parameter:

```
h3 run-pentest '{"op_name":"your-op-name-here"}'
```

To run a pentest using the default op template but specify its name and scope, use the optional `schedule_op_form` parameter:

```
h3 run-pentest '{"schedule_op_form":{"op_name":"your-op-name-here", "op_param_max_scope": "192.168.0.0/24"}}'
```

Note that `h3 run-pentest` and `h3 run-pentest-and-nodezero` accept all the same optional parameters.

To run a pentest and assign it to a NodeZero Runner named `my-nodezero-runner`:

```
h3 run-pentest '{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'
```

RUNNING EXTERNAL PENTESTS

If you have an op template configured for your external pentest:

```
h3 run-pentest '{"op_template_name":"your-op-template-here"}'
```

If you do **not** have an op template, you can run an external pentest by first looking up your Scope's (Asset Group's) uuid via `h3 asset-groups`:

```
h3 asset-groups
```

Then use the command below to run an external pentest against that Scope. Substitute your Scope's uuid for `{your-asset-group-uuid}`:

```
h3 run-pentest '{"schedule_op_form": {"op_type": "ExternalAttack", "asset_group_uuid": "{your-asset-group-uuid}"}}'
```

Powered by GraphQL

The Horizon3.ai API is powered by GraphQL. In addition to this CLI document, relevant documentation includes:

- GraphQL.org

ADVANCED: WRITING YOUR OWN GRAPHQL QUERIES

h3-cli provides a simple mechanism to run your own GraphQL queries. First you define the GraphQL query in a file (typically with a `.graphql` extension, although that is not required). Then pass the file to `h3 gql`:

```
h3 gql {your-query-file}
```

For example, define the following in a file named `my_session.graphql`:

```
query {
    session_user_account {
        email
        name
        company_name
    }
}
```

Then run:

```
h3 gql ./my_session.graphql
```

You should see the raw JSON response from the GraphQL server. You can pretty-print the JSON response using jq:

```
h3 gql ./my_session.graphql | jq .
```

Important! You must specify the path to the graphql file (full or relative, e.g. `./my_session.graphql` instead of just `my_session.graphql`), otherwise you risk colliding with graphql files that h3-cli uses internally.

ADVANCED: PARAMETERIZED GRAPHQL QUERIES

GraphQL queries can also define parameters, which are passed to `h3 gql` as a JSON object.

For example, define the following in a file named `my_pentest.graphql`:

```
query q($op_id: String!) {
    pentest(op_id:$op_id) {
        op_id
        name
        state
    }
}
```

In this example, `$op_id` is a parameter that must be provided in order to run the query. The parameter is passed to the query within a JSON object:

```
h3 gql ./my_pentest.graphql '{"op_id":"your-op-id-here"}' | jq .
```

Substitute `your-op-id-here` with an actual `op_id`.

Use cases

- **Automated NodeZero deployment using a NodeZero Runner.** Learn how to use a NodeZero Runner to deploy NodeZero on your NodeZero Host automatically, without having to manually copy+paste the NodeZero Launch Script.
- **Automated scheduling.** Learn how to run pentests automatically on a regular schedule, for example once a week or once a month.
- **Auto-injected credentials.** Learn how to automatically inject credentials into a regularly scheduled pentest using a NodeZero Runner.
- **Monitoring pentests.** Learn how to monitor pentests using h3-cli.
- **Paginating results.** Learn how to paginate through large result sets using h3-cli.
- **JSON Parsing using `jq`.** Learn how to leverage the power of `jq` to parse JSON responses from h3-cli. `jq` can parse specific fields, print the structure of a response, and even transform a JSON response to CSV.

Next steps

Getting Started

Additional info about how to deploy, schedule pentests, monitor, and parse results.

CLI Guides

Additional info about how to deploy, schedule pentests, monitor, and parse results.

7.3 CLI Guides

7.3.1 CLI Guides

Explore focused, hands-on guides for using the Horizon3 `h3-cli` tool. These examples walk through real-world use cases such as scheduling pentests, injecting credentials, monitoring test activity, and handling result data.

ESSENTIALS

Schedule Pentest with CLI

Configure automated pentests using the CLI

Automate NodeZero Deployment

Auto-deploy NodeZero to a configured host using a NodeZero Runner

ADDITIONAL CLI CAPABILITIES

Go beyond the basics with advanced CLI functionality.

Auto-Inject Credentials

Inject credentials automatically into recurring tests

Monitor Pentests

Track the progress and status of live tests

Paginate Results

Handle large result sets efficiently with pagination

Parse JSON Results

Parse and transform pentest output

7.3.2 Schedule Pentest with CLI

A common use case for `h3-cli` is running pentests automatically on a recurring basis, for example once a week or once a month, without any required user intervention - no need to log into the Portal, no need to copy+paste NodeZero launch scripts.

The instructions below walk through how to use `h3-cli` to configure a pentest to run automatically on a regular basis.

You can also create pentest schedules directly in the Portal. See Portal feature Automate Scheduling of Pentests.

1. Enable automated NodeZero deployment

If you plan to run internal pentests (which are the default), you will need to enable h3-cli to automatically deploy NodeZero on a NodeZero Host inside your network.

This is done using a NodeZero Runner. The NodeZero Runner is a background process running on your NodeZero Host that automatically launches NodeZero whenever a new pentest is assigned to it.

To set up a NodeZero Runner, follow the instructions here.

2. Create a scheduled action to run a pentest on a recurring schedule

The command below will create a recurring schedule called `my-schedule` that will automatically run a pentest every Monday at 5pm UTC.

```
h3 create-scheduled-action \
    my-schedule \
    '0 17 * * 1' \
    run-pentest '{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'
```

The command uses a CRON expression, `0 17 * * 1`, to specify the recurring schedule for the pentest.

Breaking it down:

- `my-schedule` is the name of the schedule. A schedule may contain multiple actions. For example, you can configure timing windows for pentesting around by business hours, by scheduling a pentest to launch on Monday at 5pm, pause every day at 8am, resume at 5pm, and finally terminate on Friday if it is still running.
- `0 17 * * 1` is the CRON expression. CRON expressions specify the `{minute} {hour} {day-of-month} {month} {day-of-week}` to run a given action. Visit the [link]((https://crontab.guru/){:target="_blank"} for more information about CRON expressions.
- **NOTE:** Only hourly resolution is supported. The `{minute}` component of the CRON expression is always forced to be `0` on the backend.
- CRON expressions are in **UTC** time. So the example CRON expression above is set to **5pm UTC**.
- `run-pentest` is the action. Supported actions are:
- `run-pentest`: launches a new pentest (if one is not currently active for this schedule)
- `pause-pentest`: pauses the active pentest associated with the schedule
- `resume-pentest`: resumes the active pentest associated with the schedule
- `cancel-pentest`: cancel the active pentest associated with the schedule
- `'{"schedule_op_form":{"op_name":"Pentest created via h3-cli and launched via runner", "runner_name":"my-nodezero-runner"}}'`: additional parameters for the `run-pentest` action. These parameters are the same as those you would use if you executed `h3 run-pentest` directly from the command line.

A named schedule can have **only one active pentest at a time**. This prevents a schedule from kicking off a new pentest when its previous pentest has not yet completed.

403 ERROR

- If you receive the error `[403] You are not authorized`, you may be trying to use your **NodeZero Runner** API key to create the pentest schedule. For security reasons, NodeZero Runners have restricted permissions. All they can do (more or less) is run NodeZero for an already created pentest. They cannot create new pentests, view pentest results, or create pentest schedules.

  To create a pentest schedule, you'll need to create a separate API key with **User** permissions. You can then create a separate h3-cli profile for the new API key, and easily switch between multiple h3-cli profiles as needed.

- Learn more about managing h3-cli profiles here.
- For further assistance, contact H3 support via the chat icon in the Portal.

3. Verify your schedule is registered with H3

Use the following command to view your pentest schedules:

```
h3 schedules
```

If all is well, you should see an entry for your schedule `my-schedule`.

4. Test your scheduled action by triggering it now

To ensure everything is wired up as expected, you can trigger your scheduled action immediately with the following command:

```
h3 trigger-scheduled-action my-schedule run-pentest
```

This will trigger the `run-pentest` action for the `my-schedule` schedule, which will cause a pentest to be created. The NodeZero Runner on your NodeZero Host will see the new pentest and automatically launch NodeZero.

You can monitor the NodeZero Runner process by tailing the log:

```
tail -f /tmp/my-nodezero-runner.log
```

In a minute or so you should see the Runner kick off the NodeZero Launch Script for the newly created pentest. The NodeZero Launch Script will download and launch the NodeZero container on the local machine, just as if you had copy+pasted the `curl` command from Run Pentest wizard in the Portal.

You can view the newly created pentest via:

```
h3 pentest
```

Side note: If for whatever reason you need to kill the NodeZero Launch Script before it downloads and launches NodeZero, you can use `pkill`:

```
pkill -f h3-run-nodezero
```

Once the NodeZero container is running, you can manage its lifecycle via the Docker or Podman API.

5. Troubleshooting

If you don't see NodeZero get launched in the NodeZero Runner log, use `h3 schedules` to see if any errors occurred when the action was triggered:

```
h3 schedules
```

The command output will resemble the readout below. Look at the `last_triggered_*` fields to help diagnose any problems:

```
{
  "name": "my-schedule",
  "state": "ENABLED",
  "created_at": "2023-02-06T06:52:04.660895",
  "last_updated_at": "2023-02-10T23:16:36.722392",
  "actions": [
    {
      "action": "run-pentest",
      "params": {
        "schedule_op_form": {
            "op_name": "Pentest created via h3-cli and launched via runner",
            "runner_name": "my-nodezero-runner"
        }
      },
      "cron_expression": "0 17 * * 1",
      "cron_description": "At 05:00 PM, only on Monday",
      "last_triggered_at": "2023-02-10T22:08:05.010069",
      "last_triggered_time_ago": "an hour ago",
      "last_triggered_error": null
    }
  ]
}
```


You will also receive an email notification every time a scheduled action is triggered. If the action fails, the error will be included in the email.

For further assistance, contact H3 support via the chat icon in the Portal.

6. Create a second scheduled action to cancel the pentest (optional)

Let's add a second action to our schedule for canceling the pentest. We'll schedule it to run 1hr after the pentest is launched.

```
h3 create-scheduled-action my-schedule '0 18 * * 1' cancel-pentest
```

Once again we can test the action by triggering it immediately:

```
h3 trigger-scheduled-action my-schedule cancel-pentest
```

After a moment you should see your pentest get canceled and move into the post-processing state.

7. Enabling and disabling a schedule

You can view all of your schedules via:

```
h3 schedules
```

You can disable a schedule and all its actions via `disable-schedule`:

```
h3 disable-schedule my-schedule
```

And you can re-enable a schedule via `enable-schedule`:

```
h3 enable-schedule my-schedule
```

8. Configuring pentesting windows

You can use scheduled actions to pause and resume pentests around pentesting windows, e.g. around business hours. The commands below show how to create a schedule that will:

- launch pentests on Mondays at 5pm UTC
- pause the running pentest every weekday at 8am UTC
- resume the paused pentest every weekday at 5pm UTC
- cancel the pentest if it's still running on Friday at 8am UTC

```
h3 create-scheduled-action my-schedule '0 17 * * 1' run-pentest '{"schedule_op_form":{"op_name":"Auto-scheduled weekly pentest",
"runner_name":"my-nodezero-runner"}}'
h3 create-scheduled-action my-schedule '0 8 * * 2-4' pause-pentest
```

```
h3 create-scheduled-action my-schedule '0 17 * * 2-4' resume-pentest
h3 create-scheduled-action my-schedule '0 8 * * 5' cancel-pentest
```

9. Updating and deleting scheduled actions

You can update a scheduled action by simply running the `create-scheduled-action` again with the new settings. For example, if you wish to change the schedule above such that it cancels the pentest at 7am UTC instead of 8am:

```
h3 create-scheduled-action my-schedule '0 7 * * 5' cancel-pentest
```

Or if you wish to delete a scheduled action, use the `delete-scheduled-action` command:

```
h3 delete-scheduled-action my-schedule cancel-pentest
```

10. Creating multiple schedules

You can create multiple schedules by simply using a different schedule name. For example, here's a separate schedule named `my-weekend-schedule` that launches a pentest on Friday at 5pm UTC and cancels it Monday at 8am UTC:

```
h3 create-scheduled-action my-weekend-schedule '0 17 * * 5' run-pentest '{"schedule_op_form":{"op_name":"Weekend Pentest",
"runner_name":"my-nodezero-runner"}}'
h3 create-scheduled-action my-weekend-schedule '0 8 * * 1' cancel-pentest
```

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

Automate NodeZero Deployment

Auto-deploy NodeZero to a configured host using a NodeZero Runner

7.3.3 Automate Deployment Using a NodeZero Runner

Part of the process of running an internal pentest involves launching the NodeZero container on a host within your private network. This is typically done by manually signing in to your NodeZero Host and running the NodeZero Launch Script provided in the Portal.

A NodeZero Runner enables automated deployment of the NodeZero container. The NodeZero Runner is a background process running on your NodeZero Host that listens for newly provisioned pentests and runs the NodeZero Launch Script automatically. Both Docker and Podman are supported as the container runtime.

This allows you to provision and deploy pentests fully from the Portal, without having to manually sign in and run the NodeZero Launch Script. It also enables you to run pentests on an automated schedule.

A NodeZero Runner is used for internal pentests only. For external pentests, NodeZero is automatically deployed in the H3 cloud.

Once a NodeZero Runner is up and running, you can assign new pentests to it from the Portal (or via h3-cli). The Runner will launch NodeZero for the pentests assigned to it.

As your pentesting needs grow, you can spin up multiple NodeZero Runners across your network and assign pentests to each of them, enabling autonomous pentesting from various subnets and perspectives within your environment.

Installing and running a NodeZero Runner

There are two ways to install a Runner:

1. The easy way, using the new `easy_install.sh` script that does most of the work for you (recommended).
2. The slightly more manual way of walking thru the installation steps.

Install option #1: The easy way (recommended)

Follow the steps below to install a NodeZero Runner on your NodeZero Host.

1. Visit the Runners page in the Portal.
2. Click the **Install Runner** button, specify a name for the Runner, then click **Submit**.
3. Copy the provided installation command and run it on your NodeZero Host.

Barring any errors, **your NodeZero Runner is now up and running**. The Runner will register itself with the Portal and will be listed on the Runners page (you may need to refresh the page).

Auto-Restart: If your NodeZero Host runs on a Linux system that supports `systemd`, the install script will attempt to register the Runner as a system service with `systemd`. Once registered, `systemd` will automatically restart the Runner upon a system reboot. See Auto-start Runner at system startup for more info.

Behind the scenes, the above steps perform the following actions:

1. **Create an API key for the Runner.** The API key is granted **NodeZero Runner** permissions to the API. This is a specialized role created just for NodeZero Runners, with very restricted access to your account. The role **CANNOT** read existing pentest data, nor can it provision new pentests. The only thing it can do is poll the API to detect when a pentest has been assigned to it, and then run the NodeZero Launch Script.
2. **Install h3-cli on your NodeZero Host.** h3-cli is installed in a new `h3-cli` directory within the directory where you ran the installation command.
3. **Spin up the Runner using h3-cli.** The NodeZero Runner process is started via the `h3 start-runner` command.

Install option #2: The slightly more manual way

The steps below accomplish the same thing as install option #1, in a slightly more manual and transparent way.

1. CREATE API KEY FOR RUNNER

The NodeZero Runner communicates with the H3 API using h3-cli. As such, it requires an API key. API keys can be provisioned in the Portal here (or here for EU).

We recommend setting the permission level to **NodeZero Runner**. This is a specialized role created just for NodeZero Runners, with very restricted access to your account. The role **CANNOT** read existing pentest data, nor can it provision new pentests. The only thing it can do is poll the API to detect when a pentest has been assigned to it, and then run the NodeZero Launch Script.

2. INSTALL H3-CLI ON NODEZERO HOST

The NodeZero Runner process is started via the h3-cli. Therefore, the h3-cli must be installed on your NodeZero Host.

The quick-and-easy install steps are below. Simply copy+paste the commands into a shell on your NodeZero Host. For reference, full installation instructions are available here.

zip (Ubuntu / Debian) **tar.gz (RHEL / systems without unzip)**

```
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.zip -o h3-cli.zip
unzip h3-cli.zip -d h3-cli
cd h3-cli
bash install.sh {your-api-key-here}
export H3_CLI_HOME=`pwd`
export PATH="$H3_CLI_HOME/bin:$PATH"
```

```
curl -L https://downloads.horizon3ai.com/utilities/cli/h3-cli.tar.gz -o h3-cli.tar.gz
mkdir -p h3-cli && tar -xzf h3-cli.tar.gz -C h3-cli
cd h3-cli
bash install.sh {your-api-key-here}
export H3_CLI_HOME=`pwd`
export PATH="$H3_CLI_HOME/bin:$PATH"
```

The above steps will:

1. Download h3-cli from `downloads.horizon3ai.com`
2. Extract the archive
3. `cd` into the h3-cli dir
4. Run the install.sh script, substituting `{your-api-key-here}` with the API key you created in step 1 above.
5. Add h3-cli to your command `$PATH`

Once installed, run the command below to verify h3-cli is working and is using your newly provisioned API key:

```
h3 whoami
```

3. INSTALL RUNNER WITH H3-CLI

Use the following h3-cli command to spin up the NodeZero Runner:

```
h3 start-runner my-nodezero-runner /tmp/my-nodezero-runner.log
```

The NodeZero Runner process runs in the background and logs its output to the provided log file, in this case `/tmp/my-nodezero-runner.log`. The process is disconnected from the shell session, so it will continue to run in the background after you sign out.

Auto-Restart: If your NodeZero Host runs on a Linux system that supports `systemd`, you can optionally register the Runner as a system service with `systemd`. Once registered, `systemd` will automatically restart the Runner upon a system reboot. See Auto-start Runner at system startup for more info.

Naming: Each NodeZero Runner is given a name, in this case `my-nodezero-runner`. The name can be anything you want, but it should be unique if you spin up multiple Runners across your network. The name helps you identify the Runner when assigning pentests to it.

To verify the Runner has connected to the H3 API and registered itself, run the command below:

```
h3 runners
```

You should see an entry for the Runner `my-nodezero-runner` that you just started. You can now assign pentests to your NodeZero Runner from the Portal and the Runner will automatically launch NodeZero.

Additional Runner info

- **Runs as:** The Runner process runs as the user that invoked `h3 start-runner`.
- **Background process:** The Runner process is disconnected from the shell session and runs in the background. It continues to run after the shell session is closed.
- **Auto-Restart:** To enable auto-restart of the Runner on system reboot, see Auto-start Runner at system startup.
- **Runner Log**: To view the Runner log, use `h3 tail-runner {name}`
- **Stop Runner:** To terminate the Runner process, use `h3 stop-runner {name}`.
- **Delete Runner:** To delete a Runner, use `h3 delete-runner {name}`.
- This only deletes its registration record with H3. If a deleted Runner is still active, it will be re-registered upon its next heartbeat.
- **Unique Runner names:** Runner names should be treated as unique identifiers. Avoid re-using the same name for different Runners in your account.
- **Rename Runner:** You can NOT rename an existing Runner; however you can stop (and optionally delete) a Runner, then start a new Runner with a different name.
- **NOTE:** if you saved the old Runner name to an op template, the template will need to be updated to use the new Runner name.
- **Auto-Injecting Credentials:** Runners can also be used to auto-inject credentials into a pentest.

Troubleshooting

The NodeZero Runner polls the H3 API every 60s. The last polling time is reported in the output of `h3 runners`, in the `last_heartbeat_at` field. If the Runner's last heartbeat is more than a minute ago, then the Runner process has either terminated or lost connectivity to the H3 API. In which case, sign-in to your NodeZero Host and check the health and connectivity of the Runner.

Check if the Runner process is alive:

```
h3 ps-runner
```

This will list the Runner processes on the local machine.

Look for errors in the log:

```
h3 tail-runner {runner_name}
```

VIEW RUNNER COMMAND ERRORS

Use the following to list out the last five commands executed by the Runner. The output includes the exit status and output from the command:

```
h3 runner-commands {runner_name}
```

CONTAINER RUNTIME PERMISSION ERRORS

Example:

```
[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[!] FAILED: Failed to validate Docker. Verify this account has permissions to run Docker and retry.
```

If your NodeZero Host requires `sudo` to run container commands, then you may need to start the Runner using `sudo` as well.

Docker **Podman**

Try adding the user that invokes `h3 start-runner` to the `docker` group, for example (using `ubuntu` user):

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
```

(Note: Make sure to log out and back in after changing groups for the actively logged on user)

Podman always runs with `sudo` when used with NodeZero. Ensure the user running `h3 start-runner` has `sudo` privileges:

```
sudo podman run hello-world
```

209/STDOUT ERROR

This typically means `systemd` could not start the service because it did not have permission to write to the log file (`/tmp/my-nodezero-runner.log` in the examples above). Try chmod'ing the log file, or simply delete it and let `systemd` create a new one.

RETRY

After resolving issues with your Runner, you can retry a NodeZero deployment by directly queuing a request to the Runner using the following command. Substitute `{op_id}` and `{runner_name}` for your specific usage.

```
h3 run-nodezero-on-runner {op_id} {runner_name}
```

You can use `h3 pentest` to get the `op_id` for the most recently created pentest.

Auto-start Runner at system startup

A common use case is to register your Runner to start automatically at system startup. This ensures that your Runner will continue working even after the system is rebooted.

In order to do this you must register the Runner with the system's boot service. Different systems have different boot services, but the most popular one used by various Linux distributions is `systemd`.

h3-cli provides built-in support for `systemd`. To register your Runner with `systemd`, use the `h3 start-runner-service` command, example below. Note the command will attempt to use `sudo` for running the necessary `systemd`/`systemctl` commands.

```
h3 start-runner-service my-nodezero-runner /tmp/my-nodezero-runner.log
```

If your system uses a different boot service than `systemd`, contact us for assistance with setting up your NodeZero Runner service.

If all goes well, your NodeZero Runner is now registered as a service with `systemd`. This means the Runner will...

- automatically start up at boot time
- automatically be restarted if it fails for any reason

The Runner service will be registered with `systemd` under the name `nodezero-runner-{runner-name}`. For example the command above will register the service as `nodezero-runner-my-nodezero-runner`.

Helpful `systemctl` commands for managing the Runner service

To view the status of the Runner service, use `systemctl status`, for example:

```
systemctl status nodezero-runner-my-nodezero-runner
```

To stop the Runner service:

```
sudo systemctl stop nodezero-runner-my-nodezero-runner
```

Note this will stop the current Runner service, but it will not de-register it from `systemd`. This means the Runner service will be started again upon the next system boot. To disable the Runner service such that it no longer starts at boot time, use `systemctl disable`, for example:

```
sudo systemctl disable nodezero-runner-my-nodezero-runner
```

To re-register the Runner service and re-enable it to start at boot time:

```
sudo systemctl enable nodezero-runner-my-nodezero-runner
```

To start the Runner service:

```
sudo systemctl start nodezero-runner-my-nodezero-runner
```

For more information related to `systemd` and `systemctl`, check out these resources:

- How to use systemctl to manage Linux services
- `systemctl` man page
- `systemd` wiki

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

Auto-Inject Credentials

Inject credentials automatically into recurring tests

7.3.4 Automate Injecting Credentials

A common use case when running NodeZero involves injecting a known good credential into a pentest. Credentials are typically injected manually from the Real-Time View in the NodeZero Portal while the pentest is running. However, credentials may also be injected into a pentest that runs on an automated schedule.

Auto-injected credentials enable you to inject credentials automatically into a scheduled pentest. This support allows you to pre-configure the credentials you want to auto-inject. You then add those pre-configured credentials to your pentest configuration, and they will be auto-injected when the pentest starts.

This is particularly useful for running Password Audit pentests on an automated schedule, as they require a credential to be injected.

> **Tip**
>
> As an alternative to the scheduled option outlined on this page, you can inject credentials on a Runner assigned to an internal pentest.

How it works

At a high level, the steps for configuring an auto-injected credental are:

1. Install h3-cli and spin up a NodeZero Runner on your system (the Runner is what auto-injects the credential).
2. Use h3-cli to create an auto-injected credential.
3. Add the auto-injected credential to your pentest configuration/template.

That's all. The credential will be auto-injected by the Runner when the pentest starts.

STEP 1: INSTALL `h3-cli` AND DEPLOY A NODEZERO RUNNER

Installing h3-cli and spinning up a NodeZero Runner can be done with a single operation. See here for instructions.

You can install the `h3-cli` tool and deploy a NodeZero Runner in one streamlined step.
Follow the Automate NodeZero Deployment Guide to get started quickly.

STEP 2: CREATE AN AUTO-INJECTED CREDENTIAL

Use the h3-cli command `h3 create-auto-injected-credential` to create an auto-injected credential. For example:

```
h3 create-auto-injected-credential
Please provide JSON input on a single line:
{"key_type":"cleartext", "user":"myuser", "cleartext":"mypassword"}
```

If you see errors like `h3: command not found`, try adding the `h3-cli/bin` directory to your `PATH`, for example: `export PATH="/path/to/h3-cli/bin:$PATH"` (replace `/path/to` with the actual path on your system).

You can also pass the JSON input as a command-line parameter; however be aware **this could be less secure**, as the full command, including the credential parameter, will be recorded in your bash history.

There are several types of credentials you can create. Each type requires different parameters. Some examples are:

```
# Active Directory domain user credentials
{"key_type":"cleartext", "user":"myuser", "cleartext":"mypassword"}
{"key_type":"ntlm_hash", "user":"myuser", "hash":"myhash"}

# Local user credentials (tied to a specific IP)
{"key_type":"cleartext", "user":"myuser", "cleartext":"mypassword", "ip":"192.168.1.2"}
{"key_type":"ntlm_hash", "user":"myuser", "hash":"myhash", "ip":"192.168.1.3"}

# AWS credentials
{"key_type":"aws", "aws_access_key_id":"AK123456789ABCDEFGHI", "aws_secret_access_key":"DSAfR...BOl0c"}
{"key_type":"aws", "aws_access_key_id":"AK123456789ABCDEFGHI", "aws_secret_access_key":"DSAfR...BOl0c",
"aws_session_token":"IQoJb3JpZ2lu...BUTWASw="},

# Azure credentials
```

```
{"key_type": "azure","user": "user@example.com","cleartext": "mypassword"},
{"key_type":"azure-mfa","mfa_device_flow_id":"my-azure-tenant-id-uuid"}
```

See Injecting Credentials Guide for more information about injected credentials.

Security considerations

There are a number of security considerations to be aware of when using auto-injected credentials.

- **Stored on your system:** Auto-injected credentials are encrypted and stored locally in a file on the same machine as the h3-cli and Runner. The credential's existence is registered with H3; however H3 does NOT store the credential's secret (i.e. the cleartext password, hash, or aws key) on its systems, not even in encrypted form. The encrypted secret is stored on the local filesystem under the `~/.h3` directory (same location as the h3-cli API key).
- **Encrypted:** Auto-injected credentials are encrypted using an AES-256 key that H3 creates and secures on its backend systems.
- **Unique keys:** Each auto-injected credential has its own unique AES-256 key.
- **Re-encrypting a credential:** You can re-encrypt a credential by re-invoking `h3 create-auto-injected-credential`. A new AES-256 key is generated each time the command is executed.

These security measures were taken to ensure the security of your auto-injected credentials.

STEP 3: ADD THE AUTO-INJECTED CREDENTIAL TO YOUR PENTEST CONFIGURATION

You can add an auto-injected credential to your pentest configuration/template in the Portal under the **Run Pentest** wizard.

When the pentest is initiated, the Runner is notified and it auto-injects the configured credentials into the pentest. Some important notes to be aware of:

- **The Runner must be active in order to auto-inject credentials.** If the Runner is not active, the credentials will not be injected.
- **The Runner that auto-injects credentials does not have to be the same Runner that launches NodeZero.** Any Runner can auto-inject credentials into any pentest. For example, a Runner can auto-inject credentials into an external pentest, even though external pentests are not launched by Runners (external pentests are launched automatically in the H3 cloud).

This independence between (1) launching NodeZero, and (2) auto-injecting credentials, gives you some flexibility in how you manage your Runners and credentials. * **Multiple auto-injected credentials** can be added to the same pentest configuration. Similarly, the same auto-injected credential can be added to multiple pentest configurations. * **You can store multiple auto-injected credentials across multiple Runners in your environment.** H3 keeps track of which Runners have access to which credentials, in order to notify the correct Runner to auto-inject the credential when the pentest starts. * **Do NOT store the same auto-injected credential on multiple Runners.** A new AES-256 key is created each time `h3 create-auto-injected-credential` is invoked. Once you create the same auto-injected credential on a second Runner, the encrypted secret stored on the first Runner will no longer be valid and will fail to be decrypted and injected.

An auto-injected credential is "the same" as another if all non-secret input fields are identical (key_type, user, ip, aws_access_key_id)

Listing and deleting auto-injected credentials

You can use h3-cli to list all auto-injected credentials registered under your account:

```
h3 auto-injected-credentials
```

Use the following command to delete an auto-injected credential:

```
h3 delete-auto-injected-credential {uuid}
```

This will delete the credential's registration on the backend, remove the credential from any pentest templates it has been added to, and also notify the Runner to delete the encrypted file stored locally on the Runner's machine.

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

Monitor pentests

Track the progress and status of live tests

7.3.5 Monitor Pentests

You can use `h3-cli` to monitor the status of a pentest. This is done by periodically polling the API to check on the pentest `state`.

Monitoring enables you to trigger downstream actions or alerts when a pentest completes. A pentest is fully complete when its `state` is `done` or `ended`.

Example script to poll the pentest status until it completes.

```
#!/bin/bash

# loop forever until the pentest reaches 'done' or 'ended' state.
while [ 1 ]; do
    res=`h3 pentest`    # returns the most recent pentest
    pentest_state=`cat <<<$res | jq -r .state`
    pentest_name=`cat <<<$res | jq -r .name`
    if [ "$pentest_state" = "done" -o "$pentest_state" = "ended" ]; then
        echo "Pentest \"$pentest_name\" is complete; state=$pentest_state"
        break           # exit the loop
    fi
    echo "Pentest \"$pentest_name\" is still active; state=$pentest_state ..."
    sleep 30            # sleep 30 seconds then loop around and retry
done
```

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

Paginate results

Handle large result sets efficiently with pagination

7.3.6 Paginate Results

Queries that return potentially a lot of results can be paginated by using the optional `page_input` on the GraphQL request.

For example, click here to download sample graphql file, which is parameterized to accept `page_num` and `page_size` as inputs. These parameters are passed to `page_input` within the query file.

```
h3 gql action_logs '{"op_id":"your-op-id-here", "page_num":1, "page_size":100}' | jq .
```

The following shell script iterates through paginated results and stops when the query response indicates there are no more entries to retrieve.

```
#!/bin/bash

#
# Helper function for building the JSON parameters
# for the GraphQL request.
#
function build_json_params {
    op_id=$1
    page_num=$2
    page_size=$3
    cat <<HERE
{"op_id":"$op_id", "page_num":$page_num, "page_size":$page_size}
HERE
}

#
# 1. take the op_id as a param to the shell script.
#
op_id=$1
page_num=1
page_size=100

#
# 2. read page by page until the request returns no further results.
#
while [ 1 ]; do
    json_params=`build_json_params $op_id $page_num $page_size`
    res=`h3 gql action_logs "$json_params"`
    len=`cat <<<$res | jq '.data.action_logs_page.action_logs | length'`
    echo "Read $len records on page $page_num"
    if [ -z "$len" -o $len -eq 0 ]; then
        break
    fi
    (( page_num++ ))
done
```

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

Parse JSON results

Parse and transform pentest output

7.3.7 Parsing JSON

`jq` is a useful tool for parsing and transforming JSON responses. It is a powerful utility with many options, and is best learned through examples.

Examples

Copy + paste the following commands into your terminal to get an idea of what `jq` can do.

For example, to parse just the `op_id` field from the response:

```
h3 gql pentests | jq -r '.data.pentests_page.pentests[].op_id'
```

To parse just the `name` field from the response:

```
h3 gql pentests | jq -r '.data.pentests_page.pentests[].name'
```

To parse a set of fields from the response:

```
h3 gql pentests \
    | jq -r '.data.pentests_page.pentests[] | {op_id, name, scheduled_at, state}'
```

To parse a set of fields from the response and convert them from JSON to CSV format:

```
h3 gql pentests \
    | jq -r '.data.pentests_page.pentests[] | {op_id, name, scheduled_at, state}' \
    | jq -rsf $H3_CLI_HOME/filters/to_csv.jq
```

View the JSON response structure

It's sometimes useful to view the structure of the JSON response payload:

```
h3 gql pentests | jq -rf $H3_CLI_HOME/filters/to_struct.jq
```

Output:

```
.
.data
.data.pentests_count
.data.pentests_page
.data.pentests_page.pentests
.data.pentests_page.pentests[]
.data.pentests_page.pentests[].aws_account_ids
.data.pentests_page.pentests[].canceled_at
.data.pentests_page.pentests[].client_name
.data.pentests_page.pentests[].completed_at
.data.pentests_page.pentests[].credentials_count
.data.pentests_page.pentests[].data_resources_count
.data.pentests_page.pentests[].data_stores_count
.data.pentests_page.pentests[].duration_s
.data.pentests_page.pentests[].etl_completed_at
.data.pentests_page.pentests[].exclude_scope
.data.pentests_page.pentests[].exclude_scope[]
.data.pentests_page.pentests[].external_domains_count
.data.pentests_page.pentests[].git_accounts
.data.pentests_page.pentests[].hosts_count
.data.pentests_page.pentests[].impacts_count
.data.pentests_page.pentests[].launched_at
.data.pentests_page.pentests[].max_scope
.data.pentests_page.pentests[].max_scope[]
.data.pentests_page.pentests[].min_scope
.data.pentests_page.pentests[].name
.data.pentests_page.pentests[].nodezero_ip
.data.pentests_page.pentests[].nodezero_script_url
.data.pentests_page.pentests[].op_id
.data.pentests_page.pentests[].op_type
.data.pentests_page.pentests[].osint_company_names
.data.pentests_page.pentests[].osint_company_names[]
.data.pentests_page.pentests[].osint_domains
.data.pentests_page.pentests[].osint_keywords
.data.pentests_page.pentests[].osint_keywords[]
.data.pentests_page.pentests[].out_of_scope_hosts_count
.data.pentests_page.pentests[].services_count
.data.pentests_page.pentests[].state
```

```
.data.pentests_page.pentests[].user_name
.data.pentests_page.pentests[].users_count
.data.pentests_page.pentests[].weakness_types_count
.data.pentests_page.pentests[].weaknesses_count
.data.pentests_page.pentests[].websites_count
```

Select a field/list from the response

If you want to select only the `pentests` array from the response:

```
h3 gql pentests | jq '.data.pentests_page.pentests'
```

This behaves more like a traditional REST API, where responses are often structured as a flat array of JSON objects.

You can also drop the surrounding array brackets `[]` from the response and convert the output to a stream of JSON objects by adding `[]` to the filter:

```
h3 gql pentests | jq '.data.pentests_page.pentests[]'
```

You can then select a single field from the stream of JSON objects by adding it to the filter. For example if you want just the list of op_ids:

```
h3 gql pentests | jq -r '.data.pentests_page.pentests[].op_id'
```

Select a subset of fields from an object

If you want to select a subset of fields from the JSON objects in the `pentests` array:

```
h3 gql pentests | jq '.data.pentests_page.pentests[] | {op_id, name, state, scheduled_at}'
```

Convert a list of JSON objects to CSV

If you want to convert the `pentests` array to a CSV:

```
h3 gql pentests | jq '.data.pentests_page.pentests[]' | jq -rsf $H3_CLI_HOME/filters/to_csv.jq
```

The to_csv.jq filter will automatically convert lists and objects to JSON-encoded strings in the CSV.

Next Step

CLI Guides

Learn how to deploy, schedule pentests, monitor, and parse results.

8. API

8.1 NodeZero API

Horizon3.ai provides a publicly accessible Application Programming Interface (API), powered by GraphQL. This API exposes a subset of the functionality available in the Horizon3.ai Portal, making it ideal for automation, integration, and advanced workflows.

8.1.1 Features

- Programmatic control over assessments, targets, and scheduling
- Automate pentests with custom timing & scopes
- Retrieve results in structured JSON format
- Integrate easily into CI/CD pipelines
- Secure access via token-based authentication
- Query just what you need using GraphQL syntax

8.1.2 Tools & Documentation

Getting Started with GQL

Explore the full GraphQL schema, including available queries and mutations.

API Reference

Explore the full GraphQL schema, including available queries and mutations.

8.2 Getting Started

8.2.1 Getting Started with GraphQL

This section explains our motivation for using GraphQL, and provides examples of common scenarios to help you get started using the NodeZero API. This will help accelerate your understanding of how to use the GraphQL API to interact with **your** data.

Authentication

Before getting started with the examples here, be sure to review the section on API Authentication.

Leverage the full power

Once you are comfortable interacting with the GraphQL API, head over to our API Reference for a complete list of queries and mutations available.

Why GraphQL?

The ITOps, SecOps, and automations engineering worlds are filled with REST APIs. So why did we choose a GraphQL API?

Data is power. We have a good idea how you might be able to use it, but it would be wrong of us to believe that we know better than you. GraphQL gives our developers far more control and flexibility through the ability to shape API queries to meet their needs. We wanted to provide the same power to our API users.

GraphQL.org

If you are new to GraphQL, we highly recommended reviewing the learning material at GraphQL.org.

HTTP Client

To send requests to the GraphQL API, an HTTP client is needed. This is mostly a personal choice and/or dependent on the systems you may be integrating with. For example, if you are trying to build your own HTML interface, you will likely need a client that runs in JavaScript to be compatible with internet browsers.

Here are a few examples of useful HTTP clients:

- curl - useful for command line applications
- Python requests - useful for building Python applications
- Altair GraphQL Client - useful for testing and building API requests
- Postman API Client - useful for testing and building API requests

Need a specific programming language?

HTTP clients exist for every major programming language. A quick internet search should reveal the most well-known HTTP client(s) in any programming language.

Command Line Interface

Also be sure to check out the Horizon3.ai CLI for a ready-to-use API client.

STATUS CODES

HTTP status codes returned by the `/graphql` endpoint are summarized below.

Code	Title	Description
`200`	Success	Response contains the requested data.
`400`	Bad Request	Malformed request, see `errors` field in the response data.
`401`	Unauthorized	Not authenticated due to an invalid or expired JWT.
`403`	Forbidden	Not authorized to access the requested resource.
`5xx`	Internal Server Error	An unexpected error was encountered.

Next Step

1 Authentication

Learn how to authorize your session with graphql queries

2 GraphQL Examples

Browse the exmples of the most common types of queries

8.2.2 API Authentication

Before sending requests to the GraphQL API, you must authenticate. You do this by obtaining a JWT from our `/auth` endpoint, using an API key.

Need an API key?

Users can obtain an API key from the Horizon3.ai Portal.

Authenticate

Below are examples for obtaining an authenticated JWT. Each example sends an HTTP request to the Horizon3.ai auth endpoint, which gets referenced via the environment variable `H3_AUTH_URL`:

US-based Portal EU-based Portal

```
export H3_AUTH_URL=https://api.gateway.horizon3ai.com/v1/auth
```

```
export H3_AUTH_URL=https://api.gateway.horizon3ai.eu/v1/auth
```

Store

Before proceeding, store your API key in environment variable `H3_API_KEY`:

```
export H3_API_KEY=<your-api-key>
```

curl Python

```
curl -X POST \
     -H "Content-Type: application/json" \
     -d '{"key": "'"$H3_API_KEY"'"}' \
     $H3_AUTH_URL
```

```
import os
import requests

url = os.environ["H3_AUTH_URL"]
response = requests.post(url, json={"key": os.environ["H3_API_KEY"]})
result = response.json() if response.status_code == 200 else None
```

Send

You can use the resulting token to send authenticated requests to the GraphQL API.

Result

```
{"token": "<your-api-token>"}
```

JWT expiration

Each JWT expires after **one hour**, at which point the `/graphql` endpoint will respond with the token expiration message shown below. If this occurs, obtain a new JWT by re-authenticating with the `/auth` endpoint.

```
{"message":"The incoming token has expired"}
```

Status codes

HTTP status codes returned by the `/auth` endpoint are summarized below.

Code	Title	Description
`200`	Success	Response contains the JWT.
`400`	Bad Request	Malformed request, review the error message.
`401`	Unauthorized	Not authenticated due to invalid API key.
`5xx`	Internal Server Error	An unexpected error was encountered.

Next Step

GraphQL Examples

Browse the exmples of the most common types of queries

8.2.3 GraphQL Examples

The examples here send HTTP requests to the Horizon3.ai GraphQL API endpoint, which gets referenced in the commands below via environment variable `H3_API_URL`:

US-based Portal **EU-based Portal**

```
export H3_API_URL=https://api.gateway.horizon3ai.com/v1/graphql
```

```
export H3_API_URL=https://api.gateway.horizon3ai.eu/v1/graphql
```

Before proceeding, first perform API authentication and store the JWT in environment variable `H3_API_JWT`:

```
export H3_API_JWT=<token-returned-from-auth>
```

Hello World

Let's start with the simple hello (world) query.

> For documentation of this query, see API Reference for Queries > hello.

curl **Python**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d '{"query": "query HelloWorld { hello }"}'
```

```
import os
import requests

query = '''
    query HelloWorld {
      hello
    }
'''
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query})
result = response.json() if response.status_code == 200 else None
```

Result

```
{
  "data": {
    "hello": "world!"
  }
}
```

Pentest data

Let's review how to fetch data associated with a given pentest, such as start/stop times, number of weaknesses found, credentials discovered, and hosts scanned.

A lot more data can be queried than what this example shows. For detailed documentation, see API Reference for Queries > pentest.

curl **Python** **h3-cli**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetPentest(\$op_id: String!) {
      pentest(op_id: \$op_id) {
        op_id
        op_type
        name
        state
        launched_at
        completed_at
        min_scope
        max_scope
        exclude_scope
        weaknesses_count
        credentials_count
        cred_access_count
      }
    }",
  "variables": {
    "op_id": "12341234-1234-1234-1234-123412341234"
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetPentest($op_id: String!) {
      pentest(op_id: $op_id) {
        op_id
        op_type
        name
        state
        launched_at
        completed_at
        min_scope
        max_scope
        exclude_scope
        weaknesses_count
        credentials_count
        cred_access_count
      }
    }
'''
variables = {
    "op_id": "12341234-1234-1234-1234-123412341234"
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

```
h3 pentest <op-id>
```

The command above runs the GraphQL query defined in `h3-cli/queries/pentest.graphql`.

You can also run your own custom GraphQL queries and mutations using h3-cli. Save the following query to a file named `my_pentest.graphql`.

```
query GetPentest($op_id: String!) {
  pentest(op_id: $op_id) {
    op_id
    op_type
    name
    state
    launched_at
    completed_at
    min_scope
    max_scope
    exclude_scope
    weaknesses_count
    credentials_count
    cred_access_count
  }
}
```

Then run the query using the following command:

```
h3 gql ./my_pentest.graphql '{"op_id":"<op-id>"}'
```

Result


```
{
  "data": {
    "pentest": {
      "op_id": "12341234-1234-1234-1234-123412341234",
      "op_type": "NodeZero",
      "name": "Sample Pentest",
      "state": "done",
      "launched_at": "2022-04-25T14:41:18",
      "completed_at": "2022-04-25T15:23:21",
      "min_scope": null,
      "max_scope": null,
      "exclude_scope": [],
      "weaknesses_count": 53,
      "credentials_count": 33,
      "cred_access_count": 142
    }
  }
}
```


Need more or less data?

One of the benefits of GraphQL is the ability to query only the data you need. In this example, add or remove fields of the Pentest type to suit your needs. This avoids the issue of over-fetching and eliminates the difficulty of parsing the returned data.

Pentest reports

Let's review how to download a ZIP file containing CSV data files and PDF reports associated with a given pentest. This approach to fetching data from the API is in contrast to the granular approach demonstrated in the previous section on querying pentest data.

For documentation on this query, see the API Reference for Queries > pentest_reports_zip_url.

To download individual reports, see the API Reference for List Reports and related endpoints.

curl **Python**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetPentestReports(\$input: OpInput!) {
      pentest_reports_zip_url(input: \$input)
    }",
  "variables": {
    "input": {
      "op_id": "12341234-1234-1234-1234-123412341234"
    }
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetPentestReports($input: OpInput!) {
      pentest_reports_zip_url(input: $input)
    }
'''
variables = {
    "input": {
        "op_id": "12341234-1234-1234-1234-123412341234"
    }
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

Result

```
{
  "data": {
    "pentest_reports_zip_url": "<temporary-url-to-download-zip>"
  }
}
```

Retrieve all pentests

This query illustrates how to retrieve all pentests run in your client account, with the results ordered by most-recent.

curl Python

```
H3_GQL_URL="https://api.gateway.horizon3ai.com/v1/graphql"
H3_JWT="<your-jwt>"
curl -X POST $H3_GQL_URL \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer $H3_JWT" \
    -d @- <<HERE
{
  "query": "
    query q(\$page_input: PageInput) {
      pentests_count(page_input: \$page_input)
      pentests_page(page_input: \$page_input) {
        pentests {
          ...PentestFragment
        }
      }
    }
    fragment PentestFragment on Pentest {
      op_id
      op_type
      name
      state
      scheduled_at
      launched_at
      completed_at
      etl_completed_at
      hosts_count
      weaknesses_count
    }
  ",
  "variables": {
    "page_input": {
      "page_num": 1
    }
  }
}
HERE
```

```
import os
import requests

query = '''
    query q($page_input: PageInput) {
      pentests_count(page_input: $page_input)
      pentests_page(page_input: $page_input) {
        pentests {
          ...PentestFragment
        }
      }
    }
    fragment PentestFragment on Pentest {
      op_id
      op_type
      name
      state
      scheduled_at
      launched_at
      completed_at
      etl_completed_at
      hosts_count
      weaknesses_count
    }
'''
variables = {
    "page_input": {
      "page_num": 1
    }
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json()
```

Result

```
{
  "data": {
    "pentests_count": 1,
    "pentests_page": {
```

```
        "pentests": [
          {
            "op_id": "12341234-1234-1234-1234-123412341234",
            "op_type": "NodeZero",
            "name": "Sample Pentest",
            "state": "done",
            "scheduled_at": "2025-03-06T20:37:31.973429",
            "launched_at": "2025-03-06T20:44:22.567519",
            "completed_at": "2025-03-06T21:22:33.893341",
            "etl_completed_at": "2025-03-06T21:30:01.073779",
            "hosts_count": 10,
            "weaknesses_count": 5
          }
        ]
      }
    }
  }
}
```


Paginate action logs

Some queries fetch lists of data, e.g., users, pentests, actions logs, etc., but the amount of data can be substantial. Pagination allows us to limit the quantity of data retrieved. Let's take a look at how this works, by querying the two (2) most recent action logs for a given pentest.

For documentation of this query, see API Reference for Queries > action_logs_page.

curl **Python**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    query GetActionLogs(\$input: OpInput!, \$page_input: PageInput) {
      action_logs_page(input: \$input, page_input: \$page_input) {
        page_info {
          ...PageInfoFragment
        }
        action_logs {
          ...ActionLogFragment
        }
      }
    }

    fragment PageInfoFragment on PageInfo {
      page_num
      page_size
    }

    fragment ActionLogFragment on ActionLog {
      uuid
      endpoint_ip
      start_time
      end_time
      cmd
      module_id
      module_name
      module_description
    }
  ",
  "variables": {
    "input": {
      "op_id": "12341234-1234-1234-1234-123412341234"
    },
    "page_input": {
      "page_num": 1,
      "page_size": 2,
      "order_by": "start_time",
      "sort_order": "DESC"
    }
  }
}
HERE
```

```
import os
import requests

query = '''
    query GetActionLogs($input: OpInput!, $page_input: PageInput) {
      action_logs_page(input: $input, page_input: $page_input) {
        page_info {
          ...PageInfoFragment
        }
        action_logs {
          ...ActionLogFragment
        }
      }
    }

    fragment PageInfoFragment on PageInfo {
      page_num
      page_size
    }

    fragment ActionLogFragment on ActionLog {
      uuid
      endpoint_ip
      start_time
      end_time
      cmd
      module_id
      module_name
      module_description
    }
'''
variables = {
    "input": {
        "op_id": "12341234-1234-1234-1234-123412341234"
    },
    "page_input": {
        "page_num": 1,
        "page_size": 2,
        "order_by": "start_time",
        "sort_order": "DESC"
    }
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

Result


```
    {
      "data": {
        "action_logs_page": {
          "page_info": {
            "page_num": 1,
            "page_size": 2
          },
          "action_logs": [
            {
              "uuid": "12341234-1234-1234-1234-123412341234/20307",
              "endpoint_ip": "10.0.255.10",
              "start_time": "2020-09-30T21:41:22",
              "end_time": "2020-09-30T21:41:22",
              "cmd": "rpcclient 10.0.255.10 -c enumdomusers -U xadmin%L********n",
              "module_id": "List Users Over RPC",
              "module_name": "List Users Over RPC",
              "module_description": "The List Users Over Remote Procedure Call (RPC) utilizes rpcclient and a user credential discovered
during an operation to dump all of the domain users."
            },
            {
              "uuid": "12341234-1234-1234-1234-123412341234/20308",
              "endpoint_ip": "10.0.255.10",
              "start_time": "2020-09-30T21:41:22",
              "end_time": "2020-09-30T21:41:23",
              "cmd": "net rpc group members Administrators -I 10.0.255.10 -U SMOKE.NET\\xadmin%L********n",
              "module_id": "Enumerate Admins Over RPC",
              "module_name": "Enumerate Admins Over RPC",
              "module_description": "The Enumerate Admins Over RPC module uses network Remote Procedure Call (RPC) to determine the
administrative accounts available on an endpoint and also, if the host is a Domain Controller (DC), will try to gather the Domain Admin
accounts."
            }
          ]
        }
      }
    }
```


GraphQL fragments

You may have noticed that this example used fragments to succinctly define fields to query on a given type. To learn more, see GraphQL fragments from graphql.org.

Create internal pentest

Let's consider how we can create a new **internal pentest** via the API. This will require us to send a **mutation** to the GraphQL API, instead of a **query**. Mutations are used when creating, updating, or deleting resources, whereas queries are intended only for fetching existing resources.

Creating versus launching pentest

Creating a pentest is the process of configuring and preparing a pentest to be launched. Launching a pentest is a separate process – handled **after** creating the pentest, by deploying NodeZero in your environment. This is explained further in the next section.

For documentation of this query, see API Reference for Mutations > create_op.

curl **Python**

```
curl \
  -X POST $H3_API_URL \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $H3_API_JWT" \
  -d @- <<HERE
{
  "query": "
    mutation CreatePentest(
      \$op_name: String!
      \$op_type: OpType!
    ) {
      create_op(
        schedule_op_form: {
          op_name: \$op_name
          op_type: \$op_type
        }
      ) {
        op {
          ...OpFragment
        }
      }
    }

    fragment OpFragment on Op {
      op_id
      op_name
      op_state
      op_type
      scheduled_at
      launched_at
      nodezero_script_url
    }
  ",
  "variables": {
    "op_name": "Pentest created via API",
    "op_type": "NodeZero"
  }
}
HERE
```

```
import os
import requests

query = '''
    mutation CreatePentest(
      $op_name: String!
      $op_type: OpType!
    ) {
      create_op(
        schedule_op_form: {
          op_name: $op_name
          op_type: $op_type
        }
      ) {
        op {
          ...OpFragment
        }
      }
    }

    fragment OpFragment on Op {
      op_id
      op_name
      op_state
      op_type
      scheduled_at
      launched_at
      nodezero_script_url
    }
'''
variables = {
    "op_name": "Pentest created via API",
    "op_type": "NodeZero"
}
url = os.environ["H3_API_URL"]
headers = {"Authorization": f"Bearer {os.environ['H3_API_JWT']}"}
response = requests.post(url, headers=headers, json={"query": query, "variables": variables})
result = response.json() if response.status_code == 200 else None
```

Result


```
{
  "data": {
    "create_op": {
      "op": {
        "op_id": "df087532-b6ca-48b1-bcd3-d8b1968a197f",
        "op_name": "Pentest created via API",
        "op_state": "scheduled",
        "op_type": "NodeZero",
        "scheduled_at": "2023-03-04T16:19:42",
        "launched_at": null,
        "nodezero_script_url": <url-to-download-launch-script>
      }
    }
  }
}
```


In example above, the **default** template, **Default 1 - Recommended**, was used to define the pentest configuration. To create your own op template, Horizon3.ai recommends that you do so from the NodeZero Portal, but you may also use the API with Mutations > `create_op_template`.

What is an op template?

An "op template" refers to a predefined pentest configuration, including settings for hosts to scan, passwords to spray, and other attack config. You may create op templates for various use cases and environments. Each op template gets stored in your client account and can be used when creating pentests.

Ready for launch!

Continue reading to see how our newly created pentest can be **launched**. (In the result above, note that `launched_at` has a `null` value, indicating that the pentest has yet to launch.)

LAUNCH NODEZERO

Once an internal pentest has been **created**, it is ready for NodeZero to deploy in the intended environment. To deploy the pentest, simply copy the value returned in `nodezero_script_url` from the mutation above, then pass it to curl and pipe the downloaded NodeZero script to bash:

NodeZero host only

The launch script must be run from the NodeZero host inside your intended environment. For more information on configuring this host, see Set Up NodeZero Host.

```
curl <nodezero_script_url> | bash
```

Once the above command successfully runs, the autonomous pentest has officially deployed. To monitor its progress and notable events, go to the pentest in the Horizon3.ai Portal and select **Real-Time View**.

SCHEDULING PENTESTS

See the Setup -> Schedules section of the API Reference to learn how to set up schedules for running pentests on a regular basis, for example, weekly or monthly.

Next Step

API Reference

Explore the graphql schema and create your own queries to run.

×

INTRODUCTION

- Welcome
- GraphQL vs. REST
- Authentication
- Sending Requests
- Response Format
- Handling Errors
- Versioning
- h3-cli Examples
- Python Examples
- cURL Examples

USING THE API

- Pagination
- type PageInput
- type FacetValue
- type FilterBy
- type FilterByInput
- type FilterByInputMode
- type PageInfo
- type SortInput
- type SortOrder
- Metadata
- Type Definitions
- Hello World
- type GraphQLDef

SETUP

- Op Config
- type ScheduleOpForm
- type ScheduleOpFormInput
- type AWSAccountId
- type AttackConfigValue
- type AttackConfigValueInput
- type FeatureFlag
- type FeatureFlagInput
- type FeatureFlagRiskType
- type GitAccount

- type GitAccountInput
- type GitAccountSource
- type OpInput

- **Op Templates**

- List Op Templates
- Count Op Templates
- Read an Op Template
- Create an Op Template
- Update an Op Template
- Update Op Templates
- Delete an Op Template
- Delete an Op Template
- Delete Op Templates
- Default Op Templates
- Op Template Facets
- type OpTemplate
- type DeleteOpTemplateOutput
- type OpTemplatesFacets
- type OpTemplatesPage
- type SaveOpTemplateOutput
- type UpdateOpTemplateInput
- type UpdateOpTemplateOutput

- **Runners**

- List Runners
- Count Runners
- Read a Runner
- Create a Runner
- Ping a Runner
- Upgrade a Runner
- Delete a Runner
- List K8s Operators
- Count K8s Operators
- Create an Operator
- Upgrade an Operator
- Delete an Operator
- Runner Facets
- Agent Command Facets
- K8s Operator Facets
- type Agent
- type Operator
- type AgentCommand
- type AgentCommandOutput

- type AgentCommandsFacets
- type AgentCommandsPage
- type AgentInput
- type AgentOutput
- type AgentsFacets
- type AgentsPage
- type CreateOperatorOutput
- type InstallAgentOutput
- type OperatorsFacets
- type OperatorsPage
- type UpgradeOperatorOutput

- **Schedules**

- List Schedules
- Count Schedules
- Read a Schedule
- Create a Scheduled Action
- Trigger a Scheduled Action
- Update a Scheduled Action
- Delete a Scheduled Action
- Enable a Schedule
- Disable a Schedule
- type Schedule
- type ScheduledAction
- type CreateScheduledActionInput
- type DeleteScheduledActionInput
- type ScheduleOutput
- type ScheduledActionOutput
- type SchedulesPage
- type TriggerScheduledActionInput
- type UpdateScheduledActionInput

- **Asset Groups**

- List Asset Groups
- Count Asset Groups
- Read an Asset Group
- Create an Asset Group
- Update an Asset Group
- type AssetGroup
- type AssetGroupOutput
- type AssetGroupsPage

- **Asset Groups: Domains**

- List Domains
- Count Domains

- Read a Domain
- Add Domains
- Remove Domains
- Domains CSV
- type ExternalDomainXop
- type ExcludedDomain
- type ExcludedReason
- type ExternalDomainXopsPage

- **Asset Groups: IPs**

- List IPs
- Count IPs
- Read an IP
- List In-Scope IPs
- Count In-Scope IPs
- In-Scope Hosts CSV
- type HostTabXop
- type ExcludedIP
- type HostTabXopsPage

- **Asset Groups: Authorization**

- Authorize Domains
- Deauthorize Domains
- Bulk Authorize Domains
- Bulk Deauthorize Domains
- Authorize IPs
- Deauthorize IPs
- Bulk Authorize IPs
- Bulk Deauthorize IPs
- type PentestableEntity
- type PentestableRules
- type BlockedPentestableEntity
- type PentestableEntitiesBulkOutput
- type PentestableEntitiesOutput

PENTESTING

- **Ops**

- List Ops
- Count Ops
- Read an Op
- Create an Op
- Pause an Op
- Resume an Op
- End an Op

- Update an Op
- Rerun an Op
- List Reports
- Read a Report
- Report URL
- List Sample Ops
- Read a Sample Op
- type Op
- type OpType
- type PortalOpState
- type CreateOpOutput
- type OneClickVerifyData
- type OpReport
- type OpTab
- type OpTabsPage
- type OpsPage
- type ReportType
- type ScheduleOpOutput
- type UpdateOpInput
- type UpdateOpOutput
- **Pentests**
- List Pentests
- Count Pentests
- Read a Pentest
- Download Reports
- type Pentest
- type PentestsPage
- **Impacts**
- List AttackPaths
- Count AttackPaths
- Read an AttackPath
- Attack Paths CSV
- type AttackPath
- type AttackVector
- type AttackGraph
- type Node
- type Edge
- type AttackPathsPage
- type AttackVectorAlgorithm
- type AttackVectorAlgorithmInput
- type ImpactType

- **Weaknesses**
- List Weaknesses
- Count Weaknesses
- Read a Weakness
- Weaknesses CSV
- Create 1CV Pentests
- type Weakness
- type ScheduleWeakness1CVOpsFailureOutput
- type ScheduleWeakness1CVOpsInput
- type ScheduleWeakness1CVOpsOutput
- type Severity
- type ThreatActor
- type ThreatActorReference
- type Vuln
- type VulnCategory
- type VulnImpactCategory
- type VulnMitigation
- type VulnReference
- type WeaknessesPage
- **Proofs**
- type Proof
- type CommandOutput
- type EnumerationProofOutput
- **Credentials**
- List Credentials
- Count Credentials
- Read a Credential
- Credentials CSV
- AD Password Audit CSV
- type Credential
- type CloudRole
- type CredentialsPage
- **Data Stores**
- List DataStores
- Count DataStores
- Read a DataStore
- Data Stores CSV
- type DataStore
- type DataStoresPage
- type InterestingResource

- **Hosts**
- List Hosts
- Count Hosts
- Read a Host
- Hosts CSV
- type Host
- type HostsPage
- **EDR**
- EDR Events CSV
- **External Domains**
- type ExternalDomain
- **LDAP Domains**
- type LDAPDomain
- **Cloud Resources**
- type CloudResource
- **Applications**
- type Application
- **Services**
- type Service
- **Action Logs**
- List Action Logs
- Count Action Logs
- Action Logs CSV
- type ActionLog
- type ActionLogsPage
- type MitreMapping
- type MitreSubtechnique
- type MitreTactic
- type MitreTechnique
- type ModuleInstance
- type ModuleMeta
- **Action Log V2**
- List Action Log V2 Events
- Count Action Log V2 Events
- type ActionLogV2Event
- type ActionLogV2EventsPage
- **Op Compare**
- List Weakness Diff
- Weakness Diff Stats
- Weakness Diff CSV

- type DiffStats
- type DiffStatus
- type OpDiffInput
- type WeaknessesDiffPage

VULN MANAGEMENT HUB

- **Weakness Series**
- List Weakness Series
- Count Weakness Series
- List Annotations
- Count Annotations
- Apply Annotations
- Remove Annotations
- Tickets CSV
- Weakness Series Metadata
- Weakness Series Facets
- Weakness Series CSV
- Count Attack Paths
- Create 1CV Pentests
- type WeaknessSeries
- type Annotation
- type AnnotationInput
- type AnnotationStatus
- type ApplyAnnotationOutput
- type ScheduleWeaknessSeries1CVOpsFailureOutput
- type ScheduleWeaknessSeries1CVOpsInput
- type ScheduleWeaknessSeries1CVOpsOutput
- type Tag
- type WeaknessSeriesAnnotation
- type WeaknessSeriesAnnotationsPage
- type WeaknessSeriesFacets
- type WeaknessSeriesMetadata
- type WeaknessSeriesOpStatus
- type WeaknessSeriesPage
- **Assets**
- List Assets
- Count Assets
- Download Assets CSV
- type Asset
- type AssetsPage
- **Perspectives**
- List Perspectives

- Count Perspectives
- Read a Perspective
- Read the Org Perspective
- Perspectives Up Next
- type Perspective
- type PerspectivesPage

- **Campaigns**

- List Campaigns
- Read a Campaign
- Create a Campaign
- Update a Campaign
- Delete a Campaign
- List Available Perspectives
- Count Available Perspectives
- type Campaign
- type CampaignScope
- type CreateCampaignInput
- type RecommendedPerspective
- type UpdateCampaignInput

PLATFORM FEATURES

- **Rapid Response**

- List RR Notifications
- Count RR Notifications
- Read RR Notification
- Apply Annotations to RR Assets
- type ApplyAnnotationsToRapidResponseAssetsInput
- type Notification
- type NotificationData
- type NotificationType
- type NotificationsPage

- **Tripwires**

- List Tripwires
- Tripwires CSV Presigned URL
- Count Tripwires
- List Tripwire Jobs
- Count Tripwire Jobs
- Test a Tripwire
- type Tripwire
- type FileAttributes
- type Trip
- type TripAttributes

- type TripwireState
- type TripwireType
- type TripwiresPage

- **Vuln Risk Intelligence**

- List Scanner Files
- Count Scanner Files
- Read a Scanner File
- Scanner Classifications CSV
- Upload Scanner File
- Delete Scanner Files
- Process Scanner File
- Update Scanner File
- type DeleteScannerFilesInput
- type DeleteScannerFilesOutput
- type ProcessScannerFileInput
- type ProcessScannerFileOutput
- type ScannerCategory
- type ScannerCategoryBreakdown
- type ScannerCategoryCount
- type ScannerClassification
- type ScannerClassificationFacets
- type ScannerClassificationPage
- type ScannerClassificationsSummary
- type ScannerDisplayCategory
- type ScannerDisplayCategoryCount
- type ScannerDownstreamImpactCount
- type ScannerFile
- type ScannerFileFormat
- type ScannerFileStatus
- type ScannerFileVendor
- type ScannerFilesPage
- type ScannerSeverity
- type ScannerSeverityCount
- type ScannerSourceBreakdown
- type UpdateScannerFileInput
- type UpdateScannerFileOutput
- type UploadScannerFileInput
- type UploadScannerFileOutput

- **CSV Format**

- type ActionLogCSV
- type ActiveDirPasswordCSV
- type CertificateCSV

- type CredentialCSV
- type DatabaseRepoCSV
- type DockerRegistryCSV
- type EDROverviewHostCSV
- type ExternalDomainCSV
- type ExternalDomainXopCSV
- type FileShareCSV
- type GitRepoCSV
- type HostCSV
- type HostXopCSV
- type ImpactCSV
- type RapidResponseAssetCSV
- type S3BucketCSV
- type ScannerClassificationCSV
- type ServiceCSV
- type ShareCSV
- type TicketCSV
- type TripwireCSV
- type UserCSV
- type WeaknessCSV
- type WeaknessSeriesCSV
- type WebShareCSV

- **Inventory**

- Inventory Assets CSV
- Inventory IP Ranges CSV
- Inventory Scopes CSV
- Initiate Bulk Inventory Upload
- Process Bulk Inventory Upload
- Create Inventory CIDR
- Update Inventory CIDR
- Delete Inventory CIDR
- Delete Inventory CIDRs
- Request Inventory Assets CSV Export
- Request Inventory IP Ranges CSV Export
- Update Inventory Domains
- Delete Inventory Domain
- Delete Inventory Domains
- Authorize All Owned Domains
- Update Inventory IPs
- Delete Inventory IPs
- Authorize All Owned IPs
- Assign Assets to Scope

- Unassign Assets from Scope
- Create Inventory Op Scope
- Update Inventory Op Scope
- Update Inventory Op Scopes
- Delete Inventory Op Scopes
- type BulkInventoryUploadStatus
- type BulkUpdateResult
- type CreateInventoryCidrInput
- type CreateInventoryOpScopeInput
- type InitiateBulkInventoryUploadInput
- type InitiateBulkInventoryUploadOutput
- type InventoryAssetsCsvInput
- type InventoryCidr
- type InventoryCidrDeleteOutput
- type InventoryCidrsPage
- type InventoryDomain
- type InventoryDomainDeleteOutput
- type InventoryDomainDetail
- type InventoryDomainsPage
- type InventoryExportType
- type InventoryIP
- type InventoryIpDeleteOutput
- type InventoryIpRangesCsvInput
- type InventoryOpScope
- type InventoryOpScopeDeleteOutput
- type InventoryScopesCsvInput
- type ProcessBulkInventoryUploadInput
- type ProcessBulkInventoryUploadOutput
- type ScopeDiscoveriesPage
- type UpdateInventoryCidrInput
- type UpdateInventoryDomainBatchInput
- type UpdateInventoryIpInput
- type UpdateInventoryOpScopeInput

ADMIN

- **Account Management**
- Current User
- Count Client Accounts
- List Client Accounts
- type ClientAccount
- type UserAccount
- type APIKey
- type AuthzRole

- type BrandColor
- type BrandColorType
- type ClientAccountsPage
- type HexColor
- type SignInType

MSP

- Account Summaries
- User Summaries
- Attack Path Summaries
- Weakness Summaries
- CPE Strings
- Overall Summary
- type ClientAccountCPEString
- type ClientAccountImpactSummary
- type ClientAccountUserSummary
- type ClientAccountWeaknessSummary
- type OverallSummarySubclientAccounts
- type SubclientAccountSummary

TYPES

- Boolean
- Date
- Datetime
- EmailAddress
- Float
- ID
- Int
- JSONObject
- Long
- String
- StringNoWhitespace
- StringNotEmpty

8.3 GraphQL API Reference

Welcome to the Horizon3.ai GraphQL API Reference. This guide covers every query, mutation, and type available in the API, along with everything you need to start making requests.

GETTING STARTED

- GraphQL vs. REST — Key differences if you are coming from a REST background.
- Authentication — Obtain a token using your API key.

- Sending Requests — Make GraphQL requests to the API.
- Response Format — Understand the structure of API responses.
- Handling Errors — Interpret error responses.
- Versioning — Learn how the API evolves over time.

CLIENT EXAMPLES

- h3-cli Examples — Use the Horizon3.ai CLI to run queries from the command line.
- Python Examples — Run queries with a ready-to-use Python script.
- cURL Examples — Run queries with a ready-to-use bash/cURL script.

8.3.1 GraphQL vs. REST

If you have worked with REST APIs before, GraphQL will feel familiar — you are still sending HTTP requests and receiving JSON responses. However, there are a few important differences to be aware of.

SINGLE ENDPOINT

REST APIs expose many endpoints (e.g., `/pentests`, `/pentests/:id`, `/assets`). GraphQL uses a **single endpoint** (`/graphql`) for all operations. The request body determines what data is returned.

REQUESTS ARE DEFINED IN THE POST BODY

Instead of encoding the operation in the URL path and HTTP method (GET, PUT, DELETE), GraphQL requests are always sent as an **HTTP POST** with a JSON body containing a `query` field. The query is a structured string that describes exactly which fields and relationships you want to retrieve. See Sending Requests for an example.

YOU CHOOSE THE FIELDS YOU RECEIVE

With a REST API, each endpoint returns a fixed set of fields. GraphQL lets you specify exactly which fields to include in the response — no more, no less. This reduces payload sizes and eliminates unnecessary data transfer.

FETCH RELATED DATA IN A SINGLE REQUEST

REST often requires multiple round-trips to assemble related data (e.g., fetching a pentest, then its weaknesses, then each affected host). A GraphQL query can traverse deeply nested relationships in a single request, returning all the data you need at once.

STATUS CODES WORK DIFFERENTLY

REST APIs use HTTP status codes (200, 404, 500, etc.) to indicate success or failure. GraphQL requests **always return HTTP 200**, even when errors occur. Errors are reported in the `errors` field of the response body. See Handling Errors for details.

STRONGLY TYPED SCHEMA

Every field, argument, and return type in the API is defined in a typed schema. This reference is generated directly from that schema, so it always reflects the current state of the API.

REST (Multiple Requests)

```
# Fetch a list of pentests
GET /v1/pentests

# Fetch details for one pentest
GET /v1/pentests/abc-123

# Fetch weaknesses for that pentest
GET /v1/pentests/abc-123/weaknesses

# Fetch affected host for a weakness
GET /v1/pentests/abc-123/weaknesses/weakness-1/affected_host
```

GraphQL (Single Request)

```
query {
  pentest(op_id: "abc-123") {
    name
```

```
    state
    weaknesses_page {
      weaknesses {
        vuln {
          id
          name
          description
        }
        affected_host {
          ip
          host_names
          os_names
        }
        score
        severity
      }
    }
  }
}
```

8.3.2 Authentication

All API requests require a JSON Web Token (JWT). To obtain one, send your API key to the `/auth` endpoint as shown in the example on the right. The response contains a `token` field that you will include in subsequent requests.

API KEY REQUIRED

You can generate an API key from the Horizon3.ai Portal.

- US Portal
- EU Portal

AUTH ENDPOINTS

Use the endpoint for your region:

- US: `H3_AUTH_URL=https://api.gateway.horizon3ai.com/v1/auth`
- EU: `H3_AUTH_URL=https://api.gateway.horizon3ai.eu/v1/auth`

JWT EXPIRATION

Tokens expire after 1 hour. When a token expires, the API returns the following message. Simply re-authenticate to the `/auth` endpoint to obtain a new token.

```
{"message":"The incoming token has expired"}
```

Authenticate

```
H3_AUTH_URL="https://api.gateway.horizon3ai.com/v1/auth"
H3_API_KEY="<your-api-key-here>"
curl -X POST $H3_AUTH_URL \
    -H "Content-Type: application/json" \
    -d @- <<HERE
{
  "key": "$H3_API_KEY"
}
HERE
```

Response

```
{"token": "<your-jwt>"}
```

8.3.3 Sending Requests

Send an HTTP POST request to the `/graphql` endpoint with your JWT in the `Authorization` header. The example on the right retrieves all pentests for your account, ordered by most recent.

API ENDPOINTS

Use the endpoint for your region:

- US: `H3_API_URL=https://api.gateway.horizon3ai.com/v1/graphql`
- EU: `H3_API_URL=https://api.gateway.horizon3ai.eu/v1/graphql`

REQUEST BODY

The POST body is a JSON object with two fields:

- `query` — The GraphQL query or mutation string.
- `variables` — An optional JSON object containing variables referenced in the query or mutation.

For additional examples, see the Getting Started guide.

GraphQL Request

```
H3_API_URL="https://api.gateway.horizon3ai.com/v1/graphql"
H3_API_JWT="<your-jwt>"
curl -X POST $H3_API_URL \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer $H3_API_JWT" \
    -d @- <<HERE
{
  "query": "
    query q(\$page_input: PageInput) {
      pentests_count(page_input: \$page_input)
      pentests_page(page_input: \$page_input) {
        pentests {
          ...PentestFragment
        }
      }
    }
    fragment PentestFragment on Pentest {
      op_id
      op_type
      name
      state
      scheduled_at
      launched_at
      completed_at
      etl_completed_at
      hosts_count
      weaknesses_count
    }
  ",
  "variables": {
    "page_input": {
      "page_num": 1
    }
  }
}
HERE
```

8.3.4 Response Format

Every response is a JSON object with up to two top-level fields:

- `data` — The results of your query or mutation, mirroring the shape of the request.
- `errors` — Present only if one or more errors occurred during execution. See Handling Errors for details.

GraphQL Response


```
{
  "data": {
    "pentests_count": 8,
    "pentests_page": {
      "pentests": [
        {
          "op_id": "12341234-1234-1234-1234-123412341234",
          "op_type": "NodeZero",
          "name": "Sample Pentest",
          "state": "done",
          "scheduled_at": "2025-03-06T20:37:31.973429",
          "launched_at": "2025-03-06T20:44:22.567519",
          "completed_at": "2025-03-06T21:22:33.893341",
          "etl_completed_at": "2025-03-06T21:30:01.073779",
          "hosts_count": 10,
          "weaknesses_count": 5
```

```
        },
        ...
      ]
    }
  }
}
```

8.3.5 Handling Errors

Errors are returned in the `errors` array of the response. See the example on the right.

STATUS CODES

GraphQL requests return HTTP 200 in most cases, even when there are application-level errors. An HTTP 5xx status code indicates a network or system failure that prevented the API from responding.

Each error object may include an HTTP-style status code embedded in its `message` field, enclosed in square brackets (e.g., `[403] You are not authorized`).

The table below summarizes the status codes you may encounter.

Code	Title	Description
200	Success	The response contains the requested data.
400	Bad Request	The request is malformed. Check the `errors` field for details.
401	Unauthorized	Authentication failed due to an invalid or expired JWT.
403	Forbidden	You do not have permission to access the requested resource.
5xx	Internal Server Error	An unexpected server-side error occurred.

Example Error Response

```
{
  "data": {
    "pentests_count": null,
    "pentests_page": null
  },
  "errors": [
    {
      "message": "[403] You are not authorized",
      "locations": [ { "line": 2, "column": 13 } ],
      "path": [ "pentests_count" ]
    }
  ]
}
```

8.3.6 Versioning

The Horizon3.ai GraphQL API does not use version numbers. All changes are additive and backward-compatible, so existing integrations continue to work as the API evolves.

Deprecations: Queries, mutations, and fields may be deprecated over time. Deprecated items are flagged in this documentation and remain functional until they are no longer in active use.

8.3.7 h3-cli Examples

h3-cli is a command-line tool for interacting with the Horizon3.ai GraphQL API. You can use it to run any of the query and mutation examples in this reference.

INSTALLATION

Install h3-cli using the commands shown on the right.

USAGE

Run `h3 help` to see all available subcommands.

The `h3 gql` subcommand executes any GraphQL query or mutation. Save your query in a `.graphql` file and pass it to `h3 gql`:

```
h3 gql <graphql_file> [params_json]
```

PARAMETERS

- `graphql_file` — Path to a `.graphql` file containing your query or mutation.
- `params_json` — Optional JSON string with variables for the query or mutation.

The **h3-cli** tabs throughout this reference show the exact command for each query and mutation.

PRETTY-PRINTING WITH `jq`

h3-cli bundles `jq` for formatting JSON output. Pipe the result to `jq .` to pretty-print:

```
h3 gql ./my_pentests.graphql '{"page_input": {"page_num": 1}}' | jq .
```

AUTO-PAGINATION

For queries that accept a `$page_input: PageInput` argument, use `h3 gql-stream` to automatically iterate through all pages:

```
h3 gql-stream ./my_pentests.graphql data.pentests_page.pentests
```

The second argument (`data.pentests_page.pentests` in this example) is the path to the paginated array in the response. h3-cli fetches each page and streams the individual objects until all results have been returned.

JSON STREAMING

The output of `gql-stream` is a stream of JSON objects, not a JSON array. To collect the stream into an array, pipe the output to `jq -s`:

```
h3 gql-stream ./my_pentests.graphql data.pentests_page.pentests | jq -s .
```

Installation

```
H3_API_KEY="<your-api-key-here>"
H3_DOWNLOAD_URL="https://downloads.horizon3ai.com/utilities/cli/easy_install.sh"

curl $H3_DOWNLOAD_URL | bash -s "$H3_API_KEY"

export H3_CLI_HOME="`pwd`/h3-cli"
export PATH="$H3_CLI_HOME/bin:$PATH"

# Commands
h3 help
h3 whoami
h3 pentests
```

8.3.8 Python Examples

`h3_gql.py` is a ready-to-use Python script that handles authentication and query execution in a single call. The full source is shown on the right.

REQUIRED ENVIRONMENT VARIABLES

- `H3_API_KEY` — Your API key.
- `H3_AUTH_URL` — The authentication endpoint for your region. See Authentication.
- `H3_API_URL` — The GraphQL endpoint for your region. See Sending Requests.

USAGE

```
python h3_gql.py <graphql_file> [params_json]
```

PARAMETERS

- `graphql_file` — Path to a `.graphql` file containing your query or mutation.

- `params_json` — Optional JSON string with variables for the query or mutation.

The **Python** tabs throughout this reference show the exact command for each query and mutation.

h3_gql.py

```python
import json
import os
import sys
from pathlib import Path
import requests

def authenticate():
    url = os.environ["H3_AUTH_URL"]
    api_key = os.environ["H3_API_KEY"]
    response = requests.post(url, json={"key": api_key})
    response.raise_for_status()
    return response.json()["token"]

def execute_query(                            ):
    url = os.environ["H3_API_URL"]
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer      ",
    }
    query = Path(graphql_file).read_text()
    payload = {"query": query, "variables": variables}
    response = requests.post(url, headers=headers, json=payload)
    return response.json()

if __name__ == "__main__":
    graphql_file = sys.argv[1]
    params = json.loads(sys.argv[2]) if len(sys.argv) > 2 else {}
    jwt = authenticate()
    result = execute_query(jwt, graphql_file, params)
    print(json.dumps(result, indent=2))
```

8.3.9 cURL Examples

`h3_gql.sh` is a ready-to-use bash script that handles authentication and query execution using only cURL and `sed` — no additional dependencies required. The full source is shown on the right.

REQUIRED ENVIRONMENT VARIABLES

- `H3_API_KEY` — Your API key.
- `H3_AUTH_URL` — The authentication endpoint for your region. See Authentication.
- `H3_API_URL` — The GraphQL endpoint for your region. See Sending Requests.

USAGE

```
chmod +x h3_gql.sh
./h3_gql.sh <graphql_file> [params_json]
```

PARAMETERS

- `graphql_file` — Path to a `.graphql` file containing your query or mutation.
- `params_json` — Optional JSON string with variables for the query or mutation.

The **cURL** tabs throughout this reference show the exact command for each query and mutation.

h3_gql.sh

```bash
#!/usr/bin/env bash
set -euo pipefail

graphql_file="${1:?Usage: ./h3_gql.sh <graphql_file> [params_json]}"
variables="${2:-{}}"

# Authenticate and extract JWT using sed (no jq/python dependency)
jwt=$(curl -s -X POST "$H3_AUTH_URL" \
    -H "Content-Type: application/json" \
    -d "{\"key\": \"$H3_API_KEY\"}" \
    | sed -n 's/.*"token"[[:space:]]*:[[:space:]]*"\([^"]*\)".*/\1/p')

query=$(cat "$graphql_file" | sed 's/"/\\\"/g' | tr '\n' ' ')

curl -s -X POST "$H3_API_URL" \
    -H "Content-Type: application/json" \
```

```
    -H "Authorization: Bearer $jwt" \
    -d "{\"query\": \"$query\", \"variables\": $variables}"
```

8.4 Using the API

Core utilities for working with the Horizon3.ai GraphQL API. This section covers pagination controls for managing large datasets, filtering and sorting options, and metadata queries for API introspection and type definitions.

8.5 Using the API: Pagination

Control pagination, filtering, and sorting for query results that return large datasets. Use PageInput to navigate through paginated results, apply filters to narrow results, and specify sort order.

8.5.1 PageInput

Description

Controls pagination, sorting, and filtering for any paginated query.

Pass a PageInput to any `*_page` query to specify which page of results to return, how to order them, and which filters to apply. Pages are 1-indexed, with a default size of 20 and a maximum of 100 results per page.

Fields

`page_num` - `Int`
Page number to retrieve, starting at 1. Combined with `page_size` to determine the offset into the result set. Defaults to 1 if not specified.
`page_size` - `Int`
Number of results per page. Minimum is 1, maximum is 100. Defaults to 20 if not specified.
`order_by` - `String`
Name of the field to sort by. Must correspond to a field on the returned type. Not all fields are sortable; an error is returned for non-sortable fields. For multi-field sorting, use `sort_inputs` instead.
`sort_order` - `SortOrder`
Sort direction (`ASC` or `DESC`). Used together with `order_by` .
`sort_inputs` - `[SortInput]`
List of sort criteria for multi-field ordering. Each entry specifies a field and direction. See SortInput.
`filter_by_inputs` - `[FilterByInput]`
List of filter criteria. By default, multiple filters are combined with `AND` logic. Use `filter_by_input_mode` to switch to `OR` . See FilterByInput.
`filter_by_input_mode` - `FilterByInputMode`
Logical operator (`AND` or `OR`) for combining multiple `filter_by_inputs` entries. Defaults to `AND` . See FilterByInputMode.
`text_search` - `String`
Free-text search across text-based fields of the target type. Results include any record containing a match in at least one text field.

Request

```
# Example 1: Basic pagination
query BasicPagination($page_input: PageInput = {
    page_num: 1,
    page_size: 5
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      state
    }
  }
}

# Example 2: Pagination with sorting
query PaginationWithSorting($page_input: PageInput = {
    page_size: 10,
```

```
    sort_inputs: [
        { order_by: "scheduled_at", sort_order: DESC }
    ]
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      scheduled_at
    }
  }
}

# Example 3: Pagination with filtering
query PaginationWithFiltering($page_input: PageInput = {
    filter_by_inputs: [
        { field_name: "state", values: ["done"] }
    ]
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      op_type
      state
    }
  }
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.2 FacetValue

Description

Represents a single filterable value within a faceted navigation set. Each FacetValue provides a value, its count in the current result set, and a prebuilt FilterBy object that can be passed directly to PageInput `filter_by_inputs` to narrow results to this value.

Fields

`label` - `String`
Human-readable display name for this facet value.
`value` - `String`
The raw value used when filtering via the API.
`value_count` - `Int`
Number of results matching this facet value in the current dataset.
`filter_by` - `FilterBy`
A prebuilt FilterBy object that can be passed to PageInput `filter_by_inputs` to filter results by this facet value.

Request Response

```
fragment FacetValueFragment on FacetValue {
  label
  value
  value_count
  filter_by {
    field_name
    values
  }
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.3 FilterBy

Description

Represents an active filter applied to query results. This is the output counterpart of FilterByInput, returned within PageInfo to confirm which filters were applied.

Fields

`field_name` - `StringNoWhitespace!`
Name of the field being filtered.
`values` - `[String]`
Values included in the filter. See FilterByInput.
`not_values` - `[String]`
Values excluded by the filter. See FilterByInput.
`less_than` - `String`
Upper bound (exclusive) applied to the field. See FilterByInput.
`less_than_or_equal` - `String`
Upper bound (inclusive) applied to the field. See FilterByInput.
`greater_than` - `String`
Lower bound (exclusive) applied to the field. See FilterByInput.
`greater_than_or_equal` - `String`
Lower bound (inclusive) applied to the field. See FilterByInput.

Request Response

```
fragment FilterByFragment on FilterBy {
  field_name
  values
  not_values
  less_than
  less_than_or_equal
  greater_than
  greater_than_or_equal
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.4 FilterByInput

Description

Defines a single filter criterion for querying data. Specify a `field_name` and one or more conditions (`values`, `not_values`, range comparisons, or `is_null`) to restrict results. Multiple conditions within a single FilterByInput are combined with AND logic.

Fields

`field_name` - `StringNoWhitespace!`
Name of the field to filter by. Must correspond to a field name on the returned type.
`values` - `[String]`
Include only results where the field matches one of these values. Mutually exclusive with `not_values`. Can be combined with range comparisons (`greater_than*`, `less_than*`) and `is_null`; combined conditions are AND'ed.
`not_values` - `[String]`
Exclude results where the field matches one of these values. Mutually exclusive with `values`. Can be combined with range comparisons (`greater_than*`, `less_than*`) and `is_null`; combined conditions are AND'ed.
`greater_than` - `String`
Include only results where the field value is strictly greater than this value. Mutually exclusive with `greater_than_or_equal`. Can be combined with `less_than*`, `values`, `not_values`, and `is_null`; combined conditions are AND'ed.
`greater_than_or_equal` - `String`
Include only results where the field value is greater than or equal to this value. Mutually exclusive with `greater_than`. Can be combined with `less_than*`, `values`, `not_values`, and `is_null`; combined conditions are AND'ed.
`less_than` - `String`
Include only results where the field value is strictly less than this value. Mutually exclusive with `less_than_or_equal`. Can be combined with `greater_than*`, `values`, `not_values`, and `is_null`; combined conditions are AND'ed.
`less_than_or_equal` - `String`
Include only results where the field value is less than or equal to this value. Mutually exclusive with `less_than`. Can be combined with `greater_than*`, `values`, `not_values`, and `is_null`; combined conditions are AND'ed.
`filter_by_input_mode` - `FilterByInputMode`
Logical operator (`AND` or `OR`) for combining multiple FilterByInput entries. Defaults to `AND`.
`is_null` - `Boolean`
Filter by null status. Set to `true` to include only rows where the field IS NULL. Set to `false` to include only rows where the field IS NOT NULL. Omit or set to `null` to skip null checking. Can be combined with other conditions; combined conditions are AND'ed.

Request

```
# Example 1: Filter by single value
query FilterBySingleValue($page_input: PageInput = {
    filter_by_inputs: [
        { field_name: "op_type", values: ["NodeZero"] }
    ]
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      op_type
    }
  }
}

# Example 2: Filter by multiple values
query FilterByMultipleValues($page_input: PageInput = {
    filter_by_inputs: [
        { field_name: "state", values: ["done", "running"] }
    ]
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      op_type
      state
    }
```

```
  }
}

# Example 3: Exclude specific values
query FilterExcludeValues($page_input: PageInput = {
    filter_by_inputs: [
        { field_name: "state", not_values: ["ended", "error"] }
    ]
}) {
  pentests_page(page_input: $page_input) {
    pentests {
      op_id
      name
      state
    }
  }
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.5 FilterByInputMode

Description

Logical operator used to combine multiple FilterByInput entries when filtering query results.

Values

`AND`
All filters must match. This is the default when `filter_by_input_mode` is not specified.
`OR`
At least one filter must match.

Example

```
"AND"
```

Using the API: Pagination

8.5.6 PageInfo

Description

Echoes back the pagination parameters that were applied to produce the current page of results. Returned by all paginated queries alongside the result data.

Fields

`page_num` - `Int`
The page number that was returned.
`page_size` - `Int`
The maximum number of items per page that was requested.
`text_search` - `String`
The free-text search string that was applied, if any.
`filter_by_inputs` - `[FilterBy]`
The list of filters that were applied to the query. See FilterBy.

Request Response

```
fragment PageInfoFragment on PageInfo {
  page_num
  page_size
  text_search
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.7 SortInput

Description

Defines a single sort criterion. Use within PageInput `sort_inputs` to sort results by one or more fields.

Fields

`order_by` - `StringNoWhitespace!`

Name of the field to sort by. Must correspond to a field name on the returned type.

Not all fields are sortable. An error is returned if you specify a non-sortable field. The set of sortable fields depends on the type being queried.

`sort_order` - `SortOrder`
Sort direction. Defaults to `ASC` if not specified.
`nulls_first` - `Boolean`
Controls the ordering of null values in descending sorts. When `false` (default), nulls are placed last even in `DESC` order. Set to `true` to place nulls first in `DESC` queries.

Request

```
# Example 1: Sort by a single field
query SortBySingleField($page_input: PageInput = {
    sort_inputs: [
        { order_by: "severity", sort_order: DESC }
    ]
}) {
  weaknesses_page(
    input: { op_id: "1234abcd-1234-abcd-1234-abcd1234abcd" }
    page_input: $page_input
  ) {
    weaknesses {
      uuid
      vuln_name
      severity
    }
  }
}

# Example 2: Sort by multiple fields
query SortByMultipleFields($page_input: PageInput = {
    sort_inputs: [
        { order_by: "severity", sort_order: DESC },
        { order_by: "vuln_name", sort_order: ASC }
    ]
}) {
  weaknesses_page(
    input: { op_id: "1234abcd-1234-abcd-1234-abcd1234abcd" }
    page_input: $page_input
  ) {
    weaknesses {
      uuid
      vuln_name
      severity
    }
  }
}
```

Examples may not include all available fields for a type.

Using the API: Pagination

8.5.8 SortOrder

Description

Specifies the direction in which to order query results.

Values

`ASC`
Ascending order (smallest to largest, A to Z, oldest to newest).
`DESC`
Descending order (largest to smallest, Z to A, newest to oldest).

Example

```
"ASC"
```

8.6 Using the API: Metadata

Query GraphQL schema metadata and type definitions. Introspect the API to discover available types, fields, queries, and mutations programmatically, such as with an LLM and MCP server.

8.6.1 `graphql_def`

Description

Returns the GraphQLDef schema definition for a single element in the API. No authentication is required. The `id` parameter identifies the schema element and can be one of:

- a type name, eg. `Query`, `Mutation`, `Weakness`, etc.
- a field name, eg. `Query.pentests_page`, `Mutation.create_op`, etc.
- an enum name, eg. `AuthzRole`, `PortalOpState`, etc.
- an enum value, eg. `AuthzRole.ORG_ADMIN`, `PortalOpState.running`, etc.

Response

Returns an `GraphQLDef!`

Arguments

`id - String!`
The schema element ID, eg. `Query`, `Weakness`, `Mutation.create_op`, etc.

Request | Response | Python
cURL | h3-cli

```
query GetGraphQLDefinition($id: String!) {
  graphql_def(id: $id) {
    ...GraphQLDefFragment
  }
}

fragment GraphQLDefFragment on GraphQLDef {
  id
  name
  type
  type_name
  declaration
  description
  parent_id
  default_value
  related_type_ids
  args
  fields {
    id
    name
    type
```

```
    type_name
    description
  }
}
```

Examples may not include all available fields for a type.

Using the API: Metadata

8.6.2 `hello`

Description

A simple connectivity test that returns the string `world!`. Use this query to verify that your API client can successfully reach the GraphQL endpoint. No authentication is required.

```
query {
    hello
}
```

Response

Returns a `String!`

Request	Response	Python
cURL	h3-cli	

```
query HelloWorld {
  hello
}
```

Examples may not include all available fields for a type.

Using the API: Metadata

8.6.3 GraphQLDef

Description

Represents the schema definition of a single element in the GraphQL API. Use the graphql_def query to retrieve definitions programmatically. A GraphQLDef can represent one of:

- a Type (e.g. `Query`, `Mutation`, `WeaknessesPage`, `Weakness`)
- a Field within a type (e.g. `Query.weaknesses_page`, `Weakness.op_id`)
- an Enum (e.g. `AuthzRole`, `PortalOpState`)
- an Enum Value (e.g. `AuthzRole.ORG_ADMIN`, `PortalOpState.running`)

Fields

`id` - `String!`

Unique identifier for this schema element.

- For a Type: the type name (e.g. `Query`, `WeaknessesPage`)
- For a Field: the parent type + field name (e.g. `Query.weaknesses_page`)
- For an Enum: the enum name (e.g. `AuthzRole`)
- For an Enum Value: the enum name + value (e.g. `AuthzRole.ORG_ADMIN`)

`description` - `String`
The documentation string for this schema element, sourced from its GraphQL doc comment.
`declaration` - `String`

The full GraphQL declaration syntax for this schema element. For example:

`weaknesses_page(input: OpInput!, page_input: PageInput): WeaknessesPage!`

: WeaknessesPage!)

`name` - `String!`

The name of this schema element.

- For a Type: the type name (e.g. `Query`, `WeaknessesPage`)
- For a Field: the field name (e.g. `weaknesses_page`)
- For an Enum: the enum name (e.g. `AuthzRole`)
- For an Enum Value: the enum value (e.g. `ORG_ADMIN`)

`type` - `String!`
The return type of this element, including GraphQL modifiers such as `!` (non-null) and `[]` (list).
`type_name` - `String!`
The base type name without modifiers (`!`, `[]`).
`parent_id` - `String`
The parent type ID. Applies only to fields and enum values. For example, a field `weaknesses_page` on `Query` has a `parent_id` of `Query`.
`default_value` - `String`
The default value for an input field or argument, if one is defined.
`related_type_ids` - `[String!]`
IDs of all types related to this schema element. Applies to fields with arguments: includes the field's return type and the types of all arguments.
`related_types` - `[GraphQLDef!]`
The full GraphQLDef objects for all related types. Applies to fields with arguments: includes the return type and argument types.

`args` - `[JSONObject!]`

The list of arguments accepted by this field, if any. Each argument is a JSON object with keys:

```
[
  {
    "declaration": "page_input: PageInput",
    "defaultValue": null,
    "description": null,
    "name": "page_input",
    "type": "PageInput",
    "type_name": "PageInput"
  }
]
```

`fields` - `[GraphQLDef!]`

Child fields of this type. Only populated for Object and Input types.

Request Response

```
fragment GraphQLDefFragment on GraphQLDef {
  id
  name
  description
  declaration
  type
  type_name
  parent_id
  default_value
  related_type_ids
  args
  fields {
    id
    name
    type
    type_name
    description
  }
}
```

Examples may not include all available fields for a type.

8.7 Setup

Configure your pentesting environment and define your attack surface. Create and manage pentest templates, deploy NodeZero runners, schedule recurring pentests, organize assets into groups, and authorize domains and IPs for external testing. This section provides all the tools needed to prepare your pentesting infrastructure.

8.8 Setup: Op Config

Configure pentest parameters and settings including scope, attack types, feature flags, credentials, and runtime limits. This section defines the comprehensive configuration object used when creating or scheduling pentests and other operations.

8.8.1 ScheduleOpForm

Description

The configuration used to create an Op or an OpTemplate.

This type represents the read view of an op's configuration. To create or update a configuration, use ScheduleOpFormInput.

Key configuration fields include:

- `op_type`: The type of operation (e.g. internal pentest, external pentest, AD password audit). See OpType for the full list and configuration requirements.
- `op_param_max_scope` / `op_param_min_scope`: Define the network scope for the op.
- `feature_flags`: Advanced configuration settings that control specific attack behaviors. See FeatureFlag.
- `asset_group_uuid`: The AssetGroup to target for external pentests.
- `runner_uuid`: The NodeZero Runner that launches the op.
- `k8s_deployment` and related `k8s_*` fields: Configuration for Kubernetes-based pentests.
- `auto_injected_credential_uuids`: Credentials to be auto-injected by a NodeZero Runner.
- `minimum_run_time` / `maximum_run_time`: Control how long the op runs.

Note: Despite its name, `ScheduleOpForm` is not used for creating a recurring schedule. Use `Schedule` and `ScheduledAction` for that purpose.

Fields

`op_name` - `String`

The name of the operation.

The name can be anything. It is used in reports and the UI to label the operation.

`op_type` - `OpType`

The type of operation.

See ScheduleOpFormInput.op_type for more details.

`op_param_blacklist` - `String`

Hosts and subnets that are EXCLUDED from the op.

See ScheduleOpFormInput.op_param_blacklist for more details.

`op_param_min_scope` - `String`

The minimum network scope for the op, with auto-expand enabled.

See ScheduleOpFormInput.op_param_min_scope for more details.

`op_param_max_scope` - `String`

The maximum network scope for the op, with auto-expand disabled.

See ScheduleOpFormInput.op_param_max_scope for more details.

`feature_flags` - `[FeatureFlag]`

Advanced configuration settings that control specific attack behaviors during the op.

See ScheduleOpFormInput.feature_flags for more details.

`osint_domains` - `[String]`

Public-facing domains to scan for OSINT (open-source intelligence) data such as leaked credentials.

See ScheduleOpFormInput.osint_domains for more details.

`osint_company_names` - `[String]`

Company names used for discovering OSINT (open-source intelligence) data such as leaked credentials.

See ScheduleOpFormInput.osint_company_names for more details.

`passwords_to_spray` - `[String]`

Passwords to use for password spraying during the op.

See ScheduleOpFormInput.passwords_to_spray for more details.

`git_accounts` - `[GitAccount]`

Git accounts to scan during the op for exposed repositories and secrets.

See ScheduleOpFormInput.git_accounts for more details.

`aws_account_ids` - `[AWSAccountId]`

AWS account IDs to scan during the op for cloud-related attack paths.

See ScheduleOpFormInput.aws_account_ids for more details.

`asset_group_uuid` - `String`

ID of the AssetGroup containing authorized assets to pentest.

See ScheduleOpFormInput.asset_group_uuid for more details.

`start_paused` - `Boolean`

When `true`, the op starts in a paused state.

See ScheduleOpFormInput.start_paused for more details.

`minimum_run_time` - `Int`

The minimum number of minutes the op will run.

See ScheduleOpFormInput.minimum_run_time for more details.

`maximum_run_time` - `Int`

The maximum number of minutes the op will run.

See ScheduleOpFormInput.maximum_run_time for more details.

`runner_uuid` - `String`

ID of the NodeZero Runner that will launch this op.

See ScheduleOpFormInput.runner_uuid for more details.

`auto_injected_credential_uuids` - `[String!]`

UUIDs of credentials to be auto-injected by a NodeZero Runner when launching the op.

See ScheduleOpFormInput.auto_injected_credential_uuids for more details.

`targeted_test_id` - `String`

ID of the targeted test to use for the op.

See ScheduleOpFormInput.targeted_test_id for more details.

`k8s_deployment` - `Boolean`

Set to `true` when deploying NodeZero in a Kubernetes cluster.

See ScheduleOpFormInput.k8s_deployment for more details.

`k8s_node` - `String`

The Kubernetes node to target for the op.

See ScheduleOpFormInput.k8s_node for more details.

`k8s_namespace` - `String`

The Kubernetes namespace to target for the op.

See ScheduleOpFormInput.k8s_namespace for more details.

`k8s_service_account_name` - `String`

The Kubernetes service account to use for the op.

See ScheduleOpFormInput.k8s_service_account_name for more details.

`aws_connection_uuids` - `[String!]`

UUIDs of AWSConnection objects to use for the op.

See ScheduleOpFormInput.aws_connection_uuids for more details.

`aws_connection_identifier_uuid` - `String`

Global AWS connection identifier for the client account.

See ScheduleOpFormInput.aws_connection_identifier_uuid for more details.

`azure_tenant_id` - `String`

The Azure tenant where the NodeZero VM will be deployed and run the test within.

See ScheduleOpFormInput.azure_tenant_id for more details.

`azure_subscription_id` - `String`

The Azure subscription used by the NodeZero VM.

See ScheduleOpFormInput.azure_subscription_id for more details.

`azure_subnet_name` - `String`

The Azure subnet where the NodeZero VM will be deployed and run the test from.

See ScheduleOpFormInput.azure_subnet_name for more details.

`azure_role_name` - `String`

The Azure role that will be assigned to the NodeZero VM for the test.

See ScheduleOpFormInput.azure_role_name for more details.

`azure_custom_tags` - `JSONObject`

Optional Azure resource tags to apply to the NodeZero VM and supporting infrastructure provisioned for this op.

See ScheduleOpFormInput.azure_custom_tags for more details.

`inventory_op_scope_uuids` - `[ID!]`

UUIDs of InventoryOpScope objects that define the authorized scope for external operations.

See ScheduleOpFormInput.inventory_op_scope_uuids for more details.

`all_authorized_assets` - `Boolean`

When `true`, the op includes all authorized assets with no additional scope filtering.

See ScheduleOpFormInput.all_authorized_assets for more details.

Request Response

```
fragment ScheduleOpFormFragment on ScheduleOpForm {
  op_name
  op_type
  op_param_blacklist
  op_param_max_scope
  feature_flags {
    name
    value
  }
  osint_domains
  osint_company_names
  passwords_to_spray
  asset_group_uuid
  start_paused
  minimum_run_time
  maximum_run_time
  runner_uuid
  auto_injected_credential_uuids
  targeted_test_id
  k8s_deployment
  k8s_node
  k8s_namespace
  k8s_service_account_name
  aws_connection_uuids
  aws_connection_identifier_uuid
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.2 ScheduleOpFormInput

Description

Input configuration for creating an Op or an OpTemplate.

This is the input version of ScheduleOpForm, used when creating or updating op configurations.

See OpType for the list of op types and their configuration requirements.

Key configuration fields include:

- `op_type`: The type of operation (e.g. internal pentest, external pentest, AD password audit).
- `op_param_max_scope` / `op_param_min_scope`: Define the network scope for the op.
- `feature_flags`: Advanced configuration settings that control specific attack behaviors.
- `asset_group_uuid`: The AssetGroup to target for external pentests.
- `runner_uuid`: The NodeZero Runner that launches the op.
- `k8s_deployment` and related `k8s_*` fields: Configuration for Kubernetes-based pentests.
- `auto_injected_credential_uuids`: Credentials to be auto-injected by a NodeZero Runner.
- `minimum_run_time` / `maximum_run_time`: Control how long the op runs.

Partial/patch updates of an op template's configuration are supported via `Mutation.update_op_template`.

Fields

`op_name` - `String`

The name of the operation.

The name can be anything. It is used in reports and the UI to label the operation.

This also sets `AssetGroup.name` when used with `create_asset_group` and `update_asset_group_template`.

`op_type` - `OpType`

The type of operation.

Defaults to `NodeZero` (an internal pentest).

See OpType for the full list of supported op types and their configuration requirements.

`op_param_blacklist` - `String`

Hosts and subnets that are EXCLUDED from the op.

Any hosts or subnets that fall within this scope will not be scanned during the op.

Applies to internal op types that scan network ranges, such as `NodeZero`, `NetworkEnumeration`, `K8sPentest`, `InsiderThreatAttack`, `Phishing`, and `AzureInternalPentest`.

Specified as comma-separated CIDR ranges and/or individual IP addresses.

`op_param_min_scope` - `String`

The minimum network scope for the op, with auto-expand enabled.

Applies to internal op types that scan network ranges, such as `NodeZero`, `NetworkEnumeration`, `K8sPentest`, `InsiderThreatAttack`, `Phishing`, and `AzureInternalPentest`.

`op_param_min_scope` and `op_param_max_scope` are mutually exclusive ways to define the scope of an op. You can define one or the other, but not both at the same time.

If `op_param_min_scope` is defined, then the Auto-Expand Scope feature is enabled. This means NodeZero will scan not only the hosts defined in `op_param_min_scope`, but it will dynamically expand the scope during the operation to include any additional hosts it discovers organically.

Use `op_param_min_scope` if you're not sure of your full scope, and you want NodeZero to dynamically discover and scan additional hosts and scope, beyond the scope you defined in `op_param_min_scope`.

`op_param_min_scope` is typically configured as a set of individual host IPs, not large CIDR ranges or subnets. NodeZero will specifically search for and scan the defined IPs, in addition to any other hosts and subnets it discovers during the op.

Use `op_param_max_scope` if you know exactly the scope of hosts and subnets you want to scan, and you do NOT want NodeZero to scan any hosts or subnets beyond that scope. This is the more commonly used option, as it gives you the most control over which hosts and subnets are included in the scan.

If neither `op_param_min_scope` nor `op_param_max_scope` are defined, then NodeZero defaults to using Intelligent Scope. Intelligent Scope starts with the IP of the host that NodeZero is deployed on. From there, NodeZero enumerates and scans the /16 subnet that the host belongs to. NodeZero then expands into adjacent /23 subnets, continuously identifying and testing nearby assets. The cycle repeats until no additional devices are detected. The ultimate scope of the test depends on the level of privilege and access NodeZero has, either through captured or granted permissions.

The scope is specified as a list of comma-separated IPs and/or CIDR ranges.

`op_param_max_scope` - `String`

The maximum network scope for the op, with auto-expand disabled.

Applies to internal op types that scan network ranges, such as `NodeZero`, `NetworkEnumeration`, `K8sPentest`, `InsiderThreatAttack`, `Phishing`, and `AzureInternalPentest`.

`op_param_min_scope` and `op_param_max_scope` are mutually exclusive ways to define the scope of an op. You can define one or the other, but not both at the same time.

If `op_param_max_scope` is defined, then the Auto-Expand Scope feature is disabled. This means NodeZero will only scan the hosts and subnets defined in `op_param_max_scope`. NodeZero will NOT try to dynamically expand the scope during the operation to include any additional hosts it discovers organically. Only the hosts and subnets that fall within the scope defined in `op_param_max_scope` will be scanned.

Use `op_param_max_scope` if you know exactly the scope of hosts and subnets you want to scan, and you do NOT want NodeZero to scan any hosts or subnets beyond that scope. This is the more commonly used option, as it gives you the most control over which hosts and subnets are included in the scan.

Use `op_param_min_scope` if you're not sure of your full scope, and you want NodeZero to dynamically discover and scan additional hosts and scope.

If neither `op_param_min_scope` nor `op_param_max_scope` are defined, then NodeZero defaults to using Intelligent Scope. Intelligent Scope starts with the IP of the host that NodeZero is deployed on. From there, NodeZero enumerates and scans the /16 subnet that the host belongs to. NodeZero then expands into adjacent /23 subnets, continuously identifying and testing nearby assets. The cycle repeats until no additional devices are detected. The ultimate scope of the test depends on the level of privilege and access NodeZero has, either through captured or granted permissions.

The scope is specified as a list of comma-separated IPs and/or CIDR ranges.

`feature_flags` - `[FeatureFlagInput]`
Advanced configuration settings that control specific attack behaviors during the op. See FeatureFlagInput. Partial/patch updates are supported via `Mutation.update_op_template`. Only flags included in the input will be updated; unspecified flags retain their current values.
`osint_domains` - `[String]`
Public-facing domains to scan for OSINT (open-source intelligence) data such as leaked credentials.
`osint_company_names` - `[String]`

Company names used for discovering OSINT (open-source intelligence) data such as leaked credentials.

`passwords_to_spray` - `[String]`

Passwords to use for password spraying during the op.

`git_accounts` - `[GitAccountInput]`

Git accounts to scan during the op for exposed repositories and secrets.

`aws_account_ids` - `[AWSAccountId]`

AWS account IDs to scan during the op for cloud-related attack paths.

`asset_group_uuid` - `String`

ID of the AssetGroup containing authorized assets to pentest. Applies to external pentests only (`op_type=ExternalAttack`).

`start_paused` - `Boolean`

When `true`, the op starts in a paused state. This is useful for external pentests when NodeZero's IP address must first be known in order to configure firewall rules.

`minimum_run_time` - `Int`

The minimum number of minutes the op will run, even if it has fully completed its scan of the target environment. This is useful for giving extra time to password spraying and man-in-the-middle relay attacks.

`maximum_run_time` - `Int`

The maximum number of minutes the op will run. The op will stop after this duration even if it has not yet completed its scan of the target environment.

`runner_uuid` - `String`

ID of the NodeZero Runner that will launch this op.

Runners can be used for launching internal op types (e.g. `NodeZero`, `K8sPentest`).

External op types such as `ExternalAttack` and `AWSPentest` are launched from the Horizon3.ai cloud and do not require a runner.

`auto_injected_credential_uuids` - `[String!]`

UUIDs of credentials to be auto-injected by a NodeZero Runner when launching the op.

`targeted_test_id` - `String`

ID of the targeted test to use for the op. Targeted tests focus the op on specific weaknesses or assets.

`k8s_deployment` - `Boolean`

Set to `true` when deploying NodeZero in a Kubernetes cluster. Required for the `K8sPentest` op type.

`k8s_node` - `String`

The Kubernetes node to target for the op. Used by the `K8sPentest` op type.

`k8s_namespace` - `String`

The Kubernetes namespace to target for the op. Used by the `K8sPentest` op type.

`k8s_service_account_name` - `String`

The Kubernetes service account to use for the op. Used by the `K8sPentest` op type.

`aws_connection_uuids` - `[String!]`

UUIDs of AWSConnection objects to use for the op. Required for the `AWSPentest` op type.

`aws_connection_identifier_uuid` - `String`

Global AWS connection identifier for the client account. Used by the `AWSPentest` op type.

`enable_tripwires` - `Boolean`

Enables dropping tripwires during the op, provided the client account has the tripwires feature enabled. The specific types of tripwires to drop must also be configured via the `feature_flags` field.

`azure_tenant_id` - `String`

The Azure tenant where the NodeZero VM will be deployed and run the test within. Used by Azure Internal Pentest.

`azure_subscription_id` - `String`

The Azure subscription used by the NodeZero VM. Used by Azure Internal Pentest.

`azure_subnet_name` - `String`

The Azure subnet where the NodeZero VM will be deployed and run the test from. Used by Azure Internal Pentest.

`azure_role_name` - `String`

The Azure role that will be assigned to the NodeZero VM for the test. Used by Azure Internal Pentest.

`azure_custom_tags` - `JSONObject`

Optional Azure resource tags to apply to the NodeZero VM and supporting infrastructure provisioned for this op. Provided as a key/value object, e.g. `{"Environment": "Testing", "Owner": "SecOps"}`. Allows clients with Azure tag-based policies to authorize the test infrastructure. Keys/values must satisfy Azure tag constraints (≤50 entries, key ≤512 chars, value ≤256 chars, no reserved prefixes `microsoft`, `azure`, `windows`). Used by Azure Internal Pentest.

`inventory_op_scope_uuids` - `[ID!]`

UUIDs of InventoryOpScope objects that define the authorized scope for external operations. Applies to `ExternalAssetDiscovery` and `ExternalAttack` op types. Mutually exclusive with `asset_group_uuid`.

`all_authorized_assets` - `Boolean`

When `true`, the op includes all authorized assets with no additional scope filtering. Currently only supported for the `ExternalAttack` op type. Mutually exclusive with `asset_group_uuid`, `op_series_uuid`, and `inventory_op_scope_uuids`.

Request

```
# Example 1: Create an internal pentest with basic configuration
mutation CreateInternalPentest($schedule_op_form: ScheduleOpFormInput = {
    op_name: "Weekly Internal Pentest",
    op_type: NodeZero,
    op_param_max_scope: "10.0.0.0/8",
    maximum_run_time: 480
}) {
  create_op(schedule_op_form: $schedule_op_form) {
    op {
      op_id
      op_type
      op_name
      op_state
      op_param_max_scope
      nodezero_script_url
    }
  }
}

# Example 2: Create an external pentest
mutation CreateExternalPentest($schedule_op_form: ScheduleOpFormInput = {
    op_name: "External Attack Assessment",
    op_type: ExternalAttack,
    asset_group_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
}) {
  create_op(schedule_op_form: $schedule_op_form) {
    op {
      op_id
      op_type
      op_name
      op_state
      asset_group_uuid
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.3 AWSAccountId

Description

`String` scalar type representing an AWS Account ID (a 12-digit number).

Example

```
123456789012
```

Setup: Op Config

8.8.4 AttackConfigValue

Description

The user-provided value for a FeatureFlag. Exactly one field will be set, corresponding to the option's `data_type`.

Fields

`bool_value` - `Boolean`
Value for boolean-type options.
`int_value` - `Int`
Value for integer-type options.
`float_value` - `Float`
Value for floating-point-type options.
`string_value` - `String`
Value for string-type options.

Request Response

```
fragment AttackConfigValueFragment on AttackConfigValue {
  bool_value
  int_value
  float_value
  string_value
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.5 AttackConfigValueInput

Description

Input type for specifying the value of a FeatureFlag. Provide exactly one of the value fields, matching the option's `data_type`.

Fields

`bool_value` - `Boolean`
Value for boolean-type options.
`int_value` - `Int`
Value for integer-type options.
`float_value` - `Float`
Value for floating-point-type options.
`string_value` - `String`
Value for string-type options.

Request

```
# Example: Set a boolean config value via op template update
mutation SetAttackConfig($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    schedule_op_form: {
        feature_flags: [
            {
                name: "get_proof.bluekeep",
                config_value: { bool_value: true }
            }
        ]
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      schedule_op_form {
        feature_flags {
          name
          config_value {
            bool_value
            int_value
            string_value
          }
        }
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.6 FeatureFlag

Description

An advanced configuration option (also known as a feature flag) that controls specific attack behaviors during an op.

Feature flags are configured via `ScheduleOpFormInput.feature_flags` and can be read from `Op.feature_flags` or `ScheduleOpForm.feature_flags`.

For more information about advanced configuration options, see the Attack Configurations section of the Horizon3.ai documentation.

Fields

`name` - `String!`
The identifier of the advanced configuration option.
`value` - `Boolean`
Indicates whether the option is enabled or disabled.
`config_value` - `AttackConfigValue`
The user-provided value for this option. See AttackConfigValue for details.
`property_name` - `String`
Human-friendly display name for this option.)
`property_description` - `String`
A description of what this option does and when to use it.
`category_name` - `String`
The category this option belongs to (e.g. Exploitation, Reconnaissance).
`category_description` - `String`
A description of the category.
`risk` - `FeatureFlagRiskType`
The disruption risk level associated with enabling this option. See FeatureFlagRiskType.
`enables_min_runtime` - `Boolean`
When `true`, indicates this option works best when `ScheduleOpFormInput.minimum_run_time` is also configured, to give the attack enough time to execute.
`enables_password_spray` - `Boolean`
When `true`, indicates this option works best when `ScheduleOpFormInput.passwords_to_spray` is also configured with a list of passwords.
`is_new` - `Boolean!`
When `true`, indicates this option was introduced after the containing OpTemplate was last saved. This helps alert users to new configuration options available in their template. The flag resets once the template is re-saved.

Request Response

```
fragment FeatureFlagFragment on FeatureFlag {
  name
  value
  config_value {
    bool_value
    int_value
    float_value
    string_value
  }
  property_name
  property_description
  category_name
  category_description
  risk
  enables_min_runtime
  enables_password_spray
  is_new
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.7 FeatureFlagInput

Description

Input for configuring an advanced attack configuration option.

Advanced configuration options (also known as feature flags) control specific attack behaviors during an op. Specify these via `ScheduleOpFormInput.feature_flags`.

For more information about advanced configuration options, see the Attack Configurations section of the Horizon3.ai documentation.

Fields

`name` - `String!`
The identifier of the advanced configuration option.
`value` - `Boolean`
Indicates whether the option is enabled or disabled. Used for boolean-type options only.
`config_value` - `AttackConfigValueInput`
The configuration value for this option. The value type should match the option's `data_type`. See AttackConfigValueInput for details.

Request

```
# Example 1: Enable a boolean feature flag
mutation EnableFeatureFlag($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    schedule_op_form: {
        feature_flags: [
            {
                name: "get_proof.bluekeep",
                value: true
            }
        ]
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      schedule_op_form {
        feature_flags {
          name
          value
        }
      }
    }
  }
}

# Example 2: Set a feature flag with a config value
mutation SetFeatureFlagConfig($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    schedule_op_form: {
        feature_flags: [
            {
                name: "get_proof.bluekeep",
                config_value: { bool_value: true }
            }
        ]
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      schedule_op_form {
        feature_flags {
          name
          config_value {
            bool_value
          }
        }
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.8 FeatureFlagRiskType

Description

The disruption risk level associated with an advanced configuration option (FeatureFlag).

Higher-risk options may cause service interruptions or other observable effects in your environment during the op.

Values

`none`
No risk of disruption to your environment.
`low`
Low risk of disruption to your environment.
`moderate`
Moderate risk of disruption to your environment.
`high`
High risk of disruption to your environment. Use with caution.

Example

```
"none"
```

Setup: Op Config

8.8.9 GitAccount

Description

A Git account discovered or configured during an op.

Git accounts can be specified in the op configuration (via `ScheduleOpFormInput.git_accounts`) or discovered automatically by NodeZero during the op.

Fields

`name` - `String!`
Name of the Git account (user or organization name on the hosting service).
`source` - `GitAccountSource!`
The git hosting service for this account. See GitAccountSource.

Request Response

```
fragment GitAccountFragment on GitAccount {
  name
  source
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.10 GitAccountInput

Description

Input type for specifying a Git account to scan during an op.

Configure Git accounts via `ScheduleOpFormInput.git_accounts` to have NodeZero scan the specified repositories for exposed secrets and sensitive data.

Fields

`name` - `StringNoWhitespace!`
Name of the Git account (user or organization name on the hosting service).
`source` - `GitAccountSource!`
The git hosting service for this account. See GitAccountSource.

Request

```
# Example: Add git accounts to an op template
mutation SetGitAccounts($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    schedule_op_form: {
        git_accounts: [
            { name: "my-org", source: GitHub },
            { name: "my-team", source: GitLab }
        ]
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      schedule_op_form {
        git_accounts {
          name
          source
        }
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Config

8.8.11 GitAccountSource

Description

The git hosting service for a GitAccount.

Values

`GitLab`
GitLab.
`GitHub`
GitHub.
`Bitbucket`
Bitbucket.

Example

```
"GitLab"
```

Setup: Op Config

8.8.12 OpInput

Description

An input type for identifying an Op by its unique identifier.

Fields

`op_id` - `String!`
The unique identifier of the op.

Request

```
# Example: Fetch EDR hosts CSV
query EDRHostsCSV($input: OpInput!) {
  edr_overview_hosts_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

8.9 Setup: Op Templates

Create and manage reusable pentest templates that define standard configurations for different testing scenarios. Templates can be linked to schedules for automated recurring pentests and shared across your organization.

8.9.1 `op_templates_page`

Description

Retrieve a paginated list of op templates for the current client account. Does not include default templates provided by Horizon3.ai.

Use the `OpTemplate.uuid` or `OpTemplate.op_template_name` fields from the results to fetch complete details about a specific OpTemplate via `Query.op_template`.

Response

Returns an `OpTemplatesPage!`

Arguments

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query GetOpTemplatesPage($page_input: PageInput) {
  op_templates_page(page_input: $page_input) {
    op_templates {
      uuid
      op_template_name
      op_type
      row_created_at
      schedule_op_form {
        op_name
        op_param_max_scope
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.2 `op_templates_count`

Description

Get the total count of OpTemplate records for the current client account. Does not include default templates provided by Horizon3.ai.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query GetOpTemplatesCount($page_input: PageInput) {
  op_templates_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.3 `op_template`

Description

Retrieve a single op template by its unique identifier or name. Exactly one parameter must be provided. Returns the matching OpTemplate or null if not found.

Response

Returns an `OpTemplate`

Arguments

`op_template_uuid` - `String`
Unique identifier of the op template to retrieve. Provide either this or `op_template_name`, not both.
`op_template_name` - `String`
Display name of the op template to retrieve. Provide either this or `op_template_uuid`, not both.

Request Response Python
cURL h3-cli

```
query GetOpTemplate($op_template_uuid: String) {
  op_template(op_template_uuid: $op_template_uuid) {
    ...OpTemplateFragment
  }
}

fragment OpTemplateFragment on OpTemplate {
  uuid
  op_template_name
  name
  op_type
  user_account_uuid
  client_account_uuid
  row_created_at
  row_updated_at
  schedule_op_form {
    op_name
    op_type
    op_param_max_scope
    maximum_run_time
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.4 `create_op_template`

Description

Create an `OpTemplate`. An `OpTemplate` is a reusable template for running pentests and other op types.

Response

Returns a `SaveOpTemplateOutput!`

Arguments

`op_template_name` - `String!`
Display name for the template. Must be unique within the client account.
`schedule_op_form` - `ScheduleOpFormInput!`
Configuration data for the template. See ScheduleOpFormInput for available options.
`campaign_uuid` - `String`
Optional campaign UUID. When provided, the automatically created perspective will be added to this campaign.

Request	Response	Python
cURL	h3-cli	

```
mutation CreateOpTemplate(
  $op_template_name: String!
  $schedule_op_form: ScheduleOpFormInput!
) {
  create_op_template(
    op_template_name: $op_template_name
    schedule_op_form: $schedule_op_form
  ) {
    op_template {
      uuid
      op_template_name
      op_type
      row_created_at
      schedule_op_form {
        op_name
        op_type
        op_param_max_scope
        maximum_run_time
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.5 `update_op_template`

Description

Update an `OpTemplate`.

- Allows for template renaming.
- Allows for partial/patch updates of `OpTemplate.schedule_op_form`.

Response

Returns an `UpdateOpTemplateOutput!`

Arguments

`input` - `UpdateOpTemplateInput!`
Input data for updating the template.

Request | Response | Python
cURL | h3-cli

```
mutation UpdateOpTemplate($input: UpdateOpTemplateInput!) {
  update_op_template(input: $input) {
    op_template {
      uuid
      op_template_name
      op_type
      row_updated_at
      schedule_op_form {
        op_name
        op_type
        op_param_max_scope
        maximum_run_time
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.6 `update_op_templates`

Description

Update a batch of `OpTemplates`.

- Allows for template renaming.
- Allows for partial/patch updates of `OpTemplate.schedule_op_form`.

Response

Returns `[UpdateOpTemplateOutput!]!`

Arguments

`input` - `[UpdateOpTemplateInput!]!`
Input data for updating pentest template.

Request | Response | Python
cURL | h3-cli

```
mutation UpdateOpTemplates($input: [UpdateOpTemplateInput!]!) {
  update_op_templates(input: $input) {
    op_template {
      uuid
      name
      op_type
      schedule_op_form {
        op_name
        op_type
        op_param_max_scope
      }
      row_updated_at
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.7 `delete_op_template`

Description

Delete an op template by its display name.

Response

Returns a `DeleteOpTemplateOutput!`

Arguments

`op_template_name` - `String!`
Display name of the template to delete.
`op_type` - `String`
Type of operation. See OpType for available options.
`also_delete_schedules` - `Boolean`
When true, also deletes any schedules that reference this template.

Request | Response | Python
cURL | h3-cli

```
mutation DeleteOpTemplate($op_template_name: String!) {
  delete_op_template(op_template_name: $op_template_name) {
    op_template {
      uuid
      op_template_name
      op_type
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.8 `delete_op_template_by_uuid`

Description

Delete an op template by its unique identifier.

Response

Returns a `DeleteOpTemplateOutput!`

Arguments

`uuid` - `String!`
Unique identifier of the template to delete.
`also_delete_schedules` - `Boolean`
When true, also deletes any schedules that reference this template.

Request	Response	Python
cURL	h3-cli	

```
mutation DeleteOpTemplateByUuid($uuid: String!, $also_delete_schedules: Boolean) {
  delete_op_template_by_uuid(uuid: $uuid, also_delete_schedules: $also_delete_schedules) {
    op_template {
      uuid
      name
      op_type
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.9 `delete_op_templates_by_uuids`

Description

Delete multiple op templates by their unique identifiers.

Response

Returns `[DeleteOpTemplateOutput!]!`

Arguments

`uuids` - `[String!]!`
List of unique identifiers for the templates to delete.
`also_delete_schedules` - `Boolean`
When true, also deletes any schedules that reference these templates.

Request | Response | Python
cURL | h3-cli

```
mutation DeleteOpTemplatesByUuids($uuids: [String!]!, $also_delete_schedules: Boolean) {
  delete_op_templates_by_uuids(uuids: $uuids, also_delete_schedules: $also_delete_schedules) {
    op_template {
      uuid
      name
      op_type
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.10 `op_templates`

Description

List of op templates for the current client account. Includes default templates provided by Horizon3.ai.

Response

Returns `[OpTemplate]`

Arguments

`op_type` - `String`
Filter by a single op type. Defaults to "NodeZero" when neither `op_type` nor `op_types` is provided. Ignored when `op_types` is set.
`op_types` - `[String!]`
Filter by multiple op types. Returns templates matching any of the given op types (defaults + client). When provided, `op_type` is ignored.

Request | Response | Python
cURL | h3-cli

```
query OpTemplates($op_type: String) {
  op_templates(op_type: $op_type) {
    uuid
    name
    op_type
    schedule_op_form {
      op_name
      op_type
      op_param_blacklist
      op_param_min_scope
      op_param_max_scope
      runner_uuid
    }
    row_created_at
    row_updated_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.11 `op_templates_facets`

Description

Retrieve available filter values and counts for op templates. Returns distinct values that can be used to filter the op templates list, such as runner names.

Response

Returns an `OpTemplatesFacets!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query OpTemplatesFacets($page_input: PageInput) {
  op_templates_facets(page_input: $page_input) {
    page_info {
      page_size
    }
    runner_name {
      label
      value
      value_count
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.12 OpTemplate

Description

A reusable, named configuration for running pentests and other op types.

Each `OpTemplate` stores a complete set of configuration options in its `schedule_op_form` field — including the `op_type`, network scope, attack configurations, credentials, and any other settings required for the chosen op type.

An `OpTemplate` can be linked to a `Schedule` so that ops are created and run automatically on a recurring basis.

Common operations:

- **Create** a template: `Mutation.create_op_template`
- **Update** a template's configuration: `Mutation.update_op_template` (supports partial updates)
- **Delete** a template: `Mutation.delete_op_template`
- **Run** an op from a template: `Mutation.create_op` with the `op_template_uuid` parameter

Visit Pentest Templates in the Horizon3.ai documentation for more details on using templates to streamline pentest configuration and execution.

Fields

`uuid` - `String!`
Unique identifier of the op template.
`user_account_uuid` - `String!`
Unique identifier of the user account that created this template.
`client_account_uuid` - `String!`
Unique identifier of the client account that owns this template.
`client_account` - `ClientAccount!`
The ClientAccount that owns this template.
`op_template_name` - `String!`
Display name of the template. Must be unique within the client account.
`name` - `String!`
Display name of the template. Must be unique within the client account.
`op_type` - `OpType!`
Type of op this template applies to. See `OpType` for details.
`schedule_op_form` - `ScheduleOpForm!`
The template's configuration data. See ScheduleOpForm for details on available configuration options.
`runner_name` - `String`
Name of the NodeZero Runner referenced by `schedule_op_form.runner_uuid`. Exposed as a top-level scalar so list views can render the runner without resolving the full ScheduleOpForm.
`minimum_run_time` - `Int`
Minimum runtime in minutes pulled from `schedule_op_form.minimum_run_time`. Exposed as a top-level scalar for list views.
`maximum_run_time` - `Int`
Maximum runtime in minutes pulled from `schedule_op_form.maximum_run_time`. Exposed as a top-level scalar for list views.
`schedules` - `[Schedule!]!`
The Schedules that use this template for recurring pentest execution.
`row_created_at` - `Datetime!`
Timestamp when this template was created.
`row_updated_at` - `Datetime`
Timestamp when this template was last modified.

Request Response

```
fragment OpTemplateFragment on OpTemplate {
  uuid
  op_template_name
  name
  op_type
  user_account_uuid
  client_account_uuid
  row_created_at
  row_updated_at
  schedule_op_form {
    op_name
    op_type
    op_param_max_scope
    op_param_blacklist
    maximum_run_time
    minimum_run_time
  }
  schedules {
    uuid
    name
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.13 DeleteOpTemplateOutput

Description

Output returned when deleting an OpTemplate.

Fields

`op_template` - `OpTemplate`
The deleted op template. Null if the template was not found.

Request Response

```
fragment DeleteOpTemplateOutputFragment on DeleteOpTemplateOutput {
  op_template {
    uuid
    op_template_name
    op_type
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.14 OpTemplatesFacets

Description

Facets for op templates filtering. Provides available filter values with counts for the current dataset.

Fields

`page_info` - `PageInfo`
Pagination information
`runner_name` - `[FacetValue!]!`
List of unique runner names from `OpTemplate.schedule_op_form.runner_name` values present in the data after applying any filters.

Request Response

```
fragment OpTemplatesFacetsFragment on OpTemplatesFacets {
  page_info {
    page_size
  }
  runner_name {
    label
    value
    value_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.15 OpTemplatesPage

Description

Contains a paginated list of OpTemplate records.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`op_templates` - `[OpTemplate!]!`
List of op templates in the current page.

Request Response

```
fragment OpTemplatesPageFragment on OpTemplatesPage {
  page_info {
    page_size
    page_num
  }
  op_templates {
    uuid
    op_template_name
    op_type
    row_created_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.16 SaveOpTemplateOutput

Description

Output returned when creating or saving an OpTemplate.

Fields

`op_template` - `OpTemplate!`
The newly created or saved op template.
`asset_group` - `AssetGroup`
The op template's associated asset group, if applicable.

Request Response

```
fragment SaveOpTemplateOutputFragment on SaveOpTemplateOutput {
  op_template {
    uuid
    op_template_name
    op_type
    row_created_at
    schedule_op_form {
      op_name
      op_param_max_scope
    }
  }
  asset_group {
    uuid
    name
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.17 UpdateOpTemplateInput

Description

Input type for `Mutation.update_op_template`.

Fields

`uuid` - `String!`
The unique identifier of the op template to update.
`op_template_name` - `String`
New display name for the template. Must be unique within the client account.
`schedule_op_form` - `ScheduleOpFormInput`
Configuration updates for the template. Supports partial updates - only fields provided in this input will be modified. Existing configuration values are preserved for fields not included in the update.

Request

```
# Example 1: Rename an op template
mutation RenameOpTemplate($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    op_template_name: "New Template Name"
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      op_template_name
    }
  }
}

# Example 2: Update op template scope (partial update)
mutation UpdateOpTemplateScope($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    schedule_op_form: {
        op_param_max_scope: "192.168.0.0/16"
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      op_template_name
      schedule_op_form {
        op_param_max_scope
      }
    }
  }
}

# Example 3: Update multiple fields
mutation UpdateOpTemplateMultiple($input: UpdateOpTemplateInput! = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
    op_template_name: "Updated Weekly Pentest",
    schedule_op_form: {
        op_param_max_scope: "10.0.0.0/8",
        maximum_run_time: 600
    }
}) {
  update_op_template(input: $input) {
    op_template {
      uuid
      op_template_name
      schedule_op_form {
        op_param_max_scope
        maximum_run_time
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Op Templates

8.9.18 UpdateOpTemplateOutput

Description

Output returned when updating an OpTemplate.

Fields

`op_template` - `OpTemplate!`
The op template with updated configuration.

Request Response

```
fragment UpdateOpTemplateOutputFragment on UpdateOpTemplateOutput {
  op_template {
    uuid
    op_template_name
    op_type
    row_updated_at
    schedule_op_form {
      op_name
      op_param_max_scope
      maximum_run_time
    }
  }
}
```

Examples may not include all available fields for a type.

8.10 Setup: Runners

Deploy and manage NodeZero runners (agents and Kubernetes operators) that execute pentests from within your network perimeter. Runners simplify the process of launching internal pentests on a Docker host within your network. Runners are required for running internal pentests on an automated recurring basis using a Schedule.

8.10.1 `agents_page`

Description

Returns a paginated list of NodeZero Runners (Agents) belonging to the current user's ClientAccount.

Use PageInput to control pagination, sorting, and filtering. Set `include_sub_client_agents` to `true` to also include Runners from sub-client accounts (requires multi-tenancy to be enabled).

Response

Returns an `AgentsPage!`

Arguments

`page_input` - `PageInput`
`include_sub_client_agents` - `Boolean`

Request Response Python
cURL h3-cli

```
query AgentsPage($page_input: PageInput) {
  agents_page(page_input: $page_input) {
    agents {
      uuid
      name
      user_account_name
      api_key_uuid
      created_at
      is_kubernetes
      kubernetes_namespace
      h3_cli_version
      download_url
      last_heartbeat_at
      last_heartbeat_time_ago
      active_op_id
    }
```

```
    page_info {
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.2 `agents_count`

Description

Returns the total count of NodeZero Runners (Agents) belonging to the current user's ClientAccount.

Supports the same filtering criteria as agents_page via PageInput. Set `include_sub_client_agents` to `true` to include Runners from sub-client accounts.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

`include_sub_client_agents` - `Boolean`

Request | Response | Python

cURL | h3-cli

```
query AgentsCount($page_input: PageInput, $include_sub_clients: Boolean) {
  agents_count(
    page_input: $page_input
    include_sub_client_agents: $include_sub_clients
  )
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.3 `agent`

Description

Retrieves a single NodeZero Runner (Agent) by its `uuid` or `name` within the current user's ClientAccount.

You must provide either `uuid` or `name`. Returns `null` if no matching Runner is found.

Response

Returns an `Agent`

Arguments

`uuid` - `String`
The unique identifier of the NodeZero Runner to retrieve. Either `uuid` or `name` must be provided.
`name` - `String`
The name of the NodeZero Runner to retrieve. Either `uuid` or `name` must be provided.

Request Response Python
cURL h3-cli

```
query Agent($uuid: String, $name: String) {
  agent(uuid: $uuid, name: $name) {
    uuid
    name
    user_account_name
    api_key_uuid
    api_key {
      uuid
      starts_with
    }
    uname
    log_file
    last_heartbeat_at
    last_heartbeat_time_ago
    last_heartbeat_source_ip
    last_command {
      command
      received_at
      completed_at
      exit_status
    }
    commands {
      command
      received_at
      completed_at
      exit_status
    }
    created_at
    h3_cli_version
    download_url
    is_kubernetes
    kubernetes_namespace
    active_op_id
    op_next_scheduled_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.4 `install_agent`

Description

Creates a new NodeZero Runner and returns the installation command needed to deploy it.

NodeZero Runners are lightweight agents that run on your infrastructure and automatically launch NodeZero pentests whenever an Op is created and configured to use the Runner. You can also configure an OpTemplate to use a Runner for all ops created from that template.

This mutation creates a new API key with `NODEZERO_RUNNER` permissions (or reuses the existing `NODEZERO_OPERATOR` key for Kubernetes deployments). The API key is embedded in the installation command and stored on the Runner's host.

The InstallAgentOutput includes:

- The shell command to install and start the Runner (either `command` for Docker hosts or `kubernetes_install_command` for K8s).
- The Agent record representing the Runner, whose `uuid` can be used to assign it to an Op or OpTemplate even before installation completes.

Note: NodeZero Runners are referred to as Agents throughout the API.

Response

Returns an `InstallAgentOutput!`

Arguments

`agent_name` - `String`
DEPRECATED - use `name` instead.
`name` - `String`
The name of the NodeZero Runner. This is used to identify the Runner in the API. The name is required and must not be empty. The name must not contain spaces and must conform to RFC 1123 DNS subdomain naming conventions.
`is_kubernetes` - `Boolean`
Flag to indicate if the Runner is being deployed in a Kubernetes environment. Defaults to `false`.
`kubernetes_namespace` - `String`
The Kubernetes namespace to deploy the Runner in. Defaults to `nodezero` if omitted. Only applicable when `is_kubernetes` is `true`.

Request | Response | Python
cURL | h3-cli

```
mutation InstallAgent($name: String!, $is_kubernetes: Boolean, $kubernetes_namespace: String) {
  install_agent(
    name: $name
    is_kubernetes: $is_kubernetes
    kubernetes_namespace: $kubernetes_namespace
  ) {
    command
    kubernetes_install_command
    agent {
      uuid
      name
      created_at
      is_kubernetes
      kubernetes_namespace
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.5 `hello_agent`

Description

Sends a `hello-world` command to the Runner identified by the given AgentInput.name.

Use this mutation to test connectivity with a Runner and verify that it is active and able to receive and execute commands. The command is queued and delivered to the Runner during its next heartbeat. The resulting AgentCommand can be monitored to confirm the Runner successfully received and executed the ping.

Response

Returns an `AgentCommand`

Arguments

`input` - `AgentInput!`

Request | Response | Python

cURL | h3-cli

```
mutation HelloAgent($input: AgentInput!) {
  hello_agent(input: $input) {
    uuid
    agent_uuid
    command
    received_at
    completed_at
    exit_status
    log
    created_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.6 `upgrade_agent`

Description

Queues an upgrade command to the Runner identified by the given `agent_name` or `agent_uuid`. The Runner will upgrade to the latest version of h3-cli and restart itself.

The Runner must be active and running h3-cli version `2023-06-05` or later to receive the upgrade command. The command is delivered during the Runner's next heartbeat.

You may provide either `agent_name` or `agent_uuid` to identify the target Runner.

Response

Returns an `AgentCommandOutput!`

Arguments

`agent_name` - `String`

`agent_uuid` - `String`

Request | Response | Python

cURL | h3-cli

```
mutation UpgradeAgent($agent_name: String) {
  upgrade_agent(agent_name: $agent_name) {
    agent_command {
      uuid
      agent_uuid
      command
      received_at
      completed_at
      exit_status
      log
      created_at
    }
    message
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.7 `delete_agent`

Description

Deletes the NodeZero Runner (Agent) identified by the given `agent_name` or `agent_uuid`.

When a Runner is deleted:

- For Docker-based Runners, the associated API key is revoked.
- For Kubernetes Runners, the API key is **not** revoked because it belongs to the Operator.
- The Runner is removed from any OpTemplates it was assigned to.

You may provide either `agent_name` or `agent_uuid` to identify the target Runner.

Response

Returns an `AgentOutput!`

Arguments

`agent_name` - `String`
`agent_uuid` - `String`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteAgent($agent_uuid: String) {
  delete_agent(agent_uuid: $agent_uuid) {
    agent {
      uuid
      name
      created_at
      is_kubernetes
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.8 `operators_page`

Description

Returns a paginated list of NodeZero Kubernetes Operators belonging to the current user's ClientAccount.

Use PageInput to control pagination, sorting, and filtering. Set `include_sub_client_operators` to `true` to also include Operators from sub-client accounts (requires multi-tenancy to be enabled).

Response

Returns an `OperatorsPage!`

Arguments

`page_input` - `PageInput`

`include_sub_client_operators` - `Boolean`

Request | Response | Python

cURL | h3-cli

```
query OperatorsPage($page_input: PageInput, $include_sub_clients: Boolean) {
  operators_page(
    page_input: $page_input
    include_sub_client_operators: $include_sub_clients
  ) {
    operators {
      uuid
      name
      version
      chart_version
      is_registered
    }
    page_info {
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.9 `operators_count`

Description

Returns the total count of NodeZero Kubernetes Operators belonging to the current user's ClientAccount.

Supports the same filtering criteria as operators_page via PageInput. Set `include_sub_client_operators` to `true` to include Operators from sub-client accounts.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

`include_sub_client_operators` - `Boolean`

Request | Response | Python

cURL | h3-cli

```
query OperatorsCount($page_input: PageInput, $include_sub_clients: Boolean) {
  operators_count(
    page_input: $page_input
    include_sub_client_operators: $include_sub_clients
  )
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.10 `create_operator`

Description

Creates a new NodeZero Kubernetes Operator and returns the Helm install command needed to deploy it to your cluster.

The Operator manages the lifecycle of NodeZero Runners within Kubernetes. Once installed, you can create Runners as Kubernetes custom resources via install_agent with `is_kubernetes: true`.

This mutation also creates a new API key with `NODEZERO_OPERATOR` permissions. The key is embedded in the Helm install command.

- `operator_name` (required): The name for the Operator.
- `use_cilium`: Set to `true` if the cluster uses Cilium, to include a `CiliumNetworkPolicy` in the deployment. Defaults to `false`.
- `use_kyverno`: Set to `true` if the cluster uses Kyverno, to include a Kyverno `PolicyException` in the deployment. Defaults to `false`.

Response

Returns a `CreateOperatorOutput!`

Arguments

`operator_name` - `String!`
`use_cilium` - `Boolean`
Default = `false`
`use_kyverno` - `Boolean`
Default = `false`

Request | Response | Python
cURL | h3-cli

```
mutation CreateOperator($operator_name: String!, $use_cilium: Boolean, $use_kyverno: Boolean) {
  create_operator(
    operator_name: $operator_name
    use_cilium: $use_cilium
    use_kyverno: $use_kyverno
  ) {
    command
    operator {
      uuid
      name
      version
      chart_version
      is_registered
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.11 `upgrade_operator`

Description

Returns a Helm upgrade command to upgrade the NodeZero Kubernetes Operator Helm chart to the latest available version.

Run the returned command against your Kubernetes cluster to apply the upgrade.

Response

Returns an `UpgradeOperatorOutput!`

Request	Response	Python
cURL	h3-cli	

```
mutation UpgradeOperator {
  upgrade_operator {
    command
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.12 `delete_operator`

Description

Deletes the NodeZero Kubernetes Operator identified by the given `operator_uuid`.

Deleting an Operator also revokes and deletes the API key associated with it. Any Runners managed by this Operator should be removed from the cluster before deleting the Operator.

Response

Returns an `Operator`

Arguments

`operator_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteOperator($operator_uuid: String!) {
  delete_operator(operator_uuid: $operator_uuid) {
    uuid
    name
    version
    chart_version
    is_registered
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.13 `agents_facets`

Description

Returns the available facet (filter) values for the Runner list, along with occurrence counts for each value.

Use these facets to build dynamic filter controls for the agents_page query. Supports the same filtering criteria via PageInput. Set `include_sub_client_agents` to `true` to include facet values from sub-client accounts.

Response

Returns an `AgentsFacets!`

Arguments

`page_input` - `PageInput`
`include_sub_client_agents` - `Boolean`

Request | Response | Python
cURL | h3-cli

```
query AgentsFacets($page_input: PageInput, $include_sub_clients: Boolean) {
  agents_facets(
    page_input: $page_input
    include_sub_client_agents: $include_sub_clients
  ) {
    kubernetes_namespace {
      label
      value
      value_count
    }
    user_account_name {
      label
      value
      value_count
    }
    user_role_id {
      label
      value
      value_count
    }
    page_info {
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.14 `agent_commands_facets`

Description

Returns the available facet (filter) values for a specific Runner's command list, along with occurrence counts for each value.

The `agent_uuid` parameter identifies the Runner whose commands should be analyzed.

Response

Returns an `AgentCommandsFacets!`

Arguments

`page_input` - `PageInput`

`agent_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query AgentCommandsFacets(
  $agent_uuid: String!
  $page_input: PageInput
) {
  agent_commands_facets(
    agent_uuid: $agent_uuid
    page_input: $page_input
  ) {
    exit_status {
      label
      value
      value_count
    }
    page_info {
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.15 `operators_facets`

Description

Returns the available facet (filter) values for the Operator list, along with occurrence counts for each value.

Use these facets to build dynamic filter controls for the operators_page query. Supports the same filtering criteria via PageInput. Set `include_sub_client_operators` to `true` to include facet values from sub-client accounts.

Response

Returns an `OperatorsFacets!`

Arguments

`page_input` - `PageInput`

`include_sub_client_operators` - `Boolean`

Request | Response | Python

cURL | h3-cli

```
query OperatorsFacets(
  $page_input: PageInput
  $include_sub_client_operators: Boolean
) {
  operators_facets(
    page_input: $page_input
    include_sub_client_operators: $include_sub_client_operators
  ) {
    version {
      label
      value
      value_count
    }
    chart_version {
      label
      value
      value_count
    }
    user_role_id {
      label
      value
      value_count
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.16 Agent

Description

Represents a NodeZero Runner.

A NodeZero Runner is a lightweight agent that runs on your infrastructure and enables the automated deployment and execution of NodeZero pentests. Runners periodically heartbeat to the platform to receive and execute commands such as launching an Op, upgrading, or caching credentials.

Note: Runners are referred to as "Agents" throughout the API.

Fields

`uuid` - `String!`
The unique identifier of the Runner.
`name` - `String!`
The name of the Runner. This name is used when assigning scheduled ops or OpTemplates to a specific Runner.
`user_account_name` - `String`
The display name of the user who most recently started or registered this Runner.
`api_key_uuid` - `String!`
The unique identifier of the API key associated with this Runner. The API key is used by the Runner to authenticate with the platform.
`api_key` - `APIKey!`
The API key associated with this Runner. The API key is created automatically when the Runner is installed via install_agent.
`uname` - `String`
The output of the `uname` system command on the Runner's host machine. Provides operating system and kernel information.
`log_file` - `String`
The path to the log file on the Runner's host machine.
`last_heartbeat_at` - `Datetime`
The timestamp of the Runner's most recent heartbeat to the platform. A heartbeat occurs each time the Runner contacts the API to check for new commands.
`last_heartbeat_time_ago` - `String`
A human-friendly representation of `last_heartbeat_at`, for example `"3 minutes ago"`. Returns `null` if the Runner has never sent a heartbeat.
`last_heartbeat_source_ip` - `String`
The source IP address observed during the Runner's most recent heartbeat.
`last_command` - `AgentCommand`
The most recent AgentCommand for this Runner (either pending or already executed).
`commands` - `[AgentCommand]`
The last `last_n` commands for this Runner, ordered by most recent first. Defaults to 20; maximum is 100.

Arguments

`last_n` - `Int`
`commands_page` - `AgentCommandsPage`
A paginated list of commands for this Runner. Supports filtering and sorting via PageInput.

Arguments

`page_input` - `PageInput`
`created_at` - `Datetime!`
The timestamp when this Runner record was first created.
`h3_cli_version` - `String`
The version of the h3-cli software running on this Runner. If the Runner has sent at least one heartbeat but has not reported a version, a default beta version string is returned.

`download_url` - `String`

The URL this Runner's installer/bundle is downloading from. Reported by the Runner on register/heartbeat.

`is_kubernetes` - `Boolean!`

Indicates whether this Runner is deployed in a Kubernetes environment.

`kubernetes_namespace` - `String`

The Kubernetes namespace where this Runner is deployed. Only applicable when `is_kubernetes` is `true`.

`active_op_id` - `String`

The Op ID of any currently active (running) pentest being executed by this Runner. Returns `null` if the Runner is idle.

`op_next_scheduled_at` - `Datetime`

The next scheduled execution date for any Op assigned to this Runner. Computed from the cron expressions of all scheduled actions targeting this Runner.

`supports_runner_queueing` - `Boolean`

Indicates whether this Runner's h3-cli version supports runner queueing (version date >= 2026-03-04).

`queue` - `[Op]`

All active and queued ops for this Runner, ordered with the currently active (running) op first, followed by queued ops in priority order.

`queued_ops_count` - `Int!`

Number of ops queued for this agent (queue_priority IS NOT NULL).

Request Response

```
fragment AgentFragment on Agent {
  uuid
  name
  user_account_name
  api_key_uuid
  api_key
  uname
  log_file
  last_heartbeat_at
  last_heartbeat_time_ago
  last_heartbeat_source_ip
  last_command
  commands
  created_at
  h3_cli_version
  download_url
  is_kubernetes
  kubernetes_namespace
  active_op_id
  op_next_scheduled_at
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.17 Operator

Description

A Kubernetes (K8s) Operator that manages the lifecycle of NodeZero Runners (Agents) within a Kubernetes cluster.

The Operator is installed via a Helm chart (see create_operator) and manages the deployment, scaling, and upgrading of Runners as Kubernetes custom resources.

Fields

`uuid` - `ID!`
The unique identifier of the Operator.
`name` - `String!`
The name of the Operator, as specified when it was created via create_operator.
`version` - `String`
The software version of the Operator, as reported when the Operator registers itself with the platform.
`chart_version` - `String`
The version of the Helm chart used to install or upgrade the Operator.
`is_registered` - `Boolean!`
Indicates whether the Operator has successfully registered itself with the platform. An Operator registers automatically after the Helm chart is installed and the Operator pod starts running.

Request Response

```
fragment OperatorFragment on Operator {
  uuid
  name
  version
  chart_version
  is_registered
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.18 AgentCommand

Description

Represents a command that has been queued or executed by a NodeZero Runner (see Agent).

Commands are created by the platform (for example, via hello_agent or upgrade_agent) and delivered to the Runner during its next heartbeat. The Runner then executes the command and reports back the exit status and log output.

Fields

`uuid` - `String!`
The unique identifier of this command.
`agent_uuid` - `String!`
The unique identifier of the Agent (Runner) assigned to execute this command.
`command` - `String!`
The command string to be executed by the Runner (for example, `hello-world`, `upgrade`, or `run-nodezero <op_id>`).
`received_at` - `Datetime`
The timestamp when the Runner received the command during a heartbeat. `null` if the command has not yet been picked up.
`completed_at` - `Datetime`
The timestamp when the Runner finished executing the command. `null` if the command has not yet completed.
`exit_status` - `String`
The exit status (return code) from the command process. A value of `"0"` indicates success.
`log` - `String`
The combined stdout and stderr log output from the command process. The log is retrieved from cloud storage and may be truncated for very large outputs.
`created_at` - `Datetime!`
The timestamp when this command record was created.

Request Response

```
fragment AgentCommandFragment on AgentCommand {
  uuid
  agent_uuid
  command
  received_at
  completed_at
  exit_status
  log
  created_at
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.19 AgentCommandOutput

Description

Output type for mutations that create or update AgentCommands, such as upgrade_agent and hello_agent.

Fields

`agent_command` - `AgentCommand`
The AgentCommand that was created or updated by the mutation.

`message` - `String`
An optional informational message about the result of the mutation.

Request Response

```
fragment AgentCommandOutputFragment on AgentCommandOutput {
  agent_command {
    uuid
    agent_uuid
    command
    received_at
    completed_at
    exit_status
    log
    created_at
  }
  message
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.20 AgentCommandsFacets

Description

Facet values for filtering AgentCommands. Each facet field returns the distinct values present in the current dataset along with their occurrence counts, enabling you to build dynamic filter controls.

Fields

`page_info` - `PageInfo`
Pagination metadata for the facet result set.
`exit_status` - `[FacetValue]`
The distinct AgentCommand.exit_status values present in the data after applying any active filters, along with the count of commands for each value.

Request Response

```
fragment AgentCommandsFacetsFragment on AgentCommandsFacets {
  exit_status {
    label
    value
    value_count
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.21 AgentCommandsPage

Description

A paginated list of AgentCommands for a given Runner.

Fields

`agent_commands` - `[AgentCommand]!`
The list of AgentCommands for the current page.
`page_info` - `PageInfo`
Pagination metadata for navigating the result set.
`total_count` - `Int!`
The total number of commands matching the query, across all pages.

Request Response

```
fragment AgentCommandsPageFragment on AgentCommandsPage {
  agent_commands {
    uuid
    agent_uuid
    command
    exit_status
    received_at
    completed_at
    created_at
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.22 AgentInput

Description

Input type for Mutation.hello_agent.

Identifies a NodeZero Runner by name. The `name` field is required and must match an existing Runner in your ClientAccount.

Fields

`name` - `String!`
The name of the Runner. Must match the name of an existing Agent in your account.

Request

```
mutation HelloAgent($input: AgentInput! = {
  name: "runner-production"
}) {
  hello_agent(input: $input) {
    uuid
    agent_uuid
    command
    created_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.23 AgentOutput

Description

Output type for mutations that operate on Agents, such as delete_agent.

Fields

`agent` - `Agent`
The Agent record that was affected by the mutation.

Request Response

```
fragment AgentOutputFragment on AgentOutput {
  agent {
    uuid
    name
    user_account_name
    uname
    last_heartbeat_at
    last_heartbeat_time_ago
    h3_cli_version
    download_url
    created_at
    is_kubernetes
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.24 AgentsFacets

Description

Facet values for filtering Agents (NodeZero Runners). Each facet field returns the distinct values present in the current dataset along with their occurrence counts, enabling you to build dynamic filter controls for the Runner list.

Fields

`page_info` - `PageInfo`
Pagination metadata for the facet result set.
`kubernetes_namespace` - `[FacetValue]`
The distinct Agent.kubernetes_namespace values present in the data after applying any active filters, along with the count of Runners for each namespace.
`user_account_name` - `[FacetValue]`
The distinct Agent.user_account_name values present in the data after applying any active filters, along with the count of Runners for each user.
`user_role_id` - `[FacetValue]`
The distinct APIKey role values (`user_role_id`) present in the data after applying any active filters, along with the count of Runners for each role.

Request Response

```
fragment AgentsFacetsFragment on AgentsFacets {
  kubernetes_namespace
  user_account_name
  user_role_id
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.25 AgentsPage

Description

A paginated list of NodeZero Runners (Agents).

Fields

`page_info` - `PageInfo`
Pagination metadata for navigating the result set.
`agents` - `[Agent!]!`
The list of Agents (Runners) for the current page.

Request Response

```
fragment AgentsPageFragment on AgentsPage {
  agents {
    uuid
    name
    user_account_name
    created_at
    is_kubernetes
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.26 CreateOperatorOutput

Description

Output type for create_operator.

Contains the Helm install command, the newly created API key, and the Operator record.

Fields

`command` - `String!`
The Helm install command to run against your Kubernetes cluster. This command deploys the NodeZero Operator Helm chart and configures it with the generated API key.

`api_key` - `APIKey!`
The API key that was created for the Operator. The key has `NODEZERO_OPERATOR` permissions and is embedded in the Helm install command.

`operator` - `Operator!`
The Operator record that was created in the platform.

Request Response

```
fragment CreateOperatorOutputFragment on CreateOperatorOutput {
  command
  operator {
    uuid
    name
    version
    chart_version
    is_registered
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.27 InstallAgentOutput

Description

Output type for install_agent.

Contains the shell command needed to install and start a NodeZero Runner on the target host, along with the API key and Agent record created for the Runner.

Fields

`command` - `String`

The shell command for installing the NodeZero Runner on a Docker host. Run this command on the machine where you want the Runner to operate. It downloads, installs, and starts the Runner process.

This field is populated only when `is_kubernetes` is `false` (or omitted) in install_agent. For Kubernetes deployments, use the `kubernetes_install_command` field instead.

`kubernetes_install_command` - `String`

The `kubectl` command for deploying the NodeZero Runner as a custom resource in a Kubernetes cluster. Run this command against the cluster where the Operator is installed.

This field is populated only when `is_kubernetes` is `true` in install_agent. For non-Kubernetes deployments, use the `command` field instead.

`api_key` - `APIKey`
The API key that was created for the Runner. For Docker-based Runners, the key has `NODEZERO_RUNNER` permissions. For Kubernetes Runners, the existing Operator's `NODEZERO_OPERATOR` key is returned.
`agent` - `Agent`
The Agent record representing the newly created Runner. This record is created immediately and can be referenced by its `uuid` to assign the Runner to an OpTemplate or Op even before the Runner has been installed and started on the host.

Request Response

```
fragment InstallAgentOutputFragment on InstallAgentOutput {
  command
  kubernetes_install_command
  agent {
    uuid
    name
    created_at
    is_kubernetes
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.28 OperatorsFacets

Description

Facet values for filtering Operators. Each facet field returns the distinct values present in the current dataset along with their occurrence counts, enabling you to build dynamic filter controls for the Operator list.

Fields

`page_info` - `PageInfo`
Pagination metadata for the facet result set.
`version` - `[FacetValue]`
The distinct Operator.version values present in the data after applying any active filters, along with the count of Operators for each version.
`chart_version` - `[FacetValue]`
The distinct Operator.chart_version values present in the data after applying any active filters, along with the count of Operators for each Helm chart version.
`user_role_id` - `[FacetValue]`
The distinct APIKey role values (`user_role_id`) present in the data after applying any active filters, along with the count of Operators for each role.

Request Response

```
fragment OperatorsFacetsFragment on OperatorsFacets {
  page_info {
    page_num
    page_size
  }
  version {
    label
    value
    value_count
  }
  chart_version {
    label
    value
    value_count
  }
  user_role_id {
    label
    value
    value_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.29 OperatorsPage

Description

A paginated list of NodeZero Kubernetes Operators.

Fields

`page_info` - `PageInfo`
Pagination metadata for navigating the result set.
`operators` - `[Operator!]!`
The list of Operators for the current page.

Request Response

```
fragment OperatorsPageFragment on OperatorsPage {
  operators {
    uuid
    name
    version
    chart_version
    is_registered
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Runners

8.10.30 UpgradeOperatorOutput

Description

Output type for upgrade_operator.

Fields

`command` - `String!`
The Helm upgrade command to run against your Kubernetes cluster. This command upgrades the NodeZero Operator Helm chart to the latest available version.

Request Response

```
fragment UpgradeOperatorOutputFragment on UpgradeOperatorOutput {
  command
}
```

Examples may not include all available fields for a type.

8.11 Setup: Schedules

Automate recurring pentests and operations with flexible scheduling. Create, update, and manage scheduled actions that run pentests at specified intervals (daily, weekly, monthly) to maintain continuous security validation.

8.11.1 `schedules_page`

Description

List of Schedules and ScheduledActions in the current account.

The `Schedule.uuid` field can be used to fetch more details about a specific Schedule via `Query.schedule`.

Response

Returns a `SchedulesPage!`

Arguments

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query SchedulesPage($page_input: PageInput) {
  schedules_page(page_input: $page_input) {
    schedules {
      uuid
      name
      is_disabled
      ops_count
      last_op {
        op_name
        op_state
      }
      actions {
        uuid
        action
        cadence
        cron_expression
        cron_description
        is_disabled
        schedule_uuid
        schedule_name
        op_template_uuid
        last_triggered_at
        last_triggered_time_ago
        next_triggered_at
        next_triggered_time_in
        total_triggers
      }
    }
    page_info {
```

```
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.2 `schedules_count`

Description

Count of Schedules in the current account.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query SchedulesCount($page_input: PageInput) {
  schedules_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.3 `schedule`

Description

Get the Schedule with the given uuid or name. Provide either `uuid` or `name` to identify the schedule.

Response

Returns a `Schedule`

Arguments

`uuid` - `String`
The uuid of the Schedule to fetch.
`name` - `String`
The name of the Schedule to fetch.

Request | Response | Python
cURL | h3-cli

```
query Schedule($uuid: String, $name: String) {
  schedule(uuid: $uuid, name: $name) {
    uuid
    name
    is_disabled
    ops_count
    last_op {
      op_name
      op_state
      op_type
    }
    actions {
      uuid
      action
      cadence
      cron_expression
      cron_description
      is_disabled
      schedule_uuid
      schedule_name
      op_template_uuid
      last_triggered_at
      last_triggered_time_ago
      next_triggered_at
      next_triggered_time_in
      total_triggers
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.4 `create_scheduled_action`

Description

Create or update a ScheduledAction.

ScheduledActions are actions within a Schedule that run at a specific time, similar to a cron job. A Schedule is automatically created if one doesn't already exist for the given schedule_name.

A ScheduledAction can be used to run a pentest on a recurring basis, eg. weekly or monthly.

You can also use ScheduledActions to pause, resume, and cancel a pentest at specific times, for example to pause a pentest during a maintenance window. See `CreateScheduledActionInput.action` for more details.

Response

Returns a `ScheduledActionOutput!`

Arguments

`input` - `CreateScheduledActionInput!`

Request | Response | Python
cURL | h3-cli

```
mutation CreateScheduledAction($input: CreateScheduledActionInput!) {
  create_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      action
      cadence
      cron_expression
      cron_description
      is_disabled
      schedule_uuid
      schedule_name
      op_template_uuid
      last_triggered_at
      next_triggered_at
      next_triggered_time_in
      total_triggers
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.5 `trigger_scheduled_action`

Description

Trigger a ScheduledAction to run now.

This mutation can be used to verify a ScheduledAction's configuration and ensure it behaves as expected. It can also be used as a convenient way to run a pentest immediately, without waiting for the next scheduled time.

Response

Returns a `ScheduledActionOutput!`

Arguments

`input` - `TriggerScheduledActionInput!`

Request | Response | Python

cURL | h3-cli

```
mutation TriggerScheduledAction($input: TriggerScheduledActionInput!) {
  trigger_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      action
      schedule_name
      last_triggered_at
      last_triggered_time_ago
      next_triggered_at
      next_triggered_time_in
      total_triggers
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.6 `update_scheduled_action`

Description

Update a ScheduledAction.

Response

Returns a `ScheduledActionOutput!`

Arguments

`input` - `UpdateScheduledActionInput!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateScheduledAction($input: UpdateScheduledActionInput!) {
  update_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      action
      cadence
      cron_expression
      cron_description
      is_disabled
      schedule_uuid
      schedule_name
      op_template_uuid
      last_triggered_at
      next_triggered_at
      next_triggered_time_in
      total_triggers
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.7 `delete_scheduled_action`

Description

Delete a ScheduledAction. To fully delete a Schedule, delete all ScheduledActions within it.

Response

Returns a `ScheduledActionOutput!`

Arguments

`input` - `DeleteScheduledActionInput!`

Request | Response | Python

cURL | h3-cli

```
mutation DeleteScheduledAction($input: DeleteScheduledActionInput!) {
  delete_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      action
      schedule_name
      is_disabled
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.8 `enable_schedule`

Description

Enable the specified Schedule. All ScheduledActions within the schedule will execute at their scheduled times.

Response

Returns a `ScheduleOutput!`

Arguments

`schedule_name` - `String!`

Request | Response | Python
cURL | h3-cli

```
mutation EnableSchedule($schedule_name: String!) {
  enable_schedule(schedule_name: $schedule_name) {
    schedule {
      ...ScheduleFragment
    }
  }
}

fragment ScheduleFragment on Schedule {
  uuid
  name
  state
  is_disabled
  ops_count
  total_triggers
  actions {
    uuid
    action
    cadence
    cron_expression
    cron_description
    is_disabled
    next_triggered_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.9 `disable_schedule`

Description

Disable the specified Schedule. All ScheduledActions within the schedule will not execute until the Schedule is re-enabled.

Response

Returns a `ScheduleOutput!`

Arguments

`schedule_name` - `String!`

Request | Response | Python
cURL | h3-cli

```
mutation DisableSchedule($schedule_name: String!) {
  disable_schedule(schedule_name: $schedule_name) {
    schedule {
      ...ScheduleFragment
    }
  }
}

fragment ScheduleFragment on Schedule {
  uuid
  name
  state
  is_disabled
  ops_count
  total_triggers
  actions {
    uuid
    action
    cadence
    cron_expression
    cron_description
    is_disabled
    next_triggered_at
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.10 Schedule

Description

A Schedule is a set of `ScheduledActions` that can be used together to control the execution of a pentest on a recurring basis.

Use `Mutation.create_scheduled_action` to create a Schedule.

Typically a Schedule has only one ScheduledAction, with its action set to `run-pentest`. This is used to run a pentest at a specific date and time on a recurring basis.

A Schedule may further control the execution of pentests by using additional ScheduledActions with actions set to `pause-pentest`, `resume-pentest`, and/or `cancel-pentest`, for pausing, resuming, and canceling/ending the pentest respectively.

Visit Schedules in the Horizon3.ai documentation for more details on using Schedules to automate pentest execution.

Fields

`uuid` - `String!`
Unique ID of the Schedule.
`name` - `String!`
The Schedule name.
`op_template_name` - `String`
Name of the OpTemplate assigned to this schedule's `run-pentest` action.
`actions` - `[ScheduledAction]`
The ScheduledActions defined in this schedule.
`created_at` - `Datetime!`
When the Schedule was created.
`last_updated_at` - `Datetime`
When the Schedule was last updated.
`is_disabled` - `Boolean`
Whether all actions in this schedule are disabled. When `true`, no actions will execute at their scheduled times. Use `Mutation.enable_schedule` or `Mutation.disable_schedule` to toggle.
`state` - `String`
Display label for the schedule's enabled/disabled status. Returns `ENABLED` or `DISABLED`.
`ops_count` - `Int!`
The number of operations that have been run under this schedule.

Arguments

`page_input` - `PageInput`
`ops_page` - `OpsPage!`
Paginated list of operations run under this schedule. See Op for operation details.

Arguments

`page_input` - `PageInput`
`run_pentest_action` - `ScheduledAction`
The `run-pentest` ScheduledAction for this schedule, if one exists.
`op_name` - `String`
Name of op, as defined in the op template associated with schedule.
`last_op` - `Op`
The most recent Op that ran (or is currently running) under this schedule.
`total_triggers` - `Int`
Number of times the schedule has run.

Request Response

```
fragment ScheduleFragment on Schedule {
  uuid
  name
  is_disabled
  ops_count
  last_op {
    op_name
    op_state
    op_type
  }
  actions {
    uuid
    action
    cadence
    cron_expression
    cron_description
    is_disabled
    op_template_uuid
    last_triggered_at
    next_triggered_at
    total_triggers
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.11 ScheduledAction

Description

An action to run at a specific time on a recurring schedule, like a cron job. ScheduledActions are useful for running pentests on a recurring basis, such as weekly or monthly.

There are four types of actions:

- `run-pentest` : Run a pentest using the OpTemplate configured in the ScheduledAction.
- `pause-pentest` : Pause a pentest that is currently running.
- `resume-pentest` : Resume a paused pentest.
- `cancel-pentest` : Cancel a pentest that is currently running.

Visit Schedules in the Horizon3.ai documentation for more details on using Schedules to automate pentest execution.

Fields

`uuid` - `String!`
Unique ID of the ScheduledAction.
`schedule_name` - `String!`
The name of the Schedule that the ScheduledAction belongs to. A Schedule is a collection of ScheduledActions.
`schedule_uuid` - `String!`
ID of the Schedule that the ScheduledAction belongs to. A Schedule is a collection of ScheduledActions.
`action` - `String!`

The action to execute. Supported actions are:

- run-pentest: Run a pentest using the OpTemplate configured in the ScheduledAction.
- pause-pentest: Pause a pentest that is currently running.
- resume-pentest: Resume a paused pentest.
- cancel-pentest: Cancel a pentest that is currently running.

`cron_expression` - `String!`

A cron expression that defines when the action is executed.

A cron expression is a string consisting of 5 fields separated by spaces. The 5 fields represent the following:

- minute (0–59) - this field must always be set to '*' or '0'.
- hour (0–23) - note this is in UTC time.
- day of the month (1–31)
- month (1–12)
- day of the week (0–6) (Sunday to Saturday)

Notes:

- ScheduledActions only support hourly resolution. The minutes field must always be set to '*' or '0'.
- The cron expression is configured in UTC time. If you want to run the action at a specific time in another timezone, you will need to convert that time to UTC.

- runs at midnight UTC every saturday

`cron_description` - `String`
A human-readable description of the cron schedule, e.g. 'Every Saturday at midnight UTC'.
`op_template_uuid` - `String`

This field is used with the `run-pentest` action. This is the uuid of the OpTemplate that will be used to run the pentest when the ScheduledAction is triggered.

`op_template` - `OpTemplate`

This field is used with the `run-pentest` action. This is the OpTemplate that will be used to run the pentest when the ScheduledAction is triggered.

`last_triggered_at` - `Datetime`

The last time the action was triggered.

`last_triggered_time_ago` - `String`

`last_triggered_at` as a human-friendly "time ago" statement, eg. "3 minutes ago"

`last_triggered_error` - `String`

The error message, if an error occurred the last time the action was triggered.

`next_triggered_at` - `Datetime`

The next time the action will be triggered.

`next_triggered_time_in` - `String`

`next_triggered_at` as a human-friendly statement, eg. "3 minutes from now"

`next_triggered_ats` - `[Datetime]`

The next scheduled execution times. Use the `next_n` argument to specify how many upcoming times to return.

Arguments

`next_n` - `Int`

`cadence` - `String`

The cadence text (Daily/Weekly/Monthly/Quarterly/Custom) of the schedule.

`is_disabled` - `Boolean`

Whether this scheduled action is disabled. Disabled actions will not execute at their scheduled times.

`total_triggers` - `Int`

Number of times the schedule has run.

Request Response

```
fragment ScheduledActionFragment on ScheduledAction {
  uuid
  action
  cadence
  cron_expression
  cron_description
  is_disabled
  schedule_uuid
  schedule_name
  op_template_uuid
  last_triggered_at
  last_triggered_time_ago
  last_triggered_error
  next_triggered_at
  next_triggered_ats
  next_triggered_time_in
  total_triggers
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.12 CreateScheduledActionInput

Description

Input type for `Mutation.create_scheduled_action`

Fields

`schedule_name` - `String!`
The name of the Schedule that the ScheduledAction belongs to. A Schedule is a collection of ScheduledActions.
`action` - `String!`

The action to execute. Supported actions are:

- run-pentest: Run a pentest using the OpTemplate configured in the ScheduledAction.
- pause-pentest: Pause a pentest that is currently running.
- resume-pentest: Resume a paused pentest.
- cancel-pentest: Cancel a pentest that is currently running.

`cron_expression` - `String!`

A cron expression that defines when the action is executed.

A cron expression is a string consisting of 5 fields separated by spaces. The 5 fields represent the following:

- minute (0–59) - this field must always be set to '*' or '0'.
- hour (0–23) - note this is in UTC time.
- day of the month (1–31)
- month (1–12)
- day of the week (0–6) (Sunday to Saturday)

Notes:

- ScheduledActions only support hourly resolution. The minutes field must always be set to '*' or '0'.
- The cron expression is configured in UTC time. If you want to run the action at a specific time in another timezone, you will need to convert that time to UTC.

`op_template_uuid` - `String`
This field is used with the `run-pentest` action. This is the OpTemplate that will be used to run the pentest when the ScheduledAction is triggered.

Request

```
mutation CreateScheduledActionExample(
  $input: CreateScheduledActionInput = {
    schedule_name: "Weekly Infrastructure Scan"
    action: "run-pentest"
    cron_expression: "0 2 * * MON"
    op_template_uuid: "9999aaaa-9999-aaaa-9999-aaaa9999aaaa"
  }
) {
  create_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      schedule_name
      action
      cron_expression
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.13 DeleteScheduledActionInput

Description

Input type for `Mutation.delete_scheduled_action`

Fields

`schedule_name` - `String!`
Name of the Schedule containing the action to delete.
`action` - `String!`
The action to delete. Supported actions: run-pentest, pause-pentest, resume-pentest, cancel-pentest

Request

```
mutation DeleteScheduledActionExample(
  $input: DeleteScheduledActionInput = {
    schedule_name: "Weekly Infrastructure Scan"
    action: "run-pentest"
  }
) {
  delete_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      schedule_name
      action
      is_disabled
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.14 ScheduleOutput

Description

Output type for `Mutation.enable_schedule` and other mutations.

Fields

`schedule` - `Schedule`
The updated Schedule.

Request Response

```
fragment ScheduleOutputFragment on ScheduleOutput {
  schedule {
    uuid
    name
    is_disabled
    state
    created_at
    last_updated_at
    ops_count
    total_triggers
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.15 ScheduledActionOutput

Description

Output type for `Mutation.create_scheduled_action` and other mutations.

Fields

`scheduled_action` - `ScheduledAction`
The created ScheduledAction

Request Response

```
fragment ScheduledActionOutputFragment on ScheduledActionOutput {
  scheduled_action {
    uuid
    action
    cadence
    cron_expression
    cron_description
    is_disabled
    schedule_uuid
    schedule_name
    op_template_uuid
    total_triggers
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.16 SchedulesPage

Description

Paginated list of Schedules.

Fields

`page_info` - `PageInfo`
Pagination of response.
`schedules` - `[Schedule!]!`
List of Schedules.

Request Response

```
fragment SchedulesPageFragment on SchedulesPage {
  schedules {
    uuid
    name
    is_disabled
    ops_count
    actions {
      uuid
      action
    }
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.17 TriggerScheduledActionInput

Description

Input type for `Mutation.trigger_scheduled_action`.

Fields

`schedule_name` - `String!`
Name of the Schedule containing the action to trigger.
`action` - `String!`
The action to trigger. Supported actions: run-pentest, pause-pentest, resume-pentest, cancel-pentest

Request

```
mutation TriggerScheduledActionExample(
  $input: TriggerScheduledActionInput = {
    schedule_name: "Weekly Infrastructure Scan"
    action: "run-pentest"
  }
) {
  trigger_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      schedule_name
      action
      last_triggered_at
      total_triggers
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Schedules

8.11.18 UpdateScheduledActionInput

Description

Input type for `Mutation.update_scheduled_action`

Fields

`uuid` - `String!`
ID of ScheduledAction to update.
`cron_expression` - `String`

A cron expression that defines when the action is executed.

A cron expression is a string consisting of 5 fields separated by spaces. The 5 fields represent the following:

- minute (0–59) - this field must always be set to '*' or '0'.
- hour (0–23) - note this is in UTC time.
- day of the month (1–31)
- month (1–12)
- day of the week (0–6) (Sunday to Saturday)

Notes:

- ScheduledActions only support hourly resolution. The minutes field must always be set to '*' or '0'.
- The cron expression is configured in UTC time. If you want to run the action at a specific time in another timezone, you will need to convert that time to UTC.

`op_template_uuid` - `String`
This field is used with the `run-pentest` action. This is the OpTemplate that will be used to run the pentest when the ScheduledAction is triggered.

Request

```
mutation UpdateScheduledActionExample(
  $input: UpdateScheduledActionInput = {
    uuid: "5678abcd-5678-abcd-5678-abcd5678abcd"
    cron_expression: "0 3 * * *"
    op_template_uuid: "9999aaaa-9999-aaaa-9999-aaaa9999aaaa"
  }
) {
  update_scheduled_action(input: $input) {
    scheduled_action {
      uuid
      cron_expression
      cron_description
    }
  }
}
```

Examples may not include all available fields for a type.

8.12 Setup: Asset Groups

Organize external attack surface assets (domains and IP ranges) into logical groups for targeted pentesting. Asset groups enable you to define and manage different segments of your external infrastructure for focused security assessments.

8.12.1 `asset_groups_page`

Description

Paginated list of `asset groups` in this account.

The `AssetGroup.uuid` field can be used to fetch more details about a specific AssetGroup using `Query.asset_group`.

Response

Returns an `AssetGroupsPage!`

Arguments

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query AssetGroupsPage($page_input: PageInput) {
  asset_groups_page(page_input: $page_input) {
    asset_groups {
      ...AssetGroupPageFragment
    }
  }
}

fragment AssetGroupPageFragment on AssetGroup {
  uuid
  name
  op_template_uuid
  user_account_name
  client_account_company_name
  last_ead_etl_completed_at
  created_at
  updated_at
  op_series_uuid
  op_series_status
  assets_count
  authorized_assets_count
  external_domain_xops_count
  in_scope_host_tab_xops_count
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.2 `asset_groups_count`

Description

The total number of `asset groups` in this account.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query AssetGroupsCount($page_input: PageInput) {
  asset_groups_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.3 `asset_group`

Description

Fetch a specific `asset group` by its unique identifier.

Response

Returns an `AssetGroup`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query AssetGroup($uuid: String!) {
  asset_group(uuid: $uuid) {
    ...AssetGroupFragment
  }
}

fragment AssetGroupFragment on AssetGroup {
  uuid
  name
  op_template_uuid
  user_account_uuid
  user_account_name
  client_account_uuid
  client_account_company_name
  last_ead_etl_completed_at
  created_at
  updated_at
  op_series_uuid
  op_series_status
  assets_count
  authorized_assets_count
  external_domain_xops_count
  authorized_external_domain_xops_count
  in_scope_host_tab_xops_count
  authorized_host_tab_xops_count
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.4 `create_asset_group`

Description

Create a new `asset group`.

The asset group's scan scope is defined by the provided `ScheduleOpFormInput`. Set the asset group name via `schedule_op_form.op_name`.

Response

Returns an `AssetGroupOutput!`

Arguments

`schedule_op_form` - `ScheduleOpFormInput!`
Scope for discovering assets.

Request | Response | Python
cURL | h3-cli

```
mutation CreateAssetGroup(
  $schedule_op_form: ScheduleOpFormInput!
) {
  create_asset_group(schedule_op_form: $schedule_op_form) {
    asset_group {
      uuid
      name
      op_template_uuid
      op_series_uuid
      op_series_status
      created_at
      assets_count
      external_domain_xops_count
      in_scope_host_tab_xops_count
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.5 `update_asset_group_template`

Description

Update an `asset group's` scan scope configuration.

The asset group name can be updated via `schedule_op_form.op_name`.

Response

Returns a `SaveOpTemplateOutput!`

Arguments

`asset_group_uuid` - `String!`
ID of asset group.
`schedule_op_form` - `ScheduleOpFormInput!`
Scope for discovering assets.

Request | Response | Python
cURL | h3-cli

```
mutation UpdateAssetGroupTemplate(
  $asset_group_uuid: String!
  $schedule_op_form: ScheduleOpFormInput!
) {
  update_asset_group_template(
    asset_group_uuid: $asset_group_uuid
    schedule_op_form: $schedule_op_form
  ) {
    op_template {
      uuid
      op_template_name
      op_type
      schedule_op_form {
        op_name
        osint_domains
      }
    }
    asset_group {
      uuid
      name
      op_series_uuid
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.6 AssetGroup

Description

An asset group represents a collection of discovered assets (domains and IPs) in your environment.

Assets are discovered by running External Asset Discovery scans using the scope defined in the asset group's associated `OpTemplate`. Once discovered, assets can be authorized for pentesting and used as the scope for External Attack operations.

Use `Mutation.create_asset_group` to create a new asset group.

Fields

`uuid` - `String!`
Unique identifier of the asset group.
`name` - `String!`
Display name of the asset group.
`op_template_uuid` - `String!`
Unique identifier of the `OpTemplate` that defines this asset group's scan scope.
`op_template` - `OpTemplate!`
The `OpTemplate` that defines this asset group's scan scope and configuration.
`user_account_uuid` - `String!`
Unique identifier of the user account that created this asset group.
`user_account_name` - `String!`
Display name of the user account that created this asset group.
`client_account_uuid` - `String!`
Unique identifier of the client account that owns this asset group.
`client_account_company_name` - `String!`
Company name of the client account that owns this asset group.
`last_ead_etl_completed_at` - `Datetime`
Timestamp when the most recent External Asset Discovery scan completed for this asset group.
`created_at` - `Datetime!`
Timestamp when this asset group was created.
`updated_at` - `Datetime`
Timestamp when this asset group was last modified.
`op_series_uuid` - `String!`
Unique identifier of the External Asset Discovery op series associated with this asset group.
`assets_count` - `Int!`
Total number of discovered assets (domains and IPs) in this asset group.
`authorized_assets_count` - `Int!`
Number of assets in this asset group that have been authorized for pentesting.
`external_domain_xops_count` - `Int!`
Total number of discovered domains in this asset group.
`external_domain_xops_page` - `ExternalDomainXopsPage!`
Paginated list of discovered domains in this asset group.

Arguments

`page_input` - `PageInput`
`authorized_external_domain_xops_count` - `Int!`
Number of domains in this asset group that have been authorized for pentesting.
`in_scope_host_tab_xops_count` - `Int!`
Total number of discovered IPs in this asset group.
`in_scope_host_tab_xops_page` - `HostTabXopsPage!`
Paginated list of discovered IPs in this asset group.

Arguments

`page_input` - `PageInput`

`authorized_host_tab_xops_count` - `Int!`

Number of IPs in this asset group that have been authorized for pentesting.

Request Response

```
fragment AssetGroupFragment on AssetGroup {
  uuid
  name
  op_template_uuid
  user_account_uuid
  user_account_name
  client_account_uuid
  client_account_company_name
  last_ead_etl_completed_at
  created_at
  updated_at
  op_series_uuid
  op_series_status
  assets_count
  authorized_assets_count
  external_domain_xops_count
  authorized_external_domain_xops_count
  in_scope_host_tab_xops_count
  authorized_host_tab_xops_count
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.7 AssetGroupOutput

Description

Output type returned by asset group mutations.

Fields

`asset_group` - `AssetGroup!`
The `AssetGroup`.

Request Response

```
fragment AssetGroupOutputFragment on AssetGroupOutput {
  asset_group {
    uuid
    name
    op_template_uuid
    op_series_uuid
    op_series_status
    created_at
    updated_at
    assets_count
    authorized_assets_count
    external_domain_xops_count
    in_scope_host_tab_xops_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups

8.12.8 AssetGroupsPage

Description

Contains a paginated list of `AssetGroup` records.

Fields

`page_info` - `PageInfo`
Page details.
`asset_groups` - `[AssetGroup!]!`
List of `asset groups`.

Request Response

```
fragment AssetGroupsPageFragment on AssetGroupsPage {
  page_info {
    page_num
    page_size
  }
  asset_groups {
    uuid
    name
    op_template_uuid
    user_account_name
    last_ead_etl_completed_at
    created_at
    updated_at
    op_series_uuid
    op_series_status
    assets_count
    authorized_assets_count
    external_domain_xops_count
    in_scope_host_tab_xops_count
  }
}
```

Examples may not include all available fields for a type.

8.13 Setup: Asset Groups: Domains

Manage domains within asset groups for external pentesting. Query discovered domains, view their testing history across operations, and add or remove domains from asset groups to control your external attack surface scope.

8.13.1 `external_domain_xops_page`

Description

Returns a paginated list of ExternalDomainXop records (domains) in the given `AssetGroup.op_series_uuid`. Supports filtering, sorting, and pagination via `page_input`.

Response

Returns an `ExternalDomainXopsPage!`

Arguments

`op_series_uuid` - `String!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query ExternalDomainXopsPage(
  $op_series_uuid: String!
  $page_input: PageInput
) {
  external_domain_xops_page(
    op_series_uuid: $op_series_uuid
    page_input: $page_input
  ) {
    external_domain_xops {
```

```
      ...ExternalDomainXopPageFragment
    }
  }
}

fragment ExternalDomainXopPageFragment on ExternalDomainXop {
  uuid
  op_series_uuid
  xop_id
  last_op_id
  current_op_id
  is_authorized
  is_dynamic_ip
  third_party_aliases
  pentestable_rules {
    is_allowed
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.2 `external_domain_xops_count`

Description

Returns the total number of ExternalDomainXop records (domains) in the given `AssetGroup.op_series_uuid`. Supports filtering via `page_input`.

Response

Returns an `Int!`

Arguments

`op_series_uuid` - `String!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query ExternalDomainXopsCount(
  $op_series_uuid: String!
  $page_input: PageInput
) {
  external_domain_xops_count(
    op_series_uuid: $op_series_uuid
    page_input: $page_input
  )
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.3 `external_domain_xop`

Description

Returns the ExternalDomainXop with the given `uuid`. The UUID is in the format `{op_series_uuid}/{domain_name}`.

Response

Returns an `ExternalDomainXop`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query ExternalDomainXop($uuid: String!) {
  external_domain_xop(uuid: $uuid) {
    ...ExternalDomainXopFragment
  }
}

fragment ExternalDomainXopFragment on ExternalDomainXop {
  uuid
  op_series_uuid
  xop_id
  last_op_id
  current_op_id
  is_authorized
  is_dynamic_ip
  third_party_aliases
  third_party_certificate_subject_cns
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
  }
  excluded_domain_from_last_pentest {
    op_id
    domain
    reason
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.4 `add_domains_to_asset_group`

Description

Adds one or more domains to the given AssetGroup's configured scope. The new domains are merged into the AssetGroup's existing domain list. Duplicate domains are automatically deduplicated. Returns the updated AssetGroup.

Response

Returns an `AssetGroupOutput!`

Arguments

`asset_group_uuid` - `String!`
The UUID of the AssetGroup to update.
`domains` - `[StringNotEmpty]!`
List of domain names to add to the AssetGroup's scope (e.g., `["example.com", "sub.example.com"]`).

Request | Response | Python
cURL | h3-cli

```
mutation AddDomainsToAssetGroup(
  $asset_group_uuid: String!
  $domains: [StringNotEmpty]!
) {
  add_domains_to_asset_group(
    asset_group_uuid: $asset_group_uuid
    domains: $domains
  ) {
    asset_group {
      uuid
      name
      op_series_uuid
      assets_count
      external_domain_xops_count
      authorized_external_domain_xops_count
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.5 `remove_domains_from_asset_group`

Description

Removes one or more domains from the given AssetGroup's configured scope. Returns the updated AssetGroup.

Response

Returns an `AssetGroupOutput!`

Arguments

`asset_group_uuid` - `String!`
The UUID of the AssetGroup to update.
`domains` - `[StringNotEmpty]!`
List of domain names to remove from the AssetGroup's scope.

Request | Response | Python
cURL | h3-cli

```
mutation RemoveDomainsFromAssetGroup(
  $asset_group_uuid: String!
  $domains: [StringNotEmpty]!
) {
  remove_domains_from_asset_group(
    asset_group_uuid: $asset_group_uuid
    domains: $domains
  ) {
    asset_group {
      uuid
      name
      op_series_uuid
      assets_count
      external_domain_xops_count
      authorized_external_domain_xops_count
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.6 `external_domain_xops_csv_presigned_url`

Description

Returns a temporary presigned URL for downloading a CSV export of all ExternalDomainXop records (domains) in the given `AssetGroup.op_series_uuid`.

The CSV format is documented under ExternalDomainXopCSV.

Response

Returns a `String`

Arguments

`op_series_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query ExternalDomainXopsCsvPresignedUrl(
  $op_series_uuid: String!
) {
  external_domain_xops_csv_presigned_url(
    op_series_uuid: $op_series_uuid
  )
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.7 ExternalDomainXop

Description

An ExternalDomainXop represents a unique external domain name tracked across a series of Ops within an AssetGroup. Each ExternalDomainXop is uniquely identified by its `xop_id`, which is the domain name itself (e.g., `example.com`).

As new pentests run within the same AssetGroup, the ExternalDomainXop record is updated with data from the most recent Op while retaining historical tracking information such as the first and last time the domain appeared.

Fields

`uuid` - `String!`
Unique identifier for this ExternalDomainXop, in the format `{op_series_uuid}/{xop_id}`.
`op_series_uuid` - `String!`
The ID of the OpSeries (and corresponding AssetGroup) that this domain is tracked under.
`xop_id` - `String!`
The durable cross-op identity for this asset. Always derived from the data itself rather than an arbitrary UUID. For ExternalDomainXop, the `xop_id` is the domain name (e.g., `example.com`).
`last_external_domain` - `ExternalDomain!`
The underlying ExternalDomain record from the most recent Op in which this domain name was discovered.
`last_external_domains` - `[ExternalDomain]!`
All underlying ExternalDomain records from the most recent Op in which this domain name was discovered. A domain name may resolve to multiple ExternalDomain records within a single Op.
`last_op_id` - `String`
The ID of the most recent Op in which this domain name was discovered.
`current_op_id` - `String`
The ID of the most recently run Op in the OpSeries.
`is_authorized` - `Boolean`
Indicates whether this domain has been authorized for external pentesting by the user. Domains must be authorized before they can be included in an External Attack pentest. Use authorize_domains or deauthorize_domains to change this value.
`pentestable_rules` - `PentestableRules`
The PentestableRules evaluation result for this domain. These rules determine whether the user is permitted to authorize this domain for external pentesting. Domains may be blocked if they are determined to belong to third-party providers that do not permit pentesting.
`is_dynamic_ip` - `Boolean`
Indicates whether the user has flagged this domain as using dynamic IP addresses. When set to `true`, the platform accounts for the fact that the domain's resolved IP may change between pentests.
`excluded_domain_from_last_pentest` - `ExcludedDomain`
The ExcludedDomain record from the most recent External Attack pentest against this AssetGroup. This field is non-null when the domain was authorized for pentesting but was subsequently excluded from scope during the pentest, typically due to DNS drift or unreachability.
`third_party_aliases` - `[String]`
Full list of third-party aliases for this domain. These are subdomains that are NOT covered by a top-level domain (TLD) configured in the AssetGroup. Aliases include CNAME chain entries and DNS reverse-lookup hostnames.
`third_party_certificate_subject_cns` - `[String]`
Third-party certificate subject Common Names (CNs). Includes all certificate CNs that are NOT covered by a top-level domain (TLD) configured in the AssetGroup.

Request Response

```
fragment ExternalDomainXopFragment on ExternalDomainXop {
  uuid
  op_series_uuid
  xop_id
  last_op_id
  current_op_id
  is_authorized
```

```
  is_dynamic_ip
  third_party_aliases
  third_party_certificate_subject_cns
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
    authz_warning_label
  }
  excluded_domain_from_last_pentest {
    op_id
    domain
    reason
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.8 ExcludedDomain

Description

Represents an external domain that was authorized for pentesting but was excluded from the pentest at runtime. Domains are typically excluded due to DNS drift (the domain's IP changed), unreachability, or the domain no longer resolving. See ExcludedReason for the possible reasons.

Fields

`op_id` - `String!`
The ID of the Op (pentest) during which the domain was excluded.
`domain` - `String!`
The domain name that was excluded from the pentest.
`reason` - `ExcludedReason!`
The reason the domain was excluded. See ExcludedReason for possible values.

Request Response

```
fragment ExcludedDomainFragment on ExcludedDomain {
  op_id
  domain
  reason
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Domains

8.13.9 ExcludedReason

Description

Reasons why an external domain was excluded from a pentest.

Values

`Unreachable`
The domain's IP was not reachable.
`Unresolved`
The domain could not be resolved.
`IPChanged`
The domain's IP changed

Example

```
"Unreachable"
```

Setup: Asset Groups: Domains

8.13.10 ExternalDomainXopsPage

Description

Contains a paginated list of ExternalDomainXop records (domains tracked across pentests).

Fields

`page_info` - `PageInfo`
Pagination metadata for the current page.
`external_domain_xops` - `[ExternalDomainXop!]!`
The list of ExternalDomainXop records for the current page.

Request Response

```
fragment ExternalDomainXopsPageFragment on ExternalDomainXopsPage {
  page_info {
    page_num
    page_size
  }
  external_domain_xops {
    uuid
    op_series_uuid
    xop_id
    last_op_id
    current_op_id
    is_authorized
    is_dynamic_ip
    third_party_aliases
    pentestable_rules {
      is_allowed
    }
  }
}
```

Examples may not include all available fields for a type.

8.14 Setup: Asset Groups: IPs

Manage IP addresses and ranges within asset groups for external pentesting. Query discovered IPs, view their testing history across operations, and add or remove IPs from asset groups to control your external attack surface scope.

8.14.1 `host_tab_xops_page`

Description

Returns a paginated list of HostTabXop records (IPs) in the given `AssetGroup.op_series_uuid`. This includes all discovered IPs, whether or not they fall within the configured scope. For only in-scope IPs, use in_scope_host_tab_xops_page. Supports filtering, sorting, and pagination via `page_input`.

Response

Returns a `HostTabXopsPage!`

Arguments

`op_series_uuid` - `String!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query HostTabXopsPage($op_series_uuid: String!, $page_input: PageInput) {
  host_tab_xops_page(op_series_uuid: $op_series_uuid, page_input: $page_input) {
    host_tab_xops {
      ...HostTabXopPageFragment
    }
  }
}
```

```
fragment HostTabXopPageFragment on HostTabXop {
  uuid
  op_series_uuid
  xop_id
  ip
  last_op_id
  current_op_id
  is_authorized
  third_party_aliases
  third_party_certificate_subject_cns
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.2 `host_tab_xops_count`

Description

Returns the total number of HostTabXop records (IPs) in the given `AssetGroup.op_series_uuid`. This includes all discovered IPs, whether or not they fall within the configured scope. Supports filtering via `page_input`.

Response

Returns an `Int!`

Arguments

`op_series_uuid` - `String!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query HostTabXopsCount($op_series_uuid: String!, $page_input: PageInput) {
  host_tab_xops_count(op_series_uuid: $op_series_uuid, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.3 `host_tab_xop`

Description

Returns the HostTabXop with the given `uuid`. The UUID is in the format `{op_series_uuid}/{ip_address}`.

Response

Returns a `HostTabXop`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query HostTabXop($uuid: String!) {
  host_tab_xop(uuid: $uuid) {
    ...HostTabXopFragment
  }
}

fragment HostTabXopFragment on HostTabXop {
  uuid
  op_series_uuid
  xop_id
  ip
  last_op_id
  current_op_id
  is_authorized
  third_party_aliases
  third_party_certificate_subject_cns
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.4 `in_scope_host_tab_xops_page`

Description

Returns a paginated list of HostTabXop records (unique IPs) that fall within the configured scope of the AssetGroup associated with the given `op_series_uuid`.

An External Asset Discovery may find many domains that resolve to IPs outside the AssetGroup's configured scope. Those out-of-scope IPs are excluded from this result set but are included in host_tab_xops_page.

Response

Returns a `HostTabXopsPage!`

Arguments

`op_series_uuid` - `String!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query InScopeHostTabXopsPage($op_series_uuid: String!, $page_input: PageInput) {
  in_scope_host_tab_xops_page(op_series_uuid: $op_series_uuid, page_input: $page_input) {
    host_tab_xops {
      ...InScopeHostTabXopFragment
    }
  }
}

fragment InScopeHostTabXopFragment on HostTabXop {
  uuid
  op_series_uuid
  xop_id
  ip
  last_op_id
  current_op_id
  is_authorized
  third_party_aliases
  third_party_certificate_subject_cns
  excluded_ip_from_last_pentest {
    op_id
    ip
    reason
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.5 `in_scope_host_tab_xops_count`

Description

Returns the total number of HostTabXop records (unique IPs) that fall within the configured scope of the AssetGroup associated with the given `op_series_uuid`.

An External Asset Discovery may find many domains that resolve to IPs outside the AssetGroup's configured scope. Those out-of-scope IPs are excluded from this count but are included in host_tab_xops_count.

Response

Returns an `Int!`

Arguments

`op_series_uuid` - `String!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query InScopeHostTabXopsCount($op_series_uuid: String!, $page_input: PageInput) {
  in_scope_host_tab_xops_count(op_series_uuid: $op_series_uuid, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.6 `in_scope_host_tab_xops_csv_presigned_url`

Description

Returns a temporary presigned URL for downloading a CSV export of in-scope HostTabXop records (unique IPs) in the AssetGroup associated with the given `op_series_uuid`.

The CSV format is documented under HostXopCSV.

Response

Returns a `String`

Arguments

`op_series_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query InScopeHostTabXopsCsvPresignedUrl($op_series_uuid: String!) {
  in_scope_host_tab_xops_csv_presigned_url(op_series_uuid: $op_series_uuid)
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.7 HostTabXop

Description

A HostTabXop represents a unique IP address tracked across a series of Ops within an AssetGroup. Each HostTabXop is uniquely identified by its `xop_id`, which is the IP address itself.

As new pentests run within the same AssetGroup, the HostTabXop record is updated with data from the most recent Op while retaining historical tracking information.

Fields

`uuid` - `String!`
Unique identifier for this HostTabXop, in the format `{op_series_uuid}/{xop_id}`.
`op_series_uuid` - `String!`
The ID of the OpSeries (and corresponding AssetGroup) that this IP is tracked under.
`xop_id` - `String!`
The durable cross-op identity for this asset. Always derived from the data itself rather than an arbitrary UUID. For HostTabXop, the `xop_id` is the IP address (e.g., `192.168.1.1`).
`ip` - `String!`
The IP address of this host. For AssetGroup IPs, this value uniquely identifies the HostTabXop within the OpSeries.
`last_op_id` - `String`
The ID of the most recent Op in which this IP address was discovered.
`current_op_id` - `String`
The ID of the most recently run Op in the OpSeries.
`is_authorized` - `Boolean`
Indicates whether this IP has been authorized for external pentesting by the user. IPs must be authorized before they can be included in an External Attack pentest. Use authorize_ips or deauthorize_ips to change this value.
`excluded_ip_from_last_pentest` - `ExcludedIP`
The ExcludedIP record from the most recent External Attack pentest against this AssetGroup. This field is non-null when the IP was authorized for pentesting but was subsequently excluded from scope during the pentest, typically due to drift or unreachability.
`pentestable_rules` - `PentestableRules`
The PentestableRules evaluation result for this IP. These rules determine whether the user is permitted to authorize this IP for external pentesting. IPs may be blocked if they are determined to belong to third-party providers that do not permit pentesting.
`third_party_aliases` - `[String]`
Full list of third-party aliases for domains associated with this IP. These are subdomains that are NOT covered by a top-level domain (TLD) configured in the AssetGroup. Aliases include CNAME chain entries and DNS reverse-lookup hostnames.
`third_party_certificate_subject_cns` - `[String]`
Third-party certificate subject Common Names (CNs). Includes all certificate CNs that are NOT covered by a top-level domain (TLD) configured in the AssetGroup.

Request Response

```
fragment HostTabXopFragment on HostTabXop {
  uuid
  op_series_uuid
  xop_id
  ip
  last_op_id
  current_op_id
  is_authorized
  third_party_aliases
  third_party_certificate_subject_cns
  excluded_ip_from_last_pentest {
    op_id
    ip
    reason
  }
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
```

```
    authz_warning_label
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.8 ExcludedIP

Description

Represents an IP address that was authorized for pentesting but was excluded from the pentest at runtime. IPs are typically excluded due to drift (the IP is no longer associated with the domain) or unreachability. See ExcludedReason for the possible reasons.

Fields

`op_id` - `String!`
The ID of the Op (pentest) during which the IP was excluded.
`ip` - `String!`
The IP address that was excluded from the pentest.
`reason` - `ExcludedReason!`
The reason the IP was excluded. See ExcludedReason for possible values.

Request Response

```
fragment ExcludedIPFragment on ExcludedIP {
  op_id
  ip
  reason
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: IPs

8.14.9 HostTabXopsPage

Description

Contains a paginated list of HostTabXop records (IP addresses tracked across pentests).

Fields

`page_info` - `PageInfo`
Pagination metadata for the current page.
`host_tab_xops` - `[HostTabXop!]!`
The list of HostTabXop records for the current page.

Request Response

```
fragment HostTabXopsPageFragment on HostTabXopsPage {
  page_info {
    page_num
    page_size
  }
  host_tab_xops {
    uuid
    op_series_uuid
    xop_id
    ip
    last_op_id
    current_op_id
    is_authorized
    third_party_aliases
    pentestable_rules {
      is_allowed
      action_tooltip
    }
  }
}
```

Examples may not include all available fields for a type.

8.15 Setup: Asset Groups: Authorization

Authorize or deauthorize domains and IP addresses for external pentesting.

8.15.1 `authorize_domains`

Description

Authorizes the specified ExternalDomainXop domains for External Attack pentesting. Each domain is evaluated against PentestableRules; domains that pass the rules are authorized, while domains owned by providers that do not permit pentesting are returned in the blocked_pentestable_entities list.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

`external_domain_xop_uuids` - `[String!]!`
List of ExternalDomainXop UUIDs to authorize.

Request Response Python
cURL h3-cli

```
mutation AuthorizeDomains($external_domain_xop_uuids: [String!]!) {
  authorize_domains(external_domain_xop_uuids: $external_domain_xop_uuids) {
    pentestable_entities {
      uuid
      entity_type
      is_authorized
    }
    blocked_pentestable_entities {
      pentestable_entity {
        uuid
```

```
        entity_type
        is_authorized
      }
      pentestable_rules {
        is_allowed
        action_tooltip
        authz_warning
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.2 `deauthorize_domains`

Description

Deauthorizes the specified ExternalDomainXop domains, removing them from the scope of future External Attack pentests. Deauthorization requests are always allowed (they are not subject to PentestableRules blocking).

Response

Returns a `PentestableEntitiesOutput!`

Arguments

`external_domain_xop_uuids` - `[String!]!`
List of ExternalDomainXop UUIDs to deauthorize.

Request | Response | Python
cURL | h3-cli

```
mutation DeauthorizeDomains($external_domain_xop_uuids: [String!]!) {
  deauthorize_domains(external_domain_xop_uuids: $external_domain_xop_uuids) {
    pentestable_entities {
      uuid
      entity_type
      is_authorized
    }
    blocked_pentestable_entities {
      pentestable_entity {
        uuid
        entity_type
        is_authorized
      }
      pentestable_rules {
        is_allowed
        action_tooltip
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.3 `bulk_authorize_domains`

Description

Bulk authorizes all eligible ExternalDomainXop domains in the given OpSeries for External Attack pentesting. Only domains whose PentestableRules evaluate to `is_allowed = true` will be authorized. Optionally filter to only domains within the configured scope or matching specific criteria.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

`op_series_uuid` - `String!`
The UUID of the OpSeries (and corresponding AssetGroup).
`configured_domains_only` - `Boolean`
When `true`, only domains configured in the AssetGroup's scope will be authorized. Defaults to `false`.
`page_input` - `PageInput`
Optional filters to narrow which domains are authorized. Only domains matching the filters will be authorized.

Request	Response	Python
cURL	h3-cli	

```
mutation BulkAuthorizeDomains(
  $op_series_uuid: String!,
  $configured_domains_only: Boolean,
  $page_input: PageInput
) {
  bulk_authorize_domains(
    op_series_uuid: $op_series_uuid,
    configured_domains_only: $configured_domains_only,
    page_input: $page_input
  ) {
    pentestable_entities_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.4 `bulk_deauthorize_domains`

Description

Bulk deauthorizes all ExternalDomainXop domains in the given OpSeries, removing them from the scope of future External Attack pentests. Optionally use `page_input` filters to narrow which domains are deauthorized.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

`op_series_uuid` - `String!`
The UUID of the OpSeries (and corresponding AssetGroup).
`page_input` - `PageInput`
Optional filters to narrow which domains are deauthorized. Only domains matching the filters will be deauthorized.

Request	Response	Python
cURL	h3-cli	

```
mutation BulkDeauthorizeDomains(
  $op_series_uuid: String!,
  $page_input: PageInput
) {
  bulk_deauthorize_domains(
    op_series_uuid: $op_series_uuid,
    page_input: $page_input
  ) {
    pentestable_entities_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.5 `authorize_ips`

Description

Authorizes the specified HostTabXop IPs for External Attack pentesting. Each IP is evaluated against PentestableRules; IPs that pass the rules are authorized, while IPs owned by providers that do not permit pentesting are returned in the blocked_pentestable_entities list.

Response

Returns a `PentestableEntitiesOutput!`

Arguments

`host_tab_xop_uuids` - `[String!]!`
List of HostTabXop UUIDs to authorize.

Request | Response | Python
cURL | h3-cli

```
mutation AuthorizeIps($host_tab_xop_uuids: [String!]!) {
  authorize_ips(host_tab_xop_uuids: $host_tab_xop_uuids) {
    pentestable_entities {
      uuid
      entity_type
      is_authorized
    }
    blocked_pentestable_entities {
      pentestable_entity {
        uuid
        entity_type
        is_authorized
      }
      pentestable_rules {
        is_allowed
        action_tooltip
        authz_warning
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.6 `deauthorize_ips`

Description

Deauthorizes the specified HostTabXop IPs, removing them from the scope of future External Attack pentests. Deauthorization requests are always allowed (they are not subject to PentestableRules blocking).

Response

Returns a `PentestableEntitiesOutput!`

Arguments

`host_tab_xop_uuids` - `[String!]!`
List of HostTabXop UUIDs to deauthorize.

Request | Response | Python
cURL | h3-cli

```
mutation DeauthorizeIps($host_tab_xop_uuids: [String!]!) {
  deauthorize_ips(host_tab_xop_uuids: $host_tab_xop_uuids) {
    pentestable_entities {
      uuid
      entity_type
      is_authorized
    }
    blocked_pentestable_entities {
      pentestable_entity {
        uuid
        entity_type
        is_authorized
      }
      pentestable_rules {
        is_allowed
        action_tooltip
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.7 `bulk_authorize_ips`

Description

Bulk authorizes all eligible in-scope HostTabXop IPs in the given OpSeries for External Attack pentesting. Only IPs whose PentestableRules evaluate to `is_allowed = true` and that are within the configured scope will be authorized. Optionally use `page_input` filters to narrow which IPs are authorized.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

`op_series_uuid` - `String!`
The UUID of the OpSeries (and corresponding AssetGroup).
`page_input` - `PageInput`
Optional filters to narrow which IPs are authorized. Only IPs matching the filters will be authorized.

Request | Response | Python
cURL | h3-cli

```
mutation BulkAuthorizeIps(
  $op_series_uuid: String!,
  $page_input: PageInput
) {
  bulk_authorize_ips(
    op_series_uuid: $op_series_uuid,
    page_input: $page_input
  ) {
    pentestable_entities_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.8 `bulk_deauthorize_ips`

Description

Bulk deauthorizes all HostTabXop IPs in the given OpSeries, removing them from the scope of future External Attack pentests. Optionally use `page_input` filters to narrow which IPs are deauthorized.

Response

Returns a `PentestableEntitiesBulkOutput!`

Arguments

`op_series_uuid` - `String!`
The UUID of the OpSeries (and corresponding AssetGroup).
`page_input` - `PageInput`
Optional filters to narrow which IPs are deauthorized. Only IPs matching the filters will be deauthorized.

Request | Response | Python
cURL | h3-cli

```
mutation BulkDeauthorizeIps(
  $op_series_uuid: String!,
  $page_input: PageInput
) {
  bulk_deauthorize_ips(
    op_series_uuid: $op_series_uuid,
    page_input: $page_input
  ) {
    pentestable_entities_count
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.9 PentestableEntity

Description

A PentestableEntity links a domain (ExternalDomainXop) or IP (HostTabXop) to its current authorization status for external pentesting.

Fields

`uuid` - `String!`
The UUID of the domain or IP asset.

`is_authorized` - `Boolean!`
Whether this asset is currently authorized for external pentesting.

Request Response

```
fragment PentestableEntityFragment on PentestableEntity {
  uuid
  entity_type
  is_authorized
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.10 PentestableRules

Description

The result of evaluating authorization rules for an ExternalDomainXop or HostTabXop. These rules determine whether a domain or IP asset is permitted to be authorized for External Attack pentesting. The evaluation considers factors such as third-party ownership and provider policies regarding pentesting.

Fields

`action_tooltip` - `String`
Tooltip text explaining why an asset was blocked from authorization. When an asset is blocked, this field provides a human-readable explanation of the reason.

`authz_warning` - `String`
Warning text displayed when the user attempts to authorize this asset. Advises the user about additional vetting of asset ownership that may be required before authorization.

`authz_warning_label` - `String`
Short label for the `authz_warning` message (e.g., the category of warning).

`is_allowed` - `Boolean!`
The final result of the authorization rules evaluation. When `true`, the asset is permitted to be authorized for external pentesting. When `false`, the asset is blocked.

Request Response

```
fragment PentestableRulesFragment on PentestableRules {
  is_allowed
  action_tooltip
  authz_warning
  authz_warning_label
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.11 BlockedPentestableEntity

Description

A BlockedPentestableEntity pairs a domain or IP asset (PentestableEntity) with the PentestableRules that caused it to be blocked from authorization. Assets are blocked when the rules determine they are owned by providers that do not permit pentesting.

Fields

`pentestable_entity` - `PentestableEntity`
The domain or IP asset that was blocked from authorization.
`pentestable_rules` - `PentestableRules`
The PentestableRules evaluation that determined why this asset was blocked from external pentesting.

Request Response

```
fragment BlockedPentestableEntityFragment on BlockedPentestableEntity {
  pentestable_entity {
    uuid
    entity_type
    is_authorized
  }
  pentestable_rules {
    is_allowed
    action_tooltip
    authz_warning
    authz_warning_label
  }
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.12 PentestableEntitiesBulkOutput

Description

Output type for bulk authorization mutations such as bulk_authorize_domains, bulk_deauthorize_domains, bulk_authorize_ips, and bulk_deauthorize_ips.

Fields

`pentestable_entities_count` - `Int!`
The number of domain or IP assets that were updated by this bulk request.

Request Response

```
fragment PentestableEntitiesBulkOutputFragment on PentestableEntitiesBulkOutput {
  pentestable_entities_count
}
```

Examples may not include all available fields for a type.

Setup: Asset Groups: Authorization

8.15.13 PentestableEntitiesOutput

Description

Output type for mutations that authorize or deauthorize individual domain and IP assets for external pentesting, such as authorize_domains, deauthorize_domains, authorize_ips, and deauthorize_ips.

Fields

`pentestable_entities` - `[PentestableEntity]`
List of domain and IP assets, along with their updated authorization status, that were successfully modified by this request.
`blocked_pentestable_entities` - `[BlockedPentestableEntity]`
List of domain and IP assets that were blocked from authorization. Assets are blocked when the PentestableRules determine they are owned by providers that do not permit pentesting. See BlockedPentestableEntity.pentestable_rules for details on why each asset was blocked.

Request Response

```
fragment PentestableEntitiesOutputFragment on PentestableEntitiesOutput {
  pentestable_entities {
    uuid
    entity_type
    is_authorized
  }
  blocked_pentestable_entities {
    pentestable_entity {
      uuid
      entity_type
      is_authorized
    }
    pentestable_rules {
      is_allowed
      action_tooltip
      authz_warning
      authz_warning_label
    }
  }
}
```

Examples may not include all available fields for a type.

8.16 Pentesting

Access comprehensive pentest results and operational data. Query pentests and their findings including attack paths (impacts), weaknesses (vulnerabilities), compromised credentials, discovered hosts and services, proof artifacts, EDR detections, and detailed action logs. This is the core section for retrieving and analyzing the results of your autonomous pentests.

8.17 Pentesting: Ops

Manage operations (ops) which represent cybersecurity activities conducted by NodeZero. Query ops, create new pentests and other operation types, control execution (pause, resume, cancel), and track operational status and configuration.

8.17.1 `ops_page`

Description

Returns a paginated list of ops for the current client account.

Includes all op types and archived ops. Use `page_input` to paginate, sort, and filter results.

Response

Returns an `OpsPage!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query OpsPage($page_input: PageInput) {
  ops_page(page_input: $page_input) {
    ops {
      ...OpFragment
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.2 `ops_count`

Description

Returns the total number of ops for the current client account.

Includes all op types, including non-pentest types such as `ExternalAssetDiscovery`. Includes archived ops.

Use `page_input` filters to narrow results (e.g. by `op_type` or date range).

Response

Returns an `Int!`

Arguments

`client_account_uuid` - `String`
Optional subclient UUID, for MSSP accounts that manage multiple subclients. Defaults to the current client account.
`page_input` - `PageInput`

Request	Response	Python
cURL	h3-cli	

```
query OpsCount($client_account_uuid: String) {
  ops_count(client_account_uuid: $client_account_uuid)
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.3 `op`

Description

Returns a single Op by its `op_id`.

The returned Op includes summary counts, timestamps, configuration, and links to related entities such as weaknesses, attack paths, and assets.

Response

Returns an `Op`

Arguments

`op_id` - `String!`
The unique identifier of the op to retrieve.

Request	Response	Python
cURL	h3-cli	

```
query Op($op_id: String!) {
  op(op_id: $op_id) {
    ...OpFragment
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.4 `create_op`

Description

Creates a new Op for immediate execution. An op is an autonomous cybersecurity assessment conducted by NodeZero.

See `ScheduleOpFormInput` for the full set of configuration options available when creating an op, and `OpType` for the list of supported operation types (e.g. internal pentest, external pentest, AD password audit) and their configuration requirements.

You can optionally base the op on an `OpTemplate` by passing `op_template_uuid`. Op templates store reusable configurations, making it easy to run similar ops repeatedly. See `Mutation.create_op_template` for creating op templates.

To run ops on an automated recurring schedule (e.g. weekly or monthly), see `Schedule` and `Mutation.create_scheduled_action`.

Response

Returns a `CreateOpOutput!`

Arguments

`op_template_uuid` - `String`

The `OpTemplate` UUID to use as the base configuration for this op.

If not specified, the default op template for the op type in `schedule_op_form.op_type` is used. If `schedule_op_form.op_type` is also not defined, the default op type is `NodeZero` (internal pentest).

`op_name` - `String`
Name of the pentest, defaults to the name in pentest template.
`schedule_op_form` - `ScheduleOpFormInput`
Optional configuration overrides. Any parameters defined here will override the corresponding parameters in the OpTemplate's `schedule_op_form`. See `ScheduleOpFormInput` for available options.
`agent_name` - `String`
Assign an h3-cli agent to automatically launch NodeZero for this pentest.
`runner_uuid` - `String`
Assign an h3-cli runner to automatically launch NodeZero for this pentest.
`schedule_name` - `String`
The automated schedule to which this pentest is assigned. This is used internally by automated scheduling.
`run_immediately` - `Boolean`
If an op is assigned to an active runner, pause the current action on the runner and move op to top of queue.

Request | Response | Python
cURL | h3-cli

```
mutation CreateOp($schedule_op_form: ScheduleOpFormInput) {
  create_op(schedule_op_form: $schedule_op_form) {
    op {
      ...OpFragment
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  nodezero_script_url
  nodezero_ip
  runner_uuid
```

```
  runner_name
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.5 `pause_op`

Description

Pauses a running Op.

The op must be in a pausable state (e.g. `running`). Returns an error if the op is already paused or in a non-pausable state.

Use `Mutation.resume_op` to resume a paused op.

Response

Returns an `OpTab!`

Arguments

`op_id` - `String!`
The unique identifier of the op to pause.

Request | Response | Python
cURL | h3-cli

```
mutation PauseOp($op_id: String!) {
  pause_op(op_id: $op_id) {
    op_id
    op_name
    op_state
    portal_op_state
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.6 `resume_op`

Description

Resumes a paused Op.

The op must be in a resumable state (e.g. `paused`, `user_paused`, `start_paused`). Returns an error if the op is already running or in a non-resumable state.

Use `Mutation.pause_op` to pause a running op.

Response

Returns an `OpTab!`

Arguments

`op_id` - `String!`
The unique identifier of the op to resume.

Request | Response | Python
cURL | h3-cli

```
mutation ResumeOp($op_id: String!) {
  resume_op(op_id: $op_id) {
    op_id
    op_name
    op_state
    portal_op_state
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.7 `cancel_op`

Description

Cancels an Op, ending it early.

The op must be in a cancelable state (e.g. `running`, `paused`, `ready`). Returns an error if the op is already in a terminal state such as `done` or `canceled`.

Partial results from the op may still be available in Portal after cancellation.

Response

Returns an `OpTab!`

Arguments

`op_id` - `String!`
The unique identifier of the op to cancel.

Request | Response | Python
cURL | h3-cli

```
mutation CancelOp($op_id: String!) {
  cancel_op(op_id: $op_id) {
    op_id
    op_name
    op_state
    portal_op_state
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.8 `update_op`

Description

Updates an existing Op. Supports renaming, archiving, and deleting ops.

See UpdateOpInput for the set of fields that can be updated.

Response

Returns an `UpdateOpOutput`

Arguments

`op_id` - `String!`
The unique identifier of the op to update.
`update_op_input` - `UpdateOpInput!`
The fields to update. See UpdateOpInput.

Request	Response	Python
cURL	h3-cli	

```
mutation UpdateOp($op_id: String!, $update_op_input: UpdateOpInput!) {
  update_op(op_id: $op_id, update_op_input: $update_op_input) {
    op {
      ...OpFragment
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.9 `rerun_op`

Description

Creates a new Op using the same configuration as a previously run op.

This is a convenient way to re-run an op with the same (or similar) settings. You can optionally override specific configuration parameters via `schedule_op_form`.

The new op is created for immediate execution.

Response

Returns a `ScheduleOpOutput!`

Arguments

`op_id` - `String!`
The `op_id` of the previously run op whose configuration will be used as the base.
`op_name` - `String`
Optionally override the name of the new op.
`schedule_op_form` - `ScheduleOpFormInput`
Optional parameters that, if set, override the corresponding parameters from the base op's configuration. See ScheduleOpFormInput.
`agent_name` - `String`
Optionally override the h3-cli agent in the base configuration.
`runner_uuid` - `String`
Optionally specify a NodeZero Runner to launch this op, overriding the base configuration.

Request | Response | Python
cURL | h3-cli

```
mutation RerunOp($op_id: String!, $op_name: String, $runner_uuid: String) {
  rerun_op(op_id: $op_id, op_name: $op_name, runner_uuid: $runner_uuid) {
    op {
      ...OpFragment
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  nodezero_script_url
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.10 `op_reports`

Description

Returns the list of OpReport records available for the given op.

Each entry identifies a report by ReportType and includes its `uuid`.

Response

Returns `[OpReport]`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query OpReports($input: OpInput!) {
  op_reports(input: $input) {
    uuid
    op_id
    report_type
    presigned_url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.11 `op_report_of_type`

Description

Returns a single OpReport for the given op matching the requested ReportType, or `null` if no report of that type exists for the op.

Response

Returns an `OpReport`

Arguments

`input` - `OpInput!`

`report_type` - `ReportType!`

Request | Response | Python

cURL | h3-cli

```
query OpReportOfType($input: OpInput!, $report_type: ReportType!) {
  op_report_of_type(input: $input, report_type: $report_type) {
    uuid
    op_id
    report_type
    presigned_url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.12 `op_report_url`

Description

Returns a short-lived presigned URL for downloading the report file identified by `op_report_uuid`. The `op_report_uuid` is the OpReport.uuid returned by op_reports or op_report_of_type.

The URL expires within a short period of time, so fetch a fresh URL each time you need to download the report.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

`op_report_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query OpReportUrl($input: OpInput!, $op_report_uuid: String!) {
  op_report_url(input: $input, op_report_uuid: $op_report_uuid)
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.13 `sample_ops_page`

Description

Get a list of ops for the Horizon3 sample client account. Includes all op types. Includes archived ops.

Response

Returns an `OpsPage!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query SampleOpsPage($page_input: PageInput) {
  sample_ops_page(page_input: $page_input) {
    ops {
      ...OpFragment
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.14 `sample_op`

Description

Get pentest data for a Horizon3 sample client account Op.

Response

Returns an `Op`

Arguments

`op_id` - `String!`
ID of pentest.

Request | Response | Python
cURL | h3-cli

```
query SampleOp($op_id: String!) {
  sample_op(op_id: $op_id) {
    ...OpFragment
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.15 Op

Description

An Op represents an autonomous cybersecurity operation conducted by NodeZero.

The most common type of Op is an internal Pentest, but NodeZero supports several other operation types including external pentesting, asset discovery, Active Directory password audits, and more.

See OpType for the full list of supported operation types and their configuration requirements.

Use `Query.ops_page` to list ops, `Query.op` to retrieve a single op, and `Mutation.create_op` to create a new op.

Fields

`op_id` - `String`
Unique identifier for the op.
`op_type` - `OpType`
The type of operation. See OpType for descriptions of the supported operation types.
`op_state` - `String!`

The current state of the op. Possible values include:

- `scheduled` - The op has been created and is queued to start.
- `provisioning` - Resources are being provisioned.
- `ready` - NodeZero is ready to launch.
- `running` - The op is actively scanning the environment.
- `complete` - The scan has finished; results are being processed.
- `post-processing` - Op results are undergoing post-processing.
- `done` - The op is complete and results are available.
- `cancelling` - The op is in the process of being canceled.
- `canceled` - The op was canceled by a user.
- `paused` - The op is paused.
- `error` - The op encountered an error.

See PortalOpState for an alternative representation of op state used by the Portal UI.

`portal_op_state` - `PortalOpState!`
The current state of the op, as displayed in the Portal UI. See PortalOpState for the list of possible states.
`op_name` - `String!`
Name of the op. The name is used for display purposes in Portal and in generated reports.
`op_param_min_scope` - `[String]`

The minimum scope for this op, as a list of IPs and CIDR ranges.

When set, the Auto-Expand Scope feature is enabled. NodeZero will scan these specific hosts plus any additional hosts it discovers organically during the op.

See `ScheduleOpForm.op_param_min_scope` for full details on how minimum and maximum scope interact.

`op_param_max_scope` - `[String]`

The maximum scope for this op, as a list of IPs and CIDR ranges.

When set, NodeZero will only scan hosts and subnets that fall within this scope. The Auto-Expand Scope feature is disabled.

See `ScheduleOpForm.op_param_max_scope` for full details on how minimum and maximum scope interact.

`op_param_blacklist` - `String`
IPs and CIDR ranges excluded from the op, as a comma-separated list.
`scheduled_timestamp` - `Float!`
Timestamp when the op was created, in epoch seconds.
`scheduled_timestamp_iso` - `String!`
Timestamp when the op was created, in ISO format (UTC). use `scheduled_at` instead
`scheduled_at` - `Datetime!`
Timestamp when the op was created. Alias of `created_at`.
`scheduled_at_date` - `Date!`
Date when the op was created (date only, no time component).
`created_at` - `Datetime!`
Timestamp when the op was created.
`completed_timestamp_iso` - `String`
Timestamp when the op completed its scan, in ISO format (UTC). use `completed_at` instead
`completed_at` - `Datetime`
Timestamp when the op completed its scan. This does not include post-processing time. Use `Op.etl_completed_at` to get the timestamp that includes post-processing.
`canceled_at` - `Datetime`
Timestamp when the op was canceled. Null if the op was not canceled.
`launched_timestamp_iso` - `String`
Timestamp when the op launched, in ISO format (UTC). use `launched_at` instead
`launched_at` - `Datetime`
Timestamp when the op launched. This is when NodeZero begins actively scanning the target environment.
`etl_completed_at` - `Datetime`
Timestamp when the op completed post-processing. This is when the full op results become available in Portal.
`duration_s` - `Int`
Op duration in seconds, calculated as the difference between `launched_at` and `completed_at`.
`services_count` - `Int!`
Number of network services discovered during the op.
`credentials_count` - `Int!`
Total number of credentials found, including both discovered and confirmed credentials.
`confirmed_credentials_count` - `Int`
Number of credentials that were confirmed (proven) during the op.
`hosts_count` - `Int`
Number of in-scope hosts (IPs) scanned during the op.
`out_of_scope_hosts_count` - `Int`
Number of out-of-scope hosts (IPs) discovered during the op.
`external_domains_count` - `Int`
Number of external domains discovered during the op.
`users_count` - `Int`
Number of users discovered during the op.
`websites_count` - `Int`
Number of websites discovered during the op.
`data_stores_count` - `Int`
Number of data stores discovered during the op (e.g. file shares, databases, S3 buckets, Git repos). Excludes websites.
`data_resources_count` - `Long`
Total count of data resources (files, database records, etc.) found across all discovered data stores.
`weaknesses_count` - `Int`
Number of individual weakness instances found during the op. A single weakness type (e.g. weak password) may have multiple instances across different assets.
`weaknesses_page` - `WeaknessesPage!`
Paginated list of individual weakness instances found during this op. See WeaknessesPage.

Arguments

`page_input` - `PageInput`

`proven_weaknesses_count` - `Int`
Number of weakness instances that were confirmed with proof. Proven weaknesses are those NodeZero was able to successfully exploit.
`in_scope_hosts_count` - `Int`
Number of in-scope hosts discovered during the op. Alias of `hosts_count`.
`feature_flags` - `[FeatureFlag]`
Advanced configuration settings for this op. See FeatureFlag for details.
`nodezero_script_url` - `String`
URL of the script that downloads and launches NodeZero on a Docker Host.
`nodezero_ip` - `String`
IP address assigned to NodeZero for this op.
`duration_hms` - `String`
Op duration in HH:MM:SS format, calculated as the difference between `launched_at` and `completed_at`.
`duration_humanize` - `String`
Op duration in human-readable format, calculated as the difference between `launched_at` and `completed_at`.
`op_template_uuid` - `String`
ID of the OpTemplate used to configure this op.
`op_template_name` - `String`
Name of the OpTemplate used to configure this op.
`impact_paths_count` - `Int`
Number of attack paths discovered during the op. An attack path is a sequence of steps NodeZero took to reach a business impact. Alias of `attack_paths_count`.
`attack_paths_count` - `Int`
Number of attack paths discovered during the op. An attack path is a sequence of steps NodeZero took to reach a business impact. Alias of `impact_paths_count`.
`attack_paths_page` - `AttackPathsPage!`
Paginated list of attack paths discovered during this op. See AttackPathsPage.

Arguments

`page_input` - `PageInput`
`phished_impact_paths_count` - `Int!`
Number of attack paths that originated from phished credentials. Alias of `phished_attack_paths_count`.
`phished_attack_paths_count` - `Int!`
Number of attack paths that originated from phished credentials. Alias of `phished_impact_paths_count`.
`critical_impact_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_attack_paths_count`.
`critical_attack_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_impact_paths_count`.
`runner_uuid` - `String`
ID of the NodeZero Runner assigned to launch this op.
`runner_name` - `String`
Name of the NodeZero Runner assigned to launch this op.
`schedule_uuid` - `String`
ID of the Schedule associated with this op, if the op was created by an automated schedule.
`schedule_name` - `String`
Name of the Schedule associated with this op, if the op was created by an automated schedule.
`schedule` - `Schedule`
The Schedule associated with this op, if the op was created by an automated schedule.
`schedule_op_form` - `ScheduleOpForm!`
The full configuration that was used to run this op. See ScheduleOpForm for details.
`asset_group_uuid` - `String`
The ID of the AssetGroup associated with this op. Set for `ExternalAssetDiscovery` and `ExternalAttack` op types.
`asset_group` - `AssetGroup`
The AssetGroup associated with this op. Set for `ExternalAssetDiscovery` and `ExternalAttack` op types.

`one_click_verify_data` - `OneClickVerifyData`

For 1-click verify ops, contains the weakness and vulnerability data that the op was created to verify. See OneClickVerifyData.

`portal_url` - `String`

The Portal URL for viewing this op's results.

Request Response

```
fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  op_param_blacklist
  scheduled_at
  created_at
  launched_at
  completed_at
  etl_completed_at
  canceled_at
  duration_s
  duration_hms
  duration_humanize
  hosts_count
  services_count
  credentials_count
  users_count
  weaknesses_count
  attack_paths_count
  nodezero_script_url
  nodezero_ip
  runner_uuid
  runner_name
  schedule_uuid
  schedule_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.16 OpType

Description

The type of operation to run. You must specify an op type when creating an Op or an OpTemplate.

Each op type has different configuration requirements. See the individual enum values below for details on each type and how to configure it via ScheduleOpFormInput.

For more information about op types, visit the Tests & Assessments section of the Horizon3.ai documentation.

Values

`NodeZero`

An Internal Pentest op, a simulated attack on your organization's internal network designed to identify vulnerabilities, misconfigurations, and exploitable attack paths.

NodeZero autonomously enumerates devices in scope, then chains together context, exploitable vulnerabilities, misconfigurations, and captured data to identify attack paths ranked by impact, with proof of exploitability.

Minimum configuration to get started:

1. A network scope to pentest, specified in CIDR notation via `op_param_min_scope` or `op_param_max_scope`
2. A Docker Host within your network, on which to launch NodeZero

Scope options include:

- **Intelligent Scope**: NodeZero automatically expands from the Docker Host's subnet to discover nearby assets. Useful for scenario-based testing.
- **Full private IP space** (RFC 1918): Covers `10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16` for comprehensive coverage.
- **Custom IPs or subnets**: Specify exact CIDR ranges. Can be combined with auto-expand scope (`op_param_min_scope`) to let NodeZero discover additional assets beyond the initial scope.

Additional configuration options include attack configurations, credential injection, AWS account integration, OSINT terms, and tripwire deployment.

Visit Run an Internal Pentest in the Horizon3.ai documentation for more details on configuration and capabilities.

`ExternalAssetDiscovery`

An External Asset Discovery op, for discovering public-facing domains and IPs in an `AssetGroup`.

This op is usually run in advance of an `ExternalAttack` pentest. It performs initial discovery of your external attack surface, identifying domains, subdomains, and IP addresses associated with your organization. You may then select one or more of the discovered domains and IPs to authorize for external pentesting.

To run an `ExternalAssetDiscovery` op for an `AssetGroup`, use `Mutation.create_op` and pass the `AssetGroup.uuid` via `schedule_op_form.asset_group_uuid`. Alternatively, you can pass the `AssetGroup.op_template_uuid` via the `op_template_uuid` parameter.

This is an external operation and therefore NodeZero is launched from the H3 cloud. It does not require a Docker Host within your network.

Visit Run Asset Discovery in the Horizon3.ai documentation for more details on configuration and capabilities.

`ExternalAttack`

An External Pentest op, for pentesting public-facing domains and IPs in an `AssetGroup`.

In many cases users will run an `ExternalAssetDiscovery` op first, to discover their external attack surface, then authorize NodeZero to pentest one or more of the discovered domains and IPs. This workflow involves several steps:

1. create an `AssetGroup`
2. run an `ExternalAssetDiscovery` op, to discover public domains and IPs
3. authorize discovered domains and IPs for external pentesting
4. run an `ExternalAttack` op against the authorized assets

However, this workflow is no longer strictly required. As of April 2025, you may now configure and run an external pentest directly, without first running an `ExternalAssetDiscovery` op. NodeZero will auto-discover your attack surface during the pentest, and it will auto-authorize and pentest any discovered domain that doesn't have a CNAME record from a third-party service.

After the external pentest completes, all discovered domains and IPs are included in the results in Portal. You may then authorize additional domains and IPs for further pentesting.

To run an `ExternalAttack` op for an `AssetGroup` use `Mutation.create_op` and pass the `AssetGroup.uuid` via `schedule_op_form.asset_group_uuid`.

Alternatively you can create an `OpTemplate` for the `ExternalAttack` op and set the `AssetGroup.uuid` in `OpTemplate.schedule_op_form.asset_group_uuid`.

This is an external operation and therefore NodeZero is launched from the H3 cloud. It does not require a Docker Host within your network.

Visit External Pentest in the Horizon3.ai documentation for more details on configuration and capabilities.

`NetworkEnumeration`

A Network Enumeration op (also known as a Segmentation Test), for scanning and enumerating assets on an internal network.

NodeZero discovers IPs, ports, services, and applications within the target scope, but does not perform any attacks or exploitation. This is a discovery-only operation.

Common use cases include:

- **Initial reconnaissance**: Understand what exists on a network before running exploit-based tests.
- **Asset inventorying**: Get a snapshot of hosts, services, and applications on the network.
- **Segmentation validation**: Verify which assets are reachable from a given network segment.

Scope is configured via `op_param_min_scope` or `op_param_max_scope` using IPs or subnets in CIDR notation. The full private IP space (RFC 1918) can also be used for comprehensive coverage.

This is an internal operation and therefore requires that you launch NodeZero on a Docker Host within your network.

Visit Segmentation Test in the Horizon3.ai documentation for more details on configuration and capabilities.

`ADPasswordAudit`

An Active Directory Password Audit op, for auditing the strength and security of user passwords in your Active Directory environment.

This is a gray-box style audit. NodeZero uses a privileged credential with DCSync permissions to extract NTLM password hashes from Active Directory domain controllers, then attempts to crack those hashes using multiple methods:

- Common weak passwords
- Passwords that resemble usernames
- Company-specific terms (company names, domain names)
- Dark web breach data associated with your organization
- Known breached password databases
- Custom weak password terms you provide via OSINT configuration

NodeZero also analyzes the population of passwords for **password reuse** -- cases where two or more users share the exact same or similar passwords.

The minimum required credential is a domain user account with DCSync privileges (Replicating Directory Changes, Replicating Directory Changes All, and Replicating Directory Changes in Filtered Set). A domain admin account may also be used.

Configure the domain controller IP address and the privileged credential via ScheduleOpFormInput.

This is an internal operation and therefore requires that you launch NodeZero on a Docker Host within your network.

Visit AD Password Audit in the Horizon3.ai documentation for more details on configuration and capabilities.

`Phishing`

A Phishing Impact Test op, for measuring the real-world impact of successful phishing attacks within your organization.

This op works in conjunction with your organization's phishing simulation tools (e.g. KnowBe4, ProofPoint). NodeZero generates a JavaScript payload to embed in your phishing campaign's landing page. When a user is phished, their credentials are automatically captured and injected into NodeZero, which uses them to perform an authenticated internal pentest.

The results convey the business risk of a successful phishing attack by showing what an attacker could accomplish with the phished credentials, including lateral movement and privilege escalation.

Key configuration considerations:

- A **minimum campaign runtime** must be configured to match the duration of your phishing campaign. Campaigns of up to 28 days are supported.
- Up to **100 phished credentials** can be captured per test.
- Scope should include networks where phishing targets have access, as well as networks they should not have access to, in order to validate security controls.

This is an internal operation and therefore requires that you launch NodeZero on a Docker Host within your network.

Visit Phishing Impact Test in the Horizon3.ai documentation for more details on configuration and capabilities.

`AWSPentest`

An AWS Pentest op, a gray-box style assessment of your AWS cloud account security.

NodeZero uses a read-only IAM role to gain a comprehensive view of your AWS account and identify exploitable vulnerabilities and misconfigurations in IAM policies, security configurations, and cloud resources.

To run an AWS Pentest, you must first configure one or more `AWSConnection` objects, one for each AWS account you want to assess. Cloud connections are long-lived and reusable across multiple pentests. The connections are specified via `schedule_op_form.aws_connection_uuids`.

Two IAM policy options are available when creating a connection:

- **SecurityAudit**: Grants access to read security configuration metadata only, allowing NodeZero to analyze IAM and security configurations.
- **ReadOnlyAccess**: Provides read-only access to all AWS services, allowing NodeZero to also analyze sensitive information stored in resources such as S3 buckets and Lambda functions.

This is an external operation and therefore NodeZero is launched from the H3 cloud. It does not require a Docker Host within your network.

Visit AWS Pentest in the Horizon3.ai documentation for more details on configuration and capabilities.

`K8sPentest`

A Kubernetes Pentest op, for pentesting the security of your Kubernetes clusters.

NodeZero runs from inside your cluster to simulate the scenario where an attacker has gained a foothold -- for example, through a compromised kubeconfig file or a vulnerable application running in a pod. All Kubernetes resources (nodes, pods, services) are treated as in-scope.

Prerequisites:

- The **NodeZero Operator** must be installed in the target cluster. Supported on Kubernetes cluster versions 1.24.0 and above.
- NodeZero is launched from within the cluster using a NodeZero Runner for Kubernetes. See `Mutation.install_agent` for how to install a Runner.

Scope configuration options to consider include:

- `k8s_deployment` (should be set to `true`)
- `k8s_namespace`: the namespace for NodeZero to run in
- `k8s_service_account_name`: specify a service account to simulate a compromised pod scenario
- `k8s_node`: target a specific node

Including the CIDR ranges of your cluster and the surrounding internal network or VPC is recommended, to test whether NodeZero can reach outside the cluster to other hosts.

Visit Kubernetes Pentest in the Horizon3.ai documentation for more details on configuration and capabilities.

`AzurePentest`

An Azure Entra ID Pentest op, for assessing the security of your Azure Active Directory / Entra ID environment.

This is a gray-box style pentest that evaluates the security posture of your hybrid Azure AD environment by identifying misconfigurations and exploitable weaknesses. NodeZero requires a privileged AD credential with DCSync permissions and an Azure Entra ID user credential to enumerate the environment and identify attack paths.

NodeZero uses the Azure Device Code Flow (OAuth Device Authorization Grant) for the Entra ID credential. During configuration, provide the Entra tenant ID. After the pentest launches, you will be prompted to complete the MFA authentication flow.

Visit Azure Entra ID Pentest in the Horizon3.ai documentation for more details on configuration and capabilities.

See also `AzureInternalPentest` for an enhanced Azure assessment with additional cloud-native capabilities.

`AzureInternalPentest`

An Azure Pentest op, for running a pentest inside your Azure tenant.

You must specify an Azure tenant, subscription, subnet, and role for the NodeZero VM. The VM will be deployed into the subnet and run the test using the role's privileges from an assumed breach perspective. The VM is automatically deleted after test completion.

`AzureExternalPentest`

An Azure External Pentest op, for running a pentest against your Azure tenant from an external perspective.

You must specify an Azure tenant and subscription. NodeZero will assess your Azure environment from an external attacker's perspective.

`InsiderThreatAttack`

An Insider Threat Attack op, for simulating the fallout from an insider-turned-attacker and measuring the blast radius of compromised employee credentials.

This is essentially an internal pentest with the additional requirement of injecting the credentials of an insider-threat actor (e.g. an employee's domain credentials). NodeZero uses these credentials as a starting point and demonstrates how far an attacker could move laterally, escalate privileges, and access sensitive data.

The pentest results help you understand the degree to which that employee -- or an attacker who has compromised that employee's credentials -- can attack your network, and identify security controls that can mitigate the risk.

Credential types supported include cleartext passwords and NTLM hashes. Additional configuration options include AWS account integration, OSINT terms, tripwire deployment, and attack configurations. See ScheduleOpFormInput for details.

This is an internal operation and therefore requires that you launch NodeZero on a Docker Host within your network.

Visit Insider Threat Test in the Horizon3.ai documentation for more details on configuration and capabilities.

`TripwiresInternal`

A "Drop Tripwires" op, for deploying deceptive tokens (tripwires) on designated targets within your internal network. Tripwires detect real-world unauthorized access by triggering alerts when an attacker interacts with them.

An injected credential should be included for each target in the scope of the op, allowing NodeZero to authenticate and deploy the tripwire tokens on those targets. A maximum of 100 tripwires are deployed per operation.

This op type requires that the Tripwires feature is enabled for your account.

This is an internal operation and therefore requires that you launch NodeZero on a Docker Host within your network.

Example

```
"NodeZero"
```

Pentesting: Ops

8.17.17 PortalOpState

Description

Represents the current state of an Op as displayed in the Portal UI.

These states map to the op lifecycle: scheduling, provisioning, running, pausing/resuming, post-processing, and completion. See Op.portal_op_state for usage.

Values

`done`
The op is fully complete and results are available in Portal.
`ended`
The op was ended early by the user. Partial results may still be available in Portal.
`error`
The op encountered an error during execution.
`installation_needed`
The op is waiting for NodeZero to be launched on a Docker Host. Copy and run the launch command to proceed.
`awaiting_runner`
The op is waiting for its assigned NodeZero Runner to launch NodeZero.
`start_paused`
The op was configured to start in a paused state. Use `Mutation.resume_op` to begin the scan.
`user_paused`
The op is paused by a user. Use `Mutation.resume_op` to continue.
`paused`
The op is paused, possibly due to connectivity issues with NodeZero.
`pausing`
The op is in the process of pausing.
`preparing`
Resources are being provisioned for the op.
`preparing_start_paused`
Resources are being provisioned. Once provisioning is complete, the op will start in a paused state.
`processing`
The op scan has completed and results are being processed. Results will be available shortly.
`resuming`
The op is resuming from a paused state.
`running`
The op is actively scanning the target environment.
`scheduled`
The op is queued and will begin provisioning resources shortly.
`queued`
The op is queued and will begin provisioning resources shortly.
`unknown`
The op is in an unknown state. Contact support if this persists.

Example

```
"done"
```

Pentesting: Ops

8.17.18 CreateOpOutput

Description

The output from `Mutation.create_op`.

Fields

`op - Op!`

The newly created Op.

Request Response

```
fragment CreateOpOutputFragment on CreateOpOutput {
  op {
    op_id
    op_type
    op_name
    op_state
    portal_op_state
    op_param_max_scope
    scheduled_at
    nodezero_script_url
    runner_uuid
    runner_name
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.19 OneClickVerifyData

Description

Contains the weakness and vulnerability data associated with a 1-click verify op.

Returned on `Op.one_click_verify_data` for ops created via `Mutation.schedule_weakness_series_1cv_ops`.

Fields

`weakness_uuids` - `[String!]!`
The Weakness UUIDs from the source op that the 1-click verify op was created to re-verify.
`op_id` - `String`
The ID of the source op from which the 1-click verify op was derived.
`vuln_ids` - `[String]`
The vulnerability IDs (e.g. CVE IDs) targeted by the 1-click verify op.

Request Response

```
fragment OneClickVerifyDataFragment on OneClickVerifyData {
  op_id
  vuln_ids
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.20 OpReport

Description

A report generated for a Op, such as an executive summary or a full op report. Each op produces a fixed set of reports identified by ReportType.

Use op_reports to list every report available for an op, or op_report_of_type to fetch a single report by type.

To download the report file, use the `presigned_url` field, or pass the report's `uuid` to op_report_url. Both return a short-lived presigned download URL.

Fields

`uuid` - `String!`
Unique identifier for this report.
`report_type` - `ReportType!`
The type of report. See ReportType for the list of report types produced for each op.
`presigned_url` - `String`
Short-lived presigned URL for downloading the report file. Generated at request time and expires within a short period; fetch a fresh URL each time you need to download. Returns null if the report file has not yet been uploaded.
`op_id` - `String!`
The `op_id` of the Op this report belongs to.

Request Response

```
fragment OpReportFragment on OpReport {
  uuid
  op_id
  report_type
  presigned_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.21 OpTab

Description

Pentest data.

DEPRECATED:: use Op type or Pentest type instead.

Fields

`uuid` - `String`
ID of pentest.
`op_id` - `String`
ID of pentest. Same as `uuid`.
`op_state` - `String!`

State of pentest:

- scheduled
- provisioning
- ready
- running
- complete
- post-processing
- done
- cancelling
- canceled
- paused
- error

`portal_op_state` - `PortalOpState!`
Op State surfaced in portal.
`op_name` - `String!`
Name of pentest.
`scheduled_timestamp` - `Float!`
Timestamp of pentest scheduling, in epoch seconds.
`scheduled_at` - `Datetime!`
Timestamp of pentest scheduling
`scheduled_at_date` - `Date!`
Date of pentest scheduling
`scheduled_timestamp_iso` - `String!`
Timestamp of pentest scheduling, in ISO format (UTC).
`create_timestamp` - `Int!`
Timestamp of pentest creation, in epoch seconds.
`create_timestamp_iso` - `String!`
Timestamp of pentest creation, in ISO format (UTC).
`launched_timestamp` - `Int`
Timestamp of pentest launching, in epoch seconds.
`launched_timestamp_iso` - `String`
Timestamp of pentest launching, in ISO format (UTC).
`launched_at` - `Datetime`
Timestamp of pentest launching
`completed_at` - `Datetime`
Timestamp of pentest completion.

`completed_timestamp` - `Float`
Timestamp of pentest completion, in epoch seconds.
`completed_timestamp_iso` - `String`
Timestamp of pentest completion, in ISO format (UTC).
`canceled_at` - `Datetime`
Timestamp of pentest cancellation.
`canceled_timestamp` - `Int`
Timestamp of pentest cancellation, in epoch seconds.
`canceled_timestamp_iso` - `String`
Timestamp of pentest cancellation, in ISO format (UTC).
`duration_hms` - `String`
Pentest duration in HH:MM:SS format .
`duration_humanize` - `String`
Pentest duration in "humanized" format .
`op_type` - `OpType`
Type of pentest.
`weakness_types_count` - `Int`
Number of unique weakness IDs found.
`weaknesses_count` - `Int`
Number of weaknesses found (weakness INSTANCEs, not unique weakness IDs).
`host_tabs_count` - `Int`
Number of hosts found in scope. Same as `in_scope_endpoints_count`.
`domain_controllers_count` - `Int`
Number of domain controllers found in scope.
`credentials_count` - `Int`
Number of credentials and potential credentials found.
`proven_credentials_count` - `Int`
Number of credentials and potential credentials found with proof.
`confirmed_credentials_count` - `Int`
Number of credentials with proof. Alias of `proven_credentials_count`.
`unproven_credentials_count` - `Int`
Number of credentials without proof.
`activedir_passwords_count` - `Int`
Number of credentials found in Active Directory Password Audit(s).
`enabled_activedir_passwords_count` - `Int`
Number of enabled credentials found in Active Directory Password Audit(s).
`disabled_activedir_passwords_count` - `Int`
Number of disabled credentials found in Active Directory Password Audit(s).
`feature_flags` - `[FeatureFlag]`
Advanced configuration settings for this pentest.
`impacts_headline_count` - `Int`
Number of impacts in the pentest, which generally refers to the number of assets affected by an impact.
`impact_paths_count` - `Int`
Number of impact paths in the pentest. Alias of `attack_paths_count`.
`attack_paths_count` - `Int`
Number of attack paths in the pentest. Alias of `impact_paths_count`.
`phished_impact_paths_count` - `Int!`
Number of impact paths downstream of phished credentials in the pentest. Alias of `phished_attack_paths_count`
`phished_attack_paths_count` - `Int!`
Number of attack paths downstream of phished credentials in the pentest. Alias of `phished_impact_paths_count`.
`critical_impact_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_attack_paths_count`.
`critical_attack_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_impact_paths_count`.

`nodezero_script_url` - `String`
URL of the script that downloads and launches NodeZero.
`nodezero_ip` - `String`
The IP address where NodeZero was launched.
`etl_completed_at` - `Datetime`
Timestamp of pentest post-processing completion, in ISO format (UTC).
`start_paused` - `Boolean`
Start paused.
`minimum_run_time` - `Int`
Op minimum run-time, in minutes.
`maximum_run_time` - `Int`
Op maximum run-time, in minutes.
`paused_at` - `Datetime`
Time when the pentest was most recently paused.
`paused_by_user_account_uuid` - `String`
ID of the user that most recently paused the pentest.
`paused_by_user_account` - `UserAccount`
Data of user account that most recently paused the pentest.
`op_template_uuid` - `String`
ID of OpTemplate used for this pentest.
`op_template_name` - `String`
Name of OpTemplate used for this pentest.
`excluded_ips` - `[ExcludedIP]`
IPs that were excluded from the pentest scope. Applies to external pentests only.
`excluded_domains` - `[ExcludedDomain]`
Domains that were excluded from the pentest scope. Applies to external pentests only.
`runner_uuid` - `String`
ID of NodeZero Runner the op is assigned to.
`runner_name` - `String`
Name of NodeZero Runner the op is assigned to.
`schedule_uuid` - `String`
ID of schedule the op is associated with.
`schedule_name` - `String`
Name of schedule the op is associated with.
`schedule` - `Schedule`
The schedule the op is associated with.
`auto_injected_credential_uuids` - `[String]`
The set of credentials to be auto-injected (by a NodeZero Runner) into the op.
`tripwires_count` - `Int`
The number of Tripwires dropped during the op

Request Response

```
fragment OpTabFragment on OpTab {
  uuid
  op_id
  op_state
  portal_op_state
  op_name
  op_type
  scheduled_at
  launched_at
  completed_at
  canceled_at
  etl_completed_at
  duration_hms
  duration_humanize
  weaknesses_count
  weakness_types_count
  host_tabs_count
  credentials_count
  attack_paths_count
  impacts_headline_count
  nodezero_script_url
  nodezero_ip
  runner_uuid
  runner_name
```

```
  schedule_uuid
  schedule_name
  op_template_uuid
  op_template_name
  tripwires_count
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.22 OpTabsPage

Description

A paginated list of pentests.

DEPRECATED: Use OpsPage and related APIs instead.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`op_tabs` - `[OpTab!]!`
The list of pentests in this page.

Request Response

```
fragment OpTabsPageFragment on OpTabsPage {
  page_info {
    page_num
    page_size
  }
  op_tabs {
    op_id
    op_name
    op_state
    op_type
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.23 OpsPage

Description

A paginated list of Op objects. Returned by `Query.ops_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.

`ops` - `[Op!]!`
The list of Op objects in this page.

Request Response

```
fragment OpsPageFragment on OpsPage {
  page_info {
    page_num
    page_size
  }
  ops {
    ...OpFragment
  }
}

fragment OpFragment on Op {
  op_id
  op_type
  op_name
  op_state
  portal_op_state
  op_param_max_scope
  scheduled_at
  created_at
  launched_at
  completed_at
  duration_hms
  hosts_count
  weaknesses_count
  runner_uuid
  runner_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.24 ReportType

Description

Identifies the kind of report stored in an OpReport. Each completed Op produces a fixed set of reports; the `report_type` distinguishes between executive summaries, full op reports, fix-action guidance, and segmentation reports.

To list all reports available for an op, use op_reports. To fetch a specific report, use op_report_of_type.

Values

`EXEC_SUMMARY_REPORT_V2`
Executive summary report (PDF). High-level overview of the op results, suitable for leadership and non-technical audiences. Current production format.
`EXEC_SUMMARY_REPORT`
Legacy executive summary report format. Current production uses EXEC_SUMMARY_REPORT_V2. Older ops may still have this report available.
`PENTEST_REPORT_V2`
Full op report (PDF). Detailed technical writeup of attack paths, weaknesses, credentials, hosts, and proof artifacts discovered during the op. Current production format.
`PENTEST_REPORT`
Legacy op report format. Current production uses PENTEST_REPORT_V2. Older ops may still have this report available.
`FIX_ACTIONS_REPORT_V2`
Fix actions report (PDF). Prioritized remediation guidance grouped by fix action, mapping vulnerabilities and weaknesses to the corrective steps that resolve them. Current production format.
`FIX_ACTIONS_REPORT`
Legacy fix actions report format. Current production uses FIX_ACTIONS_REPORT_V2. Older ops may still have this report available.
`SEGMENTATION_REPORT_V2`
Segmentation report (PDF). Summarizes how network segmentation held up during the op, including which segments were reachable from which entry points. Current production format.
`SEGMENTATION_REPORT`
Legacy segmentation report format. Current production uses SEGMENTATION_REPORT_V2. Older ops may still have this report available.
`SEGMENTATION_EXEC_SUMMARY_REPORT`
Segmentation executive summary report (PDF). High-level overview of network segmentation results, suitable for leadership and non-technical audiences. Complements the detailed SEGMENTATION_REPORT_V2.

Example

```
"EXEC_SUMMARY_REPORT_V2"
```

Pentesting: Ops

8.17.25 ScheduleOpOutput

Description

The output from a range of mutations that create ops.

Fields

`op - Op!`

The newly created Op.

Request Response

```
fragment ScheduleOpOutputFragment on ScheduleOpOutput {
  op {
    op_id
    op_type
    op_name
    op_state
    portal_op_state
    op_param_max_scope
    scheduled_at
    created_at
    nodezero_script_url
    runner_uuid
    runner_name
    portal_url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.26 UpdateOpInput

Description

Input for `Mutation.update_op`. Supports renaming, archiving, and deleting ops.

Fields

`op_name` - `StringNotEmpty`
Set a new name for the op. The name is used for display purposes in Portal and in generated reports.
`is_deleted` - `Boolean`
When set to `true`, permanently deletes the op. Deleted ops are removed from the main views and most aggregations, and are no longer retrievable in Portal or via the API.
`is_archived` - `Boolean`
When set to `true`, archives the op. Archived ops are removed from the main views and most aggregations, but remain retrievable in Portal and via the API.
`set_highest_queue_priority` - `Boolean`
Update op to be next in queue for assigned runner.

Request

```
# Example 1: Rename an op
mutation RenameOp($op_id: String!, $update_op_input: UpdateOpInput! = {
    op_name: "Renamed Pentest"
}) {
  update_op(op_id: $op_id, update_op_input: $update_op_input) {
    op {
      op_id
      op_name
    }
  }
}

# Example 2: Archive an op
mutation ArchiveOp($op_id: String!, $update_op_input: UpdateOpInput! = {
    is_archived: true
}) {
  update_op(op_id: $op_id, update_op_input: $update_op_input) {
    op {
      op_id
      op_name
    }
  }
}

# Example 3: Delete an op
mutation DeleteOp($op_id: String!, $update_op_input: UpdateOpInput! = {
    is_deleted: true
}) {
  update_op(op_id: $op_id, update_op_input: $update_op_input) {
    op {
      op_id
      op_name
    }
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Ops

8.17.27 UpdateOpOutput

Description

The output from `Mutation.update_op`.

Fields

`op` - `Op`
The updated Op.

Request Response

```
fragment UpdateOpOutputFragment on UpdateOpOutput {
  op {
    op_id
    op_type
    op_name
    op_state
    portal_op_state
    op_param_max_scope
    scheduled_at
    created_at
    launched_at
    completed_at
    duration_hms
    hosts_count
    weaknesses_count
    runner_uuid
    runner_name
    portal_url
  }
}
```

Examples may not include all available fields for a type.

8.18 Pentesting: Pentests

Access completed pentest results with high-level metrics and status information. Query pentests to retrieve summary data including discovered assets, findings counts, completion timestamps, and download comprehensive reports.

8.18.1 `pentests_page`

Description

Returns a paginated list of Pentest records in the current client account. By default, archived pentests are excluded unless the `is_archived` filter is explicitly set. Non-pentest operation types (such as external asset discovery) are always excluded.

Use the `Pentest.op_id` field to fetch full details about a specific pentest via `Query.pentest`.

Response

Returns a `PentestsPage!`

Arguments

`page_input` - `PageInput`
Optional pagination, sorting, and filter parameters.

Request Response Python
cURL h3-cli

```
query PentestsPage($page_input: PageInput) {
  pentests_page(page_input: $page_input) {
    pentests {
      ...PentestFragment
    }
  }
}

fragment PentestFragment on Pentest {
```

```
  op_id
  op_type
  name
  state
  user_name
  scheduled_at
  launched_at
  completed_at
  duration_s
  impacts_count
  attack_paths_count
  weaknesses_count
  hosts_count
  credentials_count
}
```

Examples may not include all available fields for a type.

Pentesting: Pentests

8.18.2 `pentests_count`

Description

Returns the total number of Pentest records in the current client account, after applying any filters specified in `page_input`. By default, archived pentests are excluded unless the `is_archived` filter is explicitly set. Non-pentest operation types (such as external asset discovery) are always excluded from this count.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`
Optional pagination and filter parameters. Supports filtering by op type, state, and other pentest attributes.

Request	Response	Python
cURL	h3-cli	

```
query PentestsCount($page_input: PageInput) {
  pentests_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Pentests

8.18.3 `pentest`

Description

Returns a single Pentest by its `op_id`. Returns `null` if the pentest does not exist or the caller does not have access to it.

Response

Returns a `Pentest`

Arguments

`op_id` - `String!`
Unique identifier of the pentest to retrieve.

Request | Response | Python
cURL | h3-cli

```
query Pentest($op_id: String!) {
  pentest(op_id: $op_id) {
    ...PentestFragment
  }
}

fragment PentestFragment on Pentest {
  op_id
  op_type
  name
  state
  user_name
  client_name
  min_scope
  max_scope
  exclude_scope
  scheduled_at
  launched_at
  completed_at
  duration_s
  impacts_count
  attack_paths_count
  weakness_types_count
  weaknesses_count
  hosts_count
  credentials_count
  nodezero_script_url
  nodezero_ip
}
```

Examples may not include all available fields for a type.

Pentesting: Pentests

8.18.4 `pentest_reports_zip_url`

Description

Returns a pre-signed URL to download a zip archive containing all CSV and PDF reports for the specified pentest. The pentest must have completed post-processing before reports are available.

If the zip file already exists, the pre-signed URL is returned immediately. If the zip file has not yet been generated (e.g., for older pentests prior to Sep 2022), the system will initiate an asynchronous build job and return a `[202]` error indicating the request is being processed. An email with a download link will be sent to the requesting user when the zip is ready. Subsequent calls after the zip is built will return the pre-signed URL directly.

The zip archive contains the following CSV files and their corresponding schema types:

- certificates.csv: CertificateCSV
- credentials.csv: CredentialCSV
- data_stores.csv: ShareCSV
- database_repos.csv: DatabaseRepoCSV
- docker_registries.csv: DockerRegistryCSV
- external_domains.csv: ExternalDomainCSV
- file_shares.csv: FileShareCSV
- git_repos.csv: GitRepoCSV
- hosts.csv: HostCSV
- impacts.csv: ImpactCSV
- s3_buckets.csv: S3BucketCSV
- services.csv: ServiceCSV
- users.csv: UserCSV
- weaknesses.csv: WeaknessCSV
- websites.csv: WebShareCSV

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query PentestReportsZipUrl($input: OpInput!) {
  pentest_reports_zip_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Pentests

8.18.5 Pentest

Description

A Pentest represents an autonomous penetration test or security assessment conducted by NodeZero. During a pentest, NodeZero identifies misconfigurations, vulnerabilities, exposed credentials, and other security risks within a defined network scope.

The Pentest type is a specialized view of Op that excludes non-pentest operation types such as external asset discovery. Use `Query.pentests_page` to list pentests, `Query.pentest` to retrieve a single pentest by ID, or `Mutation.create_op` to schedule a new pentest.

The Pentest type encompasses the following OpType values:

- **Internal Pentest** (`NodeZero`) -- Tests your internal network from a Docker host deployed within it.
- **External Pentest** (`ExternalAttack`) -- Tests your public-facing assets from the Horizon3.ai cloud.
- **AD Password Audit** (`ADPasswordAudit`) -- Audits Active Directory passwords for weak, breached, or reused credentials.
- **Phishing Campaign** (`Phishing`) -- Measures the downstream impact of phished credentials via an internal pentest.
- **AWS Pentest** (`AWSPentest`) -- Discovers vulnerabilities and misconfigurations in your AWS environment.
- **Azure Pentest** (`AzurePentest`) -- Assesses attack paths within your hybrid Azure Entra ID environment.
- **Kubernetes Pentest** (`K8sPentest`) -- Tests security controls and vulnerabilities within a Kubernetes cluster.

See OpType for configuration details on each type.

Fields

`op_id` - `String!`
Unique identifier for the pentest. Use this value with `Query.pentest` to fetch full pentest details.
`op_type` - `OpType`
The type of pentest. See OpType for the supported pentest types and their configuration requirements.
`name` - `String!`
Display name assigned to the pentest at scheduling time.
`state` - `PortalOpState!`
Current lifecycle state of the pentest as displayed in the Portal. See PortalOpState for possible values.
`user_name` - `String!`
Email or display name of the user who scheduled the pentest.
`client_name` - `String!`
Name of the client account that owns the pentest.
`min_scope` - `[String]`
The minimum scope defines the set of IPs and CIDR ranges that must be scanned during the pentest. When "auto-expand scope" is enabled, NodeZero may discover and scan additional IPs beyond this set, up to and including the `max_scope` (if defined). If auto-expand is disabled, only the IPs in `min_scope` are scanned.
`max_scope` - `[String]`
The maximum scope defines the upper boundary of IPs and CIDR ranges that NodeZero is permitted to discover and scan. Any IP addresses encountered outside this range are treated as out of scope, even if auto-expand is enabled.
`exclude_scope` - `[String]`
IPs and CIDR ranges explicitly excluded from the pentest scope. NodeZero will not scan these addresses regardless of other scope settings.
`git_accounts` - `[GitAccount]`
List of GitAccount entries included in the pentest for Git repository scanning.
`aws_account_ids` - `[AWSAccountId]`
List of AWS account IDs included in the pentest for AWS cloud security assessment.
`feature_flags` - `[FeatureFlag]`
Advanced configuration options (attack configurations) applied to this pentest. See FeatureFlag for details.

`scheduled_at` - `Datetime!`
Timestamp when the pentest was scheduled (i.e., created).
`launched_at` - `Datetime`
Timestamp when NodeZero began actively scanning. Null if the pentest has not yet launched.
`completed_at` - `Datetime`
Timestamp when the pentest scan completed. Null if the pentest is still running or has not launched.
`canceled_at` - `Datetime`
Timestamp when the pentest was canceled by a user. Null if the pentest was not canceled.
`etl_completed_at` - `Datetime`
Timestamp when post-processing (ETL) of pentest results completed. Results are fully available in the API once this timestamp is set.
`duration_s` - `Int`
Pentest duration in seconds, measured from `launched_at` to `completed_at`. For pentests still in progress, reflects the elapsed time since launch.
`impacts_count` - `Int`
Number of impacts found during the pentest. Each impact represents a unique combination of impact type and affected asset.
`impact_paths_count` - `Int`
Total number of attack paths (impact paths) found. Alias of `attack_paths_count`.
`attack_paths_count` - `Int`
Total number of attack paths found during the pentest. Alias of `impact_paths_count`.
`attack_paths_page` - `AttackPathsPage!`

Returns a paginated list of AttackPath records for this pentest.

Each attack path describes a sequence of steps NodeZero took to achieve an impact. Use the `AttackPath.uuid` field to fetch additional details via `Query.attack_path`.

Arguments

`page_input` - `PageInput`
`phished_impact_paths_count` - `Int!`
Number of attack paths discovered downstream of phished credentials. Alias of `phished_attack_paths_count`. Relevant only for Phishing pentests.
`phished_attack_paths_count` - `Int!`
Number of attack paths discovered downstream of phished credentials. Alias of `phished_impact_paths_count`. Relevant only for Phishing pentests.
`critical_impact_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_attack_paths_count`.
`critical_attack_paths_count` - `Int!`
Number of attack paths whose target Impact has a `Critical` Severity (context-adjusted score >= 9.0). Alias of `critical_impact_paths_count`.
`weakness_types_count` - `Int`
Number of distinct weakness types found during the pentest. Each weakness type represents a unique category of vulnerability or misconfiguration.
`weaknesses_count` - `Int`
Total number of Weakness instances found during the pentest.
`weaknesses_page` - `WeaknessesPage!`

Returns a paginated list of Weakness records for this pentest.

Use the `Weakness.uuid` field to fetch additional details about a specific weakness via `Query.weakness`.

Arguments

`page_input` - `PageInput`
`hosts_count` - `Int`
Number of in-scope hosts (IP addresses) scanned during the pentest.

`out_of_scope_hosts_count` - `Int`
Number of out-of-scope hosts (IP addresses) discovered but not scanned during the pentest.
`external_domains_count` - `Int`
Number of external domains discovered during the pentest.
`services_count` - `Int`
Number of network services (e.g., SSH, HTTP, SMB) discovered during the pentest.
`credentials_count` - `Int`
Number of credentials found during the pentest, including both discovered and confirmed credentials.
`users_count` - `Int`
Number of user accounts discovered during the pentest.
`cred_access_count` - `Int`
Number of unique credential-to-asset access relationships, where each represents a distinct credential used to access a specific asset.
`data_stores_count` - `Int`
Number of data stores found (e.g., file shares, database repositories, S3 buckets, Git repositories, Docker registries). Excludes websites.
`websites_count` - `Int`
Number of websites discovered during the pentest.
`data_resources_count` - `Long`
Aggregate count of all data resources (files, database records, etc.) across all discovered data stores.
`nodezero_script_url` - `String`
URL of the one-time launch script that downloads and starts NodeZero on the Docker host. Applicable to internal pentest types that require a local Docker host.
`nodezero_ip` - `String`
IP address assigned to the NodeZero instance running this pentest.
`schedule` - `Schedule`
The Schedule this pentest is associated with, if it was created from a recurring schedule.

Request Response

```
fragment PentestFragment on Pentest {
  op_id
  op_type
  name
  state
  user_name
  client_name
  min_scope
  max_scope
  exclude_scope
  scheduled_at
  launched_at
  completed_at
  etl_completed_at
  duration_s
  impacts_count
  attack_paths_count
  weakness_types_count
  weaknesses_count
  hosts_count
  out_of_scope_hosts_count
  services_count
  credentials_count
  users_count
  data_stores_count
  websites_count
  nodezero_script_url
  nodezero_ip
}
```

Examples may not include all available fields for a type.

Pentesting: Pentests

8.18.6 PentestsPage

Description

Contains a paginated list of Pentest records. Returned by `Query.pentests_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`pentests` - `[Pentest!]!`
The list of Pentest records for the current page.

Request Response

```
fragment PentestsPageFragment on PentestsPage {
  pentests {
    op_id
    op_type
    name
    state
    user_name
    scheduled_at
    completed_at
    duration_s
    impacts_count
    weaknesses_count
    hosts_count
  }
}
```

Examples may not include all available fields for a type.

8.19 Pentesting: Impacts

Explore attack paths (also called impacts) that show how NodeZero compromised critical assets. View attack graphs visualizing the sequence of exploits, examine attack vectors with detailed technique information, and understand the complete kill chain from initial access to objective.

8.19.1 `attack_paths_page`

Description

Returns a paginated list of AttackPaths discovered during the specified pentest. Each AttackPath represents the sequence of steps NodeZero took to achieve a specific Impact.

Use the `AttackPath.uuid` field to fetch full details about a specific AttackPath via `Query.attack_path`.

Response

Returns an `AttackPathsPage!`

Arguments

`input` - `OpInput!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query AttackPathsPage($input: OpInput!, $page_input: PageInput) {
  attack_paths_page(input: $input, page_input: $page_input) {
    attack_paths {
      ...AttackPathFragment
    }
  }
}

fragment AttackPathFragment on AttackPath {
  uuid
```

```
  impact_type
  impact_title
  name
  score
  severity
  op_id
  weakness_refs
  credential_refs
  host_refs
  time_to_finding_hms
  created_at
  affected_asset_text
  ip
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.2 `attack_paths_count`

Description

Returns the total count of AttackPaths discovered during the specified pentest. Supports the same filtering options as `Query.attack_paths_page` via `page_input`.

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query AttackPathsCount($input: OpInput!, $page_input: PageInput) {
  attack_paths_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.3 `attack_path`

Description

Returns a single AttackPath by its UUID. The AttackPath includes the target ImpactType, risk score, and the Weaknesses, Credentials, and Hosts involved.

Response

Returns an `AttackPath`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query AttackPath($uuid: String!) {
  attack_path(uuid: $uuid) {
    ...AttackPathFragment
  }
}

fragment AttackPathFragment on AttackPath {
  uuid
  impact_type
  impact_title
  name
  score
  severity
  op_id
  weakness_refs
  credential_refs
  host_refs
  time_to_finding_hms
  created_at
  affected_asset_text
  ip
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.4 `attack_paths_csv_url`

Description

Returns a temporary presigned URL to a CSV file containing all AttackPaths discovered during the specified pentest.

The CSV format is documented under ImpactCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python

cURL | h3-cli

```
query AttackPathsCsvUrl($input: OpInput!) {
  attack_paths_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.5 AttackPath

Description

An AttackPath represents the sequence of steps NodeZero took to achieve a specific Impact during a pentest. Each AttackPath links to a target Impact and includes the exploited Weaknesses, compromised Credentials, and traversed Hosts that enabled NodeZero to reach that Impact.

Use `Query.attack_paths_page` to retrieve AttackPaths for a given pentest, or `Query.attack_path` to fetch a single AttackPath by its UUID.

The `attack_vector` field provides the full step-by-step AttackVector graph detailing how NodeZero navigated through your environment to achieve the Impact.

Fields

`uuid` - `String!`
Unique identifier for this AttackPath.
`impact_type` - `ImpactType!`
The ImpactType of the target Impact (e.g., `DomainCompromise`, `RansomwareExposure`).
`impact_title` - `String!`
The display title for this AttackPath's target ImpactType.
`impact_description` - `String!`
A human-readable description of the target ImpactType.
`name` - `String!`
The display name of this AttackPath. Describes the Impact and the entity that caused it.
`attack_path_title` - `String!`
The display title of this AttackPath. Equivalent to `name`.
`score` - `Float`
The context-adjusted risk score for the target Impact, on a scale of 0 to 10. Higher values indicate greater risk.
`severity` - `Severity`
The Severity level corresponding to the target Impact's risk `score`.
`context_score_description_md` - `String`
A narrative description of this AttackPath, summarizing the target Impact along with the Weaknesses, Credentials, and Hosts involved. Returned in markdown format.
`context_score_description` - `String`
A narrative description of this AttackPath, summarizing the target Impact along with the Weaknesses, Credentials, and Hosts involved. Returned as plain text.
`attack_vector` - `AttackVector`
The AttackVector for this AttackPath. The AttackVector provides the full directed acyclic graph (DAG) of Nodes and Edges representing the step-by-step actions NodeZero took to reach the target Impact. Use the `algorithm` argument to select the graph version (e.g., `v3` for chronological layers, `v4` for a consolidated view).

Arguments

`algorithm` - `AttackVectorAlgorithmInput`
`op_id` - `String!`
The identifier of the pentest (op) in which this AttackPath was discovered.
`weakness_refs` - `[String]`
Reference strings for the Weaknesses that appear in this AttackPath. See Weakness.weakness_ref for the format of the ref string.
`weaknesses` - `[Weakness!]`
The Weakness records that appear in this AttackPath.
`credential_refs` - `[String]`
Reference strings for the Credentials that appear in this AttackPath. Each ref is formatted as `{credential_uuid},{user_or_role_name},{asset_name}`.
`credentials` - `[Credential!]`

The Credential records that appear in this AttackPath.

`host_refs` - `[String]`

Reference strings for the Hosts that appear in this AttackPath. Each ref is formatted as `{host_uuid},{ip}`.

`hosts` - `[Host!]`

The Host records that appear in this AttackPath.

`time_to_finding_hms` - `String`

The elapsed time from the start of the pentest to when the target Impact was achieved. Formatted as `HH:MM:SS`.

`time_to_finding_s` - `Int`

The elapsed time (in seconds) from the start of the pentest to when the target Impact was achieved.

`created_at` - `Datetime`

The UTC timestamp when the target Impact was achieved.

`target_entity_text` - `String`

The display name of the entity that is the direct cause of the target Impact (e.g., a Weakness or Credential).

`target_entity_short_text` - `String`

The abbreviated name of the entity that is the direct cause of the target Impact.

`affected_asset_text` - `String`

The display name of the asset affected by the target Impact.

`affected_asset_short_text` - `String`

The abbreviated name of the asset affected by the target Impact.

`ip` - `String`

The IP address of the Host affected by the target Impact.

`host_name` - `String`

The hostname of the Host affected by the target Impact.

`host_text` - `String`

The display name of the Host affected by the target Impact.

`affected_host` - `Host`

The Host record for the host affected by the target Impact.

`affected_credential` - `Credential`

The Credential affected by the target Impact, if applicable. Resolves based on the type of entity that caused the Impact.

`affected_data_store` - `DataStore`

The DataStore affected by the target Impact, if applicable (e.g., for Sensitive Data Exposure or Ransomware Exposure impacts).

`proofs_count` - `Int`

The number of Proofs captured during this AttackPath.

`proofs` - `[Proof]`

The list of Proofs captured during this AttackPath, providing evidence of the exploitation steps.

`portal_url` - `String`

The Portal URL where this AttackPath can be viewed in the Horizon3.ai web interface.

`business_risks` - `[String]`

List of business risk categories associated with this Impact (e.g., data loss, operational disruption).

Request Response

```
fragment AttackPathFragment on AttackPath {
  uuid
  impact_type
  impact_title
  impact_description
  name
  attack_path_title
  score
  severity
  context_score_description_md
  context_score_description
  op_id
  weakness_refs
  credential_refs
  host_refs
  time_to_finding_hms
  time_to_finding_s
  created_at
  target_entity_text
  target_entity_short_text
  affected_asset_text
  affected_asset_short_text
```

```
  ip
  host_name
  host_text
  proofs_count
  portal_url
  business_risks
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.6 AttackVector

Description

An AttackVector is a directed acyclic graph (DAG) of Nodes and Edges that represents the step-by-step attack path NodeZero took to reach a target entity.

The full graph structure can be retrieved via the `attack_graph` field as an AttackGraph.

NodeZero tracks AttackVectors for the following entity types:

- Impacts (via AttackPath)
- Weaknesses
- Credentials
- DataStores
- Relay events
- Privilege escalation events

Each Node in the attack graph has a `found_by_module_instance` field, which references the ModuleInstance used by NodeZero during that step of the attack path. The ModuleInstance can be used to retrieve further details about that step, including:

- Information about the module that was used (`ModuleInstance.module_meta`)
- How the module maps to the MITRE ATT&CK framework (`ModuleInstance.module_meta.mitre_mappings`)
- The commands it ran during the pentest (`ModuleInstance.action_logs`)

When summarizing an AttackVector, it can be useful to break it down by:

- The hosts involved (`host_nodes`)
- The credentials used (`credential_nodes`)
- The weaknesses exploited (`weakness_nodes`)

The complexity of an attack path is related to its length (number of nodes) and the types of nodes involved. The more credentials and weaknesses involved, the more complex the path. This complexity is relevant when assessing the likelihood that a real-world attacker could replicate the attack path.

Fields

`uuid` - `String!`
Unique identifier for this AttackVector.
`target_node` - `Node!`
The target Node of the attack vector -- i.e., the final entity that NodeZero reached at the end of this attack path.
`algorithm` - `AttackVectorAlgorithm!`
The AttackVectorAlgorithm used to generate this AttackVector (e.g., `POC`, `v3`, `v4`).
`total_score` - `Float`
The sum of risk scores across all Nodes in the attack vector.
`max_score` - `Float`
The highest risk score among all Nodes in the attack vector.
`attack_graph` - `AttackGraph`
The AttackGraph for this AttackVector -- the directed acyclic graph of Nodes and Edges that comprise the attack path.
`op_id` - `String!`
The identifier of the pentest (op) in which this AttackVector was observed.
`nodes_count` - `Int`
Total number of Nodes in this attack vector.
`credentials_count` - `Int`

Number of Credentials (including unverified potential credentials) involved in this attack vector.

`credential_nodes` - `[Node]`

The Nodes from the attack graph that represent Credentials or unverified potential credentials used during the attack.

`weaknesses_count` - `Int`

Number of Weaknesses exploited in this attack vector.

`weakness_nodes` - `[Node]`

The Nodes from the attack graph that represent Weaknesses exploited during the attack.

`hosts_count` - `Int`

Number of Hosts traversed in this attack vector.

`host_nodes` - `[Node]`

The Nodes from the attack graph that represent Hosts traversed during the attack.

`narrative` - `[Node]`

A curated subset of Nodes that form a step-by-step narrative of the attack path, suitable for summarizing the key actions NodeZero took.

`has_phished_cred` - `Boolean`

`true` if this attack vector includes credentials obtained through a phishing simulation.

`has_injected_cred` - `Boolean`

`true` if this attack vector includes credentials that were injected as inputs to the pentest.

Request Response

```
fragment AttackVectorFragment on AttackVector {
  uuid
  algorithm
  target_node {
    uuid
    label_line1
    label_line2
    score
    severity
  }
  total_score
  max_score
  attack_graph {
    nodes {
      uuid
      label_line1
      score
      severity
    }
    edges {
      from_uuid
      to_uuid
    }
  }
  op_id
  nodes_count
  credentials_count
  weaknesses_count
  hosts_count
  narrative {
    uuid
    label_line1
    description
    icon_label
  }
  has_phished_cred
  has_injected_cred
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.7 AttackGraph

Description

An AttackGraph is the directed acyclic graph (DAG) structure within an AttackVector, comprising the set of Nodes and Edges that represent the attack path.

The Nodes represent the hosts, services, Weaknesses, Credentials, and other entities that NodeZero discovered and exploited as part of the attack path.

The Edges define the sequence of steps from one node to the next that NodeZero took along the path.

The attack path can be organized into layers, where each layer represents a distinct time step. Each node is associated with a layer via Node.layer_label. All nodes in the same layer were discovered or exploited at the same time. The layers can be used to render the graph as a timeline showing the progression of the attack.

Fields

`nodes` - `[Node]`
The Nodes of the graph. Each node represents a host, service, Weakness, Credential, or other entity that NodeZero discovered and exploited as part of the attack path.
`edges` - `[Edge]`
The Edges of the graph. Each edge connects two nodes and defines a step in the sequence that NodeZero took along the attack path.

Request Response

```
fragment AttackGraphFragment on AttackGraph {
  nodes {
    uuid
    label_line1
    label_line2
    label_line3
    icon_label
    score
    severity
    description
    time_to_finding
  }
  edges {
    from_uuid
    to_uuid
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.8 Node

Description

A node in an AttackVector graph.

A Node wraps attack-vector context around a target entity such as a Host, Weakness, Credential, or service that NodeZero discovered or exploited during its attack path.

The `icon_label`, `label_line1`, `label_line2`, and `label_line3` fields are useful for rendering the node in a visual representation of the attack graph.

Fields

`uuid` - `String!`
The UUID of this node in the attack graph. Referenced by graph edges via Edge.from_uuid and Edge.to_uuid.
`attack_vector_node_uuid` - `String!`
The UUID of the underlying AVNode record, which can be used to fetch this specific node.
`found_by_module_meta` - `ModuleMeta`
Metadata about the NodeZero module that discovered this node, including its name, category, and MITRE ATT&CK mappings.
`found_by_module_instance` - `ModuleInstance`
The specific ModuleInstance that discovered this node during the pentest. Use this to access action logs and detailed execution information.
`score` - `Float`
The cyber risk score of this node, on a scale of 0 to 10. Higher values indicate greater risk.
`severity` - `Severity`
The Severity level of this node, derived from its risk `score`.
`proofs` - `[Proof]`
The Proofs captured at this step of the attack path, providing evidence of the exploitation.
`time_to_finding` - `String`
The elapsed time from the start of the pentest to when NodeZero discovered this node. Formatted as `HH:MM:SS`.
`label_line1` - `String`
The first line of the node label. Used in `v3` and `v4` attack vector renderings.
`label_line2` - `String`
The second line of the node label. Used in `v3` and `v4` attack vector renderings.
`label_line3` - `String`
The third line of the node label. Used in `v3` and `v4` attack vector renderings.
`icon_label` - `String`
The icon label describing the action NodeZero performed on this node (e.g., `Found Host`, `Exploited Weakness`).
`description` - `String`
A human-readable description of the action NodeZero took at this step of the attack path.
`subflow_nodes` - `[Node]`
Downstream nodes in the attack path that occurred at the same time as this node. These nodes are grouped in the same timestamp layer in the attack graph timeline, and are rendered alongside this node in `v3` and `v4` attack vector visualizations.
`weakness` - `Weakness`
The corresponding Weakness for this Node, if the node represents an exploited vulnerability.
`credential` - `Credential`
The corresponding Credential for this Node, if the node represents a compromised or unverified potential credential.
`host` - `Host`
The corresponding Host for this Node, if the node represents an endpoint or network address.

Request Response

```
fragment NodeFragment on Node {
  uuid
  attack_vector_node_uuid
  score
```

```
    severity
    label_line1
    label_line2
    label_line3
    icon_label
    description
    time_to_finding
    found_by_module_meta {
      id
      name
    }
    proofs {
      uuid
    }
    subflow_nodes {
      uuid
      label_line1
      score
    }
  }
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.9 Edge

Description

An edge in an AttackVector graph. Connects two Nodes in the directed acyclic graph (DAG): the source (from) node and the target (to) node. Each edge represents a single step in the attack path that NodeZero traversed.

Fields

`from_uuid` - `String!`
The UUID of the source Node (the "from" side of this edge). Corresponds to `Node.uuid` in the AttackGraph.
`to_uuid` - `String!`
The UUID of the target Node (the "to" side of this edge). Corresponds to `Node.uuid` in the AttackGraph.

Request Response

```
fragment EdgeFragment on Edge {
  from_uuid
  to_uuid
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.10 AttackPathsPage

Description

A paginated list of AttackPath records. Returned by `Query.attack_paths_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`attack_paths` - `[AttackPath!]!`
The list of AttackPath records for the current page.

Request Response

```
fragment AttackPathsPageFragment on AttackPathsPage {
  page_info {
    page_num
    page_size
  }
  attack_paths {
    uuid
    impact_type
    impact_title
    name
    score
    severity
    op_id
    portal_url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Impacts

8.19.11 AttackVectorAlgorithm

Description

The available algorithms for generating AttackVector graphs. Each algorithm produces a different level of detail and graph structure.

Values

`POC`
Also known as `v1`. The base algorithm that produces the most detailed graph. No nodes are pruned or consolidated.
`v2`
This algorithm is no longer used.
`v3`

A streamlined version of the `POC` graph with the following modifications:

- Impact and proof nodes are removed and surfaced as labels on their connected nodes.
- AWS Connection nodes are removed.
- The graph is structured chronologically by timestep "layers". All nodes created at the same time are grouped into the same layer. The first node in each layer remains in the main graph; subsequent nodes are stored as `subflow_nodes` of that first node. This structure facilitates rendering the graph on a timeline, as it appears in the Portal.

`v4`
A further consolidated version of the `v3` graph, providing the most compact representation of the attack path.

Example

```
"POC"
```

Pentesting: Impacts

8.19.12 AttackVectorAlgorithmInput

Description

The available algorithms for generating AttackVector graphs, as an input type. Accepts the same values as AttackVectorAlgorithm.

Values

`POC`
Also known as `v1`. The base algorithm that produces the most detailed graph. No nodes are pruned or consolidated.
`v2`
This algorithm is no longer used.
`v3`

A streamlined version of the `POC` graph with the following modifications:

- Impact and proof nodes are removed and surfaced as labels on their connected nodes.
- AWS Connection nodes are removed.
- The graph is structured chronologically by "layers". All nodes created at the same time are grouped into the same layer. The first node in each layer remains in the main graph; subsequent nodes are stored as `subflow_nodes` of that first node. This structure facilitates rendering the graph on a timeline, as it appears in the Portal.

`v4`
A further consolidated version of the `v3` graph, providing the most compact representation of the attack path.

Example

```
"POC"
```

Pentesting: Impacts

8.19.13 ImpactType

Description

The categories of business impact that NodeZero can demonstrate during a pentest. Each AttackPath targets one of these impact types, representing the real-world consequence that NodeZero was able to achieve by chaining together exploited Weaknesses and compromised Credentials.

Values

`AWSUserRoleCompromise`
An AWS IAM user or role has been compromised. Anything that user or role has access to -- including cloud resources, cloud services, and data -- should be considered compromised.
`AWSAccountCompromise`
An AWS account has been fully compromised. All cloud resources, cloud services, and data that exist in that AWS account should be considered fully compromised.
`AzureADUserCompromise`
DEPRECATED.
`MicrosoftEntraUserCompromise`
A Microsoft Entra user has been compromised. Anything that user has access to should be considered compromised, including access to the Microsoft Entra tenant, Microsoft 365, and Azure subscriptions.
`MicrosoftEntraAccountCompromise`
A Microsoft Entra (AzureAD) tenant has been fully compromised. Any application, service, or resource that utilizes the Entra tenant for Identity and Access Management (IAM) should be considered compromised. This includes cloud services such as Microsoft 365 and Azure-hosted resources such as virtual machines or databases.
`MicrosoftEntraFullTenantCompromise`
DEPRECATED
`BrandCompromise`
Brand compromise covers ways in which an attacker can harm your company's reputation, such as defacing the company's website, hosting malware on the company's domain, or carrying out phishing attacks that appear to originate from the company.
`BusinessEmailCompromise`
Business email compromise enables attackers to send and receive emails under the guise of a legitimate user. Attackers commonly leverage email access to conduct business accounting fraud, carry out highly targeted phishing attacks, gain access to sensitive information, and elicit trusting coworkers to perform actions on their behalf.
`CloudCompromise`
DEPRECATED.
`CloudServiceCompromise`
DEPRECATED.
`CriticalInfrastructureCompromise`
Critical infrastructure compromise involves key devices and applications that provide attackers a privileged position in the network from which they can access sensitive data and launch further attacks.
`DomainCompromise`
Full domain compromise means all hosts, domain user accounts, data, infrastructure, and applications tied to that Active Directory domain should be considered fully compromised. Additionally, any application running on a domain-joined machine or using Active Directory integration for authentication should be considered fully compromised.
`DomainUserCompromise`
A domain user account has been compromised. Anything that user account has access to should be considered compromised.
`HostCompromise`
Host compromise can lead to attackers gaining access to sensitive information, maintaining persistence within your network, and performing lateral movement to other systems.
`PerimeterBreach`
Perimeter breach indicates that an attacker can gain access to your internal network from the public internet.
`RansomwareExposure`

Ransomware exposure indicates that attackers could obtain access to business-critical data stores, encrypt them with a secret key, and demand a ransom payment before releasing the decryption key. Ransomware attacks can cause severe disruption to business operations, even after the ransom is paid, as data stores must be decrypted and affected services restored.

`SensitiveDataExposure`

Sensitive data exposure indicates that attackers could obtain user credentials, PII (Personally Identifiable Information), financial account data, and other business-critical information to further exploit or gain profit.

`ThirdPartySaaSUserCompromise`

A Third Party SaaS user has been compromised. Anything that user has access to should be considered compromised, including private messages, files, and other application-specific items.

`KubernetesIdentityCompromise`

A Kubernetes identity has been compromised. Anything that identity has access to -- including pods, services, secrets, service accounts, and data -- should be considered compromised.

`KubernetesClusterCompromise`

A Kubernetes cluster has been fully compromised. All pods, services, secrets, service accounts, and data should be considered compromised.

`WebSessionCompromise`

Web session compromise allows attackers to hijack authenticated user sessions through vulnerabilities such as cross-site scripting (XSS). Once a session is compromised, attackers can perform actions as the victim user, access sensitive data visible to that user, steal session tokens or credentials, and potentially escalate access to other systems that trust the compromised application.

`WebApplicationUserCompromise`

A web application user has been compromised. Anything that user account has access to within the application should be considered compromised, including sensitive data, application functionality, and any integrated services that trust the user's session.

`CloudResourceCompromise`

Cloud resource compromise indicates that a cloud resource such as a storage account, container registry, or other cloud-managed service has a security misconfiguration that could allow unauthorized access, data exposure, or further exploitation of the cloud environment.

`SupplyChainCompromise`

Supply chain compromise indicates that an attacker can access or tamper with software artifacts such as container images in a registry, enabling them to inject malicious code into downstream build and deployment pipelines.

Example

```
"AWSUserRoleCompromise"
```

8.20 Pentesting: Weaknesses

Query discovered vulnerabilities and security weaknesses including CVE details, severity ratings, affected assets, threat actor intelligence, and remediation guidance. Weaknesses represent exploitable security flaws found during pentesting.

8.20.1 `weaknesses_page`

Description

Returns a paginated list of Weaknesses discovered during the specified pentest.

Each Weakness record represents a specific vulnerability found on a specific asset -- there is one record per unique `vuln_id` + affected asset combination.

Use the `Weakness.uuid` field to fetch full details about a specific Weakness via `Query.weakness`.

Response

Returns a `WeaknessesPage!`

Arguments

`input` - `OpInput`

`page_input` - `PageInput`

Request Response Python

cURL h3-cli

```
query WeaknessesPage($input: OpInput, $page_input: PageInput) {
  weaknesses_page(input: $input, page_input: $page_input) {
    weaknesses {
      ...WeaknessFragment
    }
  }
}

fragment WeaknessFragment on Weakness {
  uuid
  vuln_id
  vuln_name
  vuln_short_name
  vuln_category
  vuln_cisa_kev
  op_id
  ip
  has_proof
  score
  severity
  base_score
  context_score
  time_to_finding_hms
  affected_asset_display_name
  attack_paths_count
  portal_url
  weakness_ref
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.2 `weaknesses_count`

Description

Returns the total count of Weaknesses discovered during the specified pentest. Supports the same filtering options as `Query.weaknesses_page` via `page_input`.

Response

Returns an `Int!`

Arguments

`input` - `OpInput`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query WeaknessesCount($input: OpInput, $page_input: PageInput) {
  weaknesses_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.3 `weakness`

Description

Returns a single Weakness by its UUID, including its risk scores, affected asset, vulnerability details, and related AttackPaths.

Response

Returns a `Weakness`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query Weakness($uuid: String!) {
  weakness(uuid: $uuid) {
    ...WeaknessFragment
  }
}

fragment WeaknessFragment on Weakness {
  uuid
  vuln_id
  vuln_name
  vuln_short_name
  vuln_category
  vuln_cisa_kev
  op_id
  ip
  has_proof
  score
  severity
  base_score
  context_score
  time_to_finding_hms
  affected_asset_display_name
  attack_paths_count
  portal_url
  weakness_ref
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.4 `weaknesses_csv_url`

Description

Returns a temporary presigned URL to a CSV file containing all Weaknesses discovered during the specified pentest.

Each row in the CSV represents one Weakness (one vulnerability on one asset). The CSV format is documented under WeaknessCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query WeaknessesCsvUrl($input: OpInput!) {
  weaknesses_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.5 `schedule_weakness_1cv_ops`

Description

Creates one or more 1CV (One-Click-Verify) pentests for a set of Weakness UUIDs.

A 1CV pentest is a targeted re-pentest that verifies whether a specific weakness has been remediated. Each weakness specified corresponds to a specific pentest, so its configuration — including scope, runner, and attack parameters — is derived directly from that pentest. Weaknesses from the same original pentest are grouped into a single 1CV op.

Returns the scheduled pentests and any failures via ScheduleWeakness1CVOpsOutput.

Response

Returns a `ScheduleWeakness1CVOpsOutput!`

Arguments

`input` - `ScheduleWeakness1CVOpsInput!`

Request | Response | Python

cURL | h3-cli

```
mutation ScheduleWeakness1CVOps($input: ScheduleWeakness1CVOpsInput!) {
  schedule_weakness_1cv_ops(input: $input) {
    ops_count
    ops {
      op {
        ...OpFragment
      }
    }
    failures {
      reason
      for_weakness_uuids
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_name
  op_type
  op_state
  op_param_max_scope
  runner_name
  nodezero_script_url
  one_click_verify_data {
    op_id
    vuln_ids
    weakness_uuids
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.6 Weakness

Description

A Weakness represents a specific vulnerability found on a specific asset by NodeZero during a pentest. One Weakness record is created for each unique combination of vulnerability ID (`vuln_id`) + affected asset + pentest (`op_id`).

The `vuln_id` field identifies the vulnerability type (e.g., a CVE or an H3-defined vulnerability), while the Vuln field provides static catalog information about that vulnerability, including its name, description, recommended mitigations, and CISA KEV status.

Weaknesses have three levels of risk scoring:

- `base_score` : The statically assigned risk score for the vulnerability type.
- `context_score` : An environment-adjusted score that accounts for downstream impacts this weakness enables.
- `score` : The effective risk score (equals `context_score` if set, otherwise `base_score`).

Use `Query.weaknesses_page` to list weaknesses for a pentest, or `Query.weakness` to fetch a single weakness by UUID.

To track how a weakness evolves across pentests, see WeaknessSeries.

Fields

`uuid` - `String!`
Unique identifier for this Weakness instance.
`created_at` - `Datetime!`
The UTC timestamp when NodeZero discovered this Weakness during the pentest.
`vuln_id` - `String!`
The vulnerability identifier for this Weakness (e.g., a CVE ID or an H3-defined vulnerability ID).
`vuln` - `Vuln!`
Static catalog information about the vulnerability, including its name, description, mitigations, and references. See Vuln.
`vuln_aliases` - `[String!]`
Well-known aliases for this vulnerability (e.g., alternate CVE IDs or common names).
`vuln_category` - `VulnCategory`
The high-level VulnCategory of this vulnerability (e.g., `SECURITY_MISCONFIGURATION` , `VULNERABILITY`).
`vuln_name` - `String`
The official name of the vulnerability.
`vuln_short_name` - `String`
An abbreviated name for the vulnerability, useful for display in compact views.
`vuln_cisa_kev` - `Boolean`
`true` if the vulnerability is listed in the CISA Known Exploited Vulnerabilities (KEV) catalog.
`vuln_known_ransomware_campaign_use` - `Boolean`
`true` if the vulnerability is known to be used in ransomware campaigns, according to CISA.
`op_id` - `String!`
The identifier of the pentest (op) in which this Weakness was discovered.
`ip` - `String`
The IP address of the asset where this Weakness was observed.
`has_proof` - `Boolean`
`true` if NodeZero captured proof of successful exploitation for this Weakness. See also the `proofs` field.
`proof_failure_code` - `String`
A code indicating why proof was not captured, if applicable. Possible values: `exploit_failed` , `not_configured` , `no_exploit` , `timeout` .
`proof_failure_reason` - `String`
A human-readable explanation of why proof was not captured, if applicable.
`score` - `Float`

The effective risk score for this Weakness (0-10). Equals `context_score` if available, otherwise `base_score`. Higher values indicate greater risk.

`severity` - `Severity`

The Severity level corresponding to the effective `score`.

`base_score` - `Float`

The statically assigned base risk score for the vulnerability type (0-10), before environment-specific context adjustments.

`base_severity` - `Severity`

The Severity level corresponding to the `base_score`.

`context_score` - `Float`

The environment-adjusted risk score (0-10), which accounts for the downstream impacts this Weakness enables. May be higher than the `base_score` when the weakness contributes to significant downstream impacts.

`context_severity` - `Severity`

The Severity level corresponding to the `context_score`.

`context_score_description_md` - `String`

A narrative explaining how the context score was determined, including the downstream impacts this Weakness enables. Returned in markdown format.

`context_score_description` - `String`

A narrative explaining how the context score was determined, including the downstream impacts this Weakness enables. Returned as plain text.

`time_to_finding_hms` - `String`

The elapsed time from the start of the pentest to when this Weakness was discovered. Formatted as `HH:MM:SS`.

`time_to_finding_s` - `Int`

The elapsed time (in seconds) from the start of the pentest to when this Weakness was discovered.

`affected_asset_display_name` - `String`

The display name of the asset directly affected by this Weakness.

`affected_asset_short_name` - `String`

An abbreviated name for the asset directly affected by this Weakness.

`downstream_impact_types` - `[ImpactType!]`

The list of ImpactTypes found downstream of this Weakness (e.g., `DomainCompromise`, `RansomwareExposure`).

`attack_paths_count` - `Int!`

The number of AttackPaths that include this Weakness. Equivalent to the number of Impacts found downstream of this Weakness.

Arguments

`page_input` - `PageInput`

`attack_paths_page` - `AttackPathsPage!`

A paginated list of AttackPaths that include this Weakness. Each AttackPath represents an Impact that this Weakness contributed to.

Arguments

`page_input` - `PageInput`

`proofs` - `[Proof]`

Proof records captured for this Weakness, providing evidence of successful exploitation.

`affected_host` - `Host`

The Host on which this Weakness was discovered.

`diff_status` - `DiffStatus`

Populated only when performing a diff between pentests, for example via `Query.weaknesses_diff_page`.

If a weakness was present in the first pentest but not the second, its `diff_status` will be `REMOVED`.

If a weakness was present in the second pentest but not the first, its `diff_status` will be `ADDED`.

`mitre_mappings` - `[MitreMapping!]`

The MITRE ATT&CK techniques associated with the discovery and exploitation of this Weakness.

`affected_credentials_count` - `Int!`

The number of Credentials affected by this Weakness. Typically there is only one affected Credential per Weakness, but in some edge cases there can be multiple.

Arguments

`page_input` - `PageInput`

`affected_credentials_page` - `CredentialsPage!`

A paginated list of Credentials affected by this Weakness. Typically there is only one affected Credential per Weakness, but in some edge cases there can be multiple.

Arguments

`page_input` - `PageInput`

`affected_service` - `Service`

The Service affected by this Weakness, if applicable.

`affected_application` - `Application`

The Application affected by this Weakness, if applicable.

`affected_cloud_resource` - `CloudResource`

The CloudResource affected by this Weakness, if applicable.

`affected_cloud_role` - `CloudRole`

The CloudRole affected by this Weakness, if applicable.

`affected_external_domain` - `ExternalDomain`

The ExternalDomain affected by this Weakness, if applicable.

`affected_interesting_resource` - `InterestingResource`

The InterestingResource affected by this Weakness, if applicable.

`affected_ldap_domain` - `LDAPDomain`

The LDAPDomain affected by this Weakness, if applicable.

`affected_data_store` - `DataStore`

The DataStore affected by this Weakness, if applicable.

`portal_url` - `String`

The Portal URL where this Weakness can be viewed in the Horizon3.ai web interface.

`weakness_ref` - `String`

A reference string that uniquely identifies this Weakness instance. Formatted as `{weakness_uuid},{vuln_short_name},{affected_asset_short_name}`. Used for filtering in various APIs, such as `Query.attack_paths_page`. See also AttackPath.weakness_refs.

Request Response

```
fragment WeaknessFragment on Weakness {
  uuid
  created_at
  vuln_id
  vuln_aliases
  vuln_category
  vuln_name
  vuln_short_name
  vuln_cisa_kev
  vuln_known_ransomware_campaign_use
  op_id
  ip
  has_proof
  proof_failure_code
  proof_failure_reason
  score
  severity
  base_score
  base_severity
  context_score
  context_severity
  context_score_description_md
  context_score_description
  time_to_finding_hms
  time_to_finding_s
  affected_asset_display_name
  affected_asset_short_name
  downstream_impact_types
  attack_paths_count
  diff_status
  affected_credentials_count
  portal_url
  weakness_ref
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.7 ScheduleWeakness1CVOpsFailureOutput

Description

Describes a 1CV pentest that could not be scheduled. Common reasons include the assigned runner being unavailable, no valid pentest configuration being found for the weakness, or a configuration error.

Fields

`reason` - `String!`
A human-readable explanation of why the 1CV pentest could not be scheduled.
`op_form` - `ScheduleOpForm`
The ScheduleOpForm configuration that was built for the failed op, if one was available before the error occurred. May be null if the configuration could not be determined.
`for_weakness_uuids` - `[String!]`
The Weakness UUIDs that were associated with the op that failed to schedule.

Request Response

```
fragment ScheduleWeakness1CVOpsFailureOutputFragment on ScheduleWeakness1CVOpsFailureOutput {
  reason
  for_weakness_uuids
  op_form {
    op_param_max_scope
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.8 ScheduleWeakness1CVOpsInput

Description

Input for schedule_weakness_1cv_ops. Provide the UUIDs of the Weakness records you want to re-verify.

Fields

`weakness_uuids` - `[StringNotEmpty]`
UUIDs of the Weakness records to re-verify. Pentest configurations are derived automatically from the pentest in which each weakness was found. Weaknesses from the same original pentest are batched into a single 1CV op.
`skip_if_active_or_queued` - `Boolean`
When set to `true`, any weakness that already has an active or queued 1CV op will be skipped rather than scheduled. Skipped ops appear in `failures` with a reason that identifies the existing op. When omitted or `false`, the default behavior is preserved and no skip check is performed.
`runner_uuid` - `StringNotEmpty`
ID of the NodeZero Runner to use for all 1CV ops scheduled by this call. When provided, all ops use this runner instead of the runner from each weakness's original pentest. The runner must belong to the same client account associated with the `weakness_uuids` specified.
`run_immediately` - `Boolean`
When `true`, the first op scheduled on each runner will pause any currently running action on that runner and move to the top of the queue. If multiple ops target the same runner (either by coincidence or via `runner_uuid`), only the first can run immediately — the rest queue normally behind it. Requires runner queueing to be enabled for the account and the runner to support queueing.

Request

```
mutation Schedule1CVOps($input: ScheduleWeakness1CVOpsInput! = {
  weakness_uuids: ["1234abcd-1234-abcd-1234-abcd1234abcd"]
  skip_if_active_or_queued: false
}) {
  schedule_weakness_1cv_ops(input: $input) {
    ops_count
    ops {
      op {
        op_id
        op_name
      }
    }
    failures {
      reason
      for_weakness_uuids
    }
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.9 ScheduleWeakness1CVOpsOutput

Description

Output from schedule_weakness_1cv_ops. Contains the successfully scheduled 1CV pentests and any that could not be scheduled.

Fields

`ops_count` - `Int!`
The number of 1CV pentests successfully scheduled.
`ops` - `[ScheduleOpOutput!]`
The 1CV pentests that were successfully scheduled. Each entry is a ScheduleOpOutput containing the newly created op.
`failures` - `[ScheduleWeakness1CVOpsFailureOutput!]`
Pentests that could not be scheduled, along with the reason for each failure. See ScheduleWeakness1CVOpsFailureOutput for details.

Request Response

```
fragment ScheduleWeakness1CVOpsOutputFragment on ScheduleWeakness1CVOpsOutput {
  ops_count
  ops {
    op {
      op_id
      op_name
    }
  }
  failures {
    reason
    for_weakness_uuids
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.10 Severity

Description

Severity levels derived from a finding's cyber risk score (0-10 scale). Used throughout the API to classify the risk level of Weaknesses, AttackPaths, and Nodes.

```
       score <= 0  : INFO
0   <  score <  4.0: LOW
4.0 <= score <  7.0: MEDIUM
7.0 <= score <  9.0: HIGH
9.0 <= score       : CRITICAL
```

Values

`INFO`
Informational finding. Score is 0 or below.
`LOW`
Low-severity finding. Score is between 0 (exclusive) and 4.0.
`MEDIUM`
Medium-severity finding. Score is between 4.0 (inclusive) and 7.0.
`HIGH`
High-severity finding. Score is between 7.0 (inclusive) and 9.0.
`CRITICAL`
Critical-severity finding. Score is 9.0 or above.

Example

```
"INFO"
```

Pentesting: Weaknesses

8.20.11 ThreatActor

Description

Information about a known threat actor group that is associated with specific vulnerabilities. Threat actors are linked to Vulns via the `Vuln.threat_actors` field, enabling you to assess whether vulnerabilities in your environment are being actively targeted by known adversaries.

Fields

`name` - `String!`
The commonly known name of the threat actor group.
`short_description` - `String`
A brief summary of the threat actor's known activities and targets.
`description` - `String!`
A detailed description of the threat actor, including tactics, techniques, and known campaigns.
`origin_country` - `String!`
The country or region where the threat actor is believed to originate.
`remediation_urgency` - `String!`
A brief assessment of how urgently vulnerabilities exploited by this threat actor should be remediated.
`aliases` - `[String!]!`
Well-known alternative names for this threat actor (e.g., APT group designations used by different vendors).
`references` - `[ThreatActorReference!]!`
External references and resources with additional information about this threat actor. See ThreatActorReference.

Request Response

```
fragment ThreatActorFragment on ThreatActor {
  name
  short_description
  description
  origin_country
  remediation_urgency
  aliases
  references {
    display_name
    url
  }
}
```

Examples may not include all available fields for a type.

8.20.12 ThreatActorReference

Description

A reference to external documentation or resources related to a ThreatActor.

Fields

`display_name` - `String!`
The display name or title of the reference document.
`url` - `String!`
The URL of the reference document.

Request Response

```
fragment ThreatActorReferenceFragment on ThreatActorReference {
  display_name
  url
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.13 Vuln

Description

Static catalog information about a vulnerability type. Each Weakness references a Vuln via the `vuln_id` field to provide details about the vulnerability, including its name, description, base risk score, recommended mitigations, CISA KEV status, and associated ThreatActors.

Fields

`id` - `String!`
The unique identifier for this Vuln (e.g., a CVE ID like `CVE-2021-44228` or an H3-defined ID).
`base_score` - `Float!`
The statically assigned base risk score for this Vuln (0-10). Note that the corresponding Weakness may have a different effective score, based on context scoring that considers the weakness's actual impact within the environment.
`name` - `String!`
The official name of the Vuln.
`short_name` - `String!`
An abbreviated name for the Vuln, suitable for display in compact views.
`description` - `String!`
A detailed description of the vulnerability, including how it can be exploited.
`impact` - `String`
A description of the potential impact if an attacker successfully exploited this Vuln.
`category` - `VulnCategory!`
The high-level VulnCategory for this Vuln (e.g., `SECURITY_MISCONFIGURATION`, `VULNERABILITY`).
`base_severity` - `Severity!`
The Severity level corresponding to the Vuln's `base_score`.
`aliases` - `[String]!`
Well-known aliases for this Vuln (e.g., alternate CVE IDs or common names).
`mitigations` - `[VulnMitigation]!`
Recommended remediation actions to mitigate this Vuln. See VulnMitigation.
`references` - `[VulnReference]!`
External references and resources related to this Vuln. See VulnReference.
`impact_categories` - `[VulnImpactCategory]!`
Categories describing the potential impact if this Vuln were exploited. See VulnImpactCategory.
`one_click_verify` - `Boolean!`
`true` if this Vuln supports one-click verification, allowing you to quickly re-test whether the vulnerability has been remediated. Most Vulns are one-click verifiable. Vulns that require complex multi-step attack paths may not be one-click verifiable.
`cisa_kev` - `Boolean!`
`true` if this Vuln is listed in the CISA Known Exploited Vulnerabilities (KEV) catalog, indicating active exploitation in the wild.
`known_ransomware_campaign_use` - `Boolean`
`true` if this Vuln is known to be used in ransomware campaigns, according to CISA.
`threat_actors` - `[ThreatActor!]!`
ThreatActors that are known to exploit this Vuln.

Request Response

```
fragment VulnFragment on Vuln {
  id
  name
  short_name
  description
  base_score
  base_severity
  category
  impact
  aliases
  mitigations {
```

```
    description
  }
  references {
    name
    url
  }
  impact_categories
  one_click_verify
  cisa_kev
  known_ransomware_campaign_use
  threat_actors {
    name
    short_description
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.14 VulnCategory

Description

High-level categories for classifying vulnerabilities.

Values

`SECURITY_MISCONFIGURATION`
The vulnerability stems from a security misconfiguration of an asset (e.g., default settings, open ports, unnecessary services).
`VULNERABILITY`
A product or asset vulnerability, such as a software bug or known CVE.
`SECURITY_CONTROLS`
The vulnerability stems from ineffective or insufficient security controls or policies (e.g., missing MFA, weak access controls).
`CREDENTIALS`
The vulnerability is related to credential management or policies (e.g., weak passwords, default credentials, credential reuse).

Example

```
"SECURITY_MISCONFIGURATION"
```

Pentesting: Weaknesses

8.20.15 VulnImpactCategory

Description

Categories describing the potential impact if a vulnerability is exploited.

Values

`FILE_UPLOAD`
Exploitation may allow unauthorized file uploads to the target system.
`DENIAL_OF_SERVICE`
Exploitation may result in a Denial-of-Service (DoS) condition, disrupting availability.
`REMOTE_CODE_EXECUTION`
Exploitation may enable remote code execution on the target system.
`INFORMATION_DISCLOSURE`
Exploitation may lead to disclosure of sensitive information.
`PRIVILEGE_ESCALATION`
Exploitation may enable an attacker to escalate privileges beyond their intended level.
`DEFACEMENT`
Exploitation may enable defacement of a website or web application.
`IMPERSONATION`
Exploitation may allow an attacker to impersonate a legitimate user.
`UNAUTHORIZED_ACCESS`
Exploitation may grant unauthorized access to a system or resource.

Example

```
"FILE_UPLOAD"
```

Pentesting: Weaknesses

8.20.16 VulnMitigation

Description

A recommended remediation action to mitigate a Vuln.

Fields

`description` - `String!`
A description of the recommended mitigation action.

Request Response

```
fragment VulnMitigationFragment on VulnMitigation {
  description
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.17 VulnReference

Description

A reference to external documentation or resources related to a Vuln.

Fields

`name` - `String!`
The display name or title of the reference document.
`url` - `String`
The URL of the reference document.

Request Response

```
fragment VulnReferenceFragment on VulnReference {
  name
  url
}
```

Examples may not include all available fields for a type.

Pentesting: Weaknesses

8.20.18 WeaknessesPage

Description

A paginated list of Weakness records. Returned by `Query.weaknesses_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`weaknesses` - `[Weakness!]!`
The list of Weakness records for the current page.

Request Response

```
fragment WeaknessesPageFragment on WeaknessesPage {
  page_info {
    page_num
    page_size
  }
  weaknesses {
    uuid
    vuln_id
    vuln_name
    score
    severity
    ip
    op_id
    portal_url
  }
}
```

Examples may not include all available fields for a type.

8.21 Pentesting: Proofs

Access proof artifacts demonstrating successful exploitation including command outputs, screenshots, and enumeration results. Proof provides forensic evidence validating each finding discovered during pentesting.

8.21.1 Proof

Description

A Proof represents evidence captured by NodeZero during a pentest that demonstrates a specific finding, such as a compromised credential, an exploited weakness, or a discovered resource.

Each Proof is one of three types, as indicated by the `proof_type` field:

- **ScreenshotProofResource** -- A screenshot image. Use the `presigned_url` field to download it. Presigned URLs are generated on demand and expire after a short period.
- **CommandOutputProofResource** -- Output from one or more shell commands executed by NodeZero. Retrieve the data via the `command_outputs` field, which returns a list of CommandOutput objects.
- **EnumerationProofResource** -- Structured enumeration data (e.g., processes, programs, or users discovered on a host). Retrieve the data via the `enumeration_outputs` field, which returns a list of EnumerationProofOutput objects.

A single Proof may apply to more than one entity (e.g., a Weakness, a Credential, or a web resource). The `proven_entity_labels` field lists labels for all associated entities.

Fields

`uuid` - `String!`
Unique identifier for this Proof.
`created_at` - `Datetime!`

When this Proof record was created (UTC). See also `collected_at`, which indicates when the proof was actually captured during the pentest.

`proof_type` - `String!`

The type of proof. One of `CommandOutputProofResource`, `ScreenshotProofResource`, or `EnumerationProofResource`.

`collected_at` - `Datetime`

When this Proof was captured by NodeZero during the pentest (UTC). This represents the actual collection time, as opposed to `created_at` which is the record creation time.

`annotation` - `String`

A human-readable caption or annotation that describes what this Proof demonstrates.

`url` - `String`

The target URL that was captured in a screenshot. Only populated for screenshot-type proofs.

`presigned_url` - `String`

A presigned URL for downloading the screenshot image. Only populated for screenshot-type proofs. Presigned URLs are generated on demand at request time and expire within a short period.

`command_outputs` - `[CommandOutput]`

The list of CommandOutput objects containing the shell commands and their output. Only populated for command-output-type proofs.

`enumeration_type` - `String`

The type of enumeration performed. Known values include `Process`, `Program`, and `User`. Only populated for enumeration-type proofs.

`enumeration_outputs` - `[EnumerationProofOutput]`

The structured enumeration data captured during discovery. Returns a list of EnumerationProofOutput objects. Only populated for enumeration-type proofs.

`proven_entity_labels` - `[String]`

A set of human-readable labels identifying the entities this Proof applies to. A single Proof may apply to more than one entity -- for example, a Weakness, a Credential, and/or a web resource.

`found_by_module_meta` - `ModuleMeta`

Metadata about the NodeZero module that found this Proof, including associated MITRE ATT&CK tactics and techniques. See ModuleMeta.

`op_id` - `String!`

The ID of the Op (pentest) in which this Proof was captured.

Request Response

```
fragment ProofFragment on Proof {
  uuid
  created_at
  proof_type
  collected_at
  annotation
  url
  presigned_url
  command_outputs {
    command
    output
  }
  enumeration_type
  enumeration_outputs {
    name
    metadata
  }
  proven_entity_labels
  found_by_module_meta {
    id
    name
    description
    mitre_mappings {
      mitre_tactic_id
      mitre_technique_id
      mitre_subtechnique_id
    }
  }
  op_id
}
```

Examples may not include all available fields for a type.

Pentesting: Proofs

8.21.2 CommandOutput

Description

Output from a shell command executed by NodeZero during a pentest. Each CommandOutput contains the command that was run and the resulting output. A single Proof of type `CommandOutputProofResource` may contain multiple CommandOutput entries.

Fields

`command` - `String`
The shell command that was executed by NodeZero.
`output` - `String`
The text output produced by the command.

Request Response

```
fragment CommandOutputFragment on CommandOutput {
  command
  output
}
```

Examples may not include all available fields for a type.

Pentesting: Proofs

8.21.3 EnumerationProofOutput

Description

A structured enumeration item captured during host discovery. Enumeration proofs contain lists of items such as running processes, installed programs, or user accounts found on a compromised host. Each item has a display name and a set of metadata key-value pairs with additional detail.

Fields

`name` - `String!`
The primary display name for this enumeration item (e.g., a process name, program name, or username).
`metadata` - `JSONObject!`
Additional metadata for this enumeration item, provided as key-value pairs. The available keys vary by enumeration type.

Request Response

```
fragment EnumerationProofOutputFragment on EnumerationProofOutput {
  name
  metadata
}
```

Examples may not include all available fields for a type.

8.22 Pentesting: Credentials

View credentials compromised during pentesting including usernames, password hashes, cloud roles, and their usage in attacks. Track credential exposure across your environment and identify weak password patterns in Active Directory.

8.22.1 `credentials_page`

Description

Returns a paginated list of Credential records for the given pentest. Anonymous credentials are excluded unless they were used in an AttackPath and have a high risk score.

Response

Returns a `CredentialsPage!`

Arguments

`input` - `OpInput!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query CredentialsPage($input: OpInput!, $page_input: PageInput) {
  credentials_page(input: $input, page_input: $page_input) {
    credentials {
      ...CredentialPageFragment
    }
  }
}

fragment CredentialPageFragment on Credential {
  uuid
  created_at
  cred_type
  user_name
  ip
  product
  cleartext
  has_proof
  is_validated
  score
```

```
  severity
  weaknesses_count
  attack_paths_count
  display_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.2 `credentials_count`

Description

Returns the total number of Credential records for the given pentest. Anonymous credentials are excluded unless they were used in an AttackPath and have a high risk score. Uses the same filtering logic as `Query.credentials_page`.

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query CredentialsCount($input: OpInput!, $page_input: PageInput) {
  credentials_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.3 `credential`

Description

Retrieves a single Credential by its unique identifier.

Response

Returns a `Credential`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query Credential($uuid: String!) {
  credential(uuid: $uuid) {
    ...CredentialFragment
  }
}

fragment CredentialFragment on Credential {
  uuid
  created_at
  cred_type
  user_name
  ip
  product
  cleartext
  hash
  has_proof
  is_validated
  role
  role_name
  score
  severity
  weaknesses_count
  attack_paths_count
  display_name
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.4 `credentials_csv_url`

Description

Returns a presigned URL to download all Credential records for the given pentest as a CSV file. The presigned URL is generated on demand and expires after a short period.

The CSV format is documented under CredentialCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query CredentialsCsvUrl($input: OpInput!) {
  credentials_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.5 `activedir_passwords_csv_url`

Description

Returns a presigned URL to download Active Directory password audit results as a CSV file. The presigned URL is generated on demand and expires after a short period. The CSV format is documented under ActiveDirPasswordCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query ActivedirPasswordsCsvUrl($input: OpInput!) {
  activedir_passwords_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.6 Credential

Description

A Credential represents a user account, password, hash, API key, or other secret discovered or validated by NodeZero during a pentest.

Credentials encompass a range of access types, from traditional username/password pairs to cloud IAM roles, SNMP community strings, Kubernetes service accounts, and more. The `cred_type` field indicates the category of the credential, and the `role` field indicates the privilege level (e.g., Domain Admin, Local User).

Use `Query.credentials_page` to list credentials for a pentest, `Query.credential` to fetch a single credential by UUID, or `Query.credentials_csv_url` to export credentials as CSV.

Fields

`uuid` - `String!`
Unique identifier for this Credential.
`created_at` - `Datetime!`
When this Credential was discovered by NodeZero during the pentest (UTC).
`cred_type` - `String`
The type of Credential. Common values include `STANDARD` (username/password), `AWS`, `SNMP_COMMUNITY_STRING`, `PUBLIC_WEB`, `SMB_NULL_BIND`, `LDAP_NULL_BIND`, `NFS_NULL_BIND`, `API_ANONYMOUS`, and `AWS_CROSS_ACCOUNT_USER`.
`user_name` - `String`
The name of the user or subject associated with this Credential. This resolves to the first non-empty value among the Kubernetes identity name, cloud role name, user name, or API key ID. Defaults to `(anonymous)` if none are available.
`k8s_identity_name` - `String`
The Kubernetes identity name (e.g., service account) associated with this Credential, if applicable.
`cloud_role_name` - `String`
The name of the CloudRole (e.g., an AWS IAM role or Azure role) associated with this Credential.
`cloud_account_id` - `String`
The ID of the cloud account (e.g., an AWS account ID) where this Credential was found.
`ip` - `String`
One of the IPs where this Credential was used. Some Credentials (e.g., Domain User Credentials) may be used across multiple IPs. Use `ips` for the complete list.
`ips` - `[String]`
The complete list of IP addresses where this Credential was used. Some Credentials (e.g., Domain User Credentials) may span multiple IPs.
`sockets` - `[String]`
The list of IP:port socket pairs where this Credential was used. Each entry is formatted as `{ip}:{port}`.
`product` - `String`
The product or service name (e.g., `ssh`, `smb`, `rdp`) that this Credential was used to authenticate against.
`permissions_list` - `[String]`
The list of permissions (e.g., `read`, `write`, `delete`) associated with this Credential, indicating what access it grants.
`api_key_id` - `String`
The API key ID (e.g., an AWS Access Key ID) associated with this Credential.
`api_secret_key` - `String`
The API key secret (e.g., an AWS Secret Access Key) associated with this Credential. Values are masked for security.
`api_session_token` - `String`
The API key session token (e.g., an AWS Session Token) associated with this Credential. Values are masked for security.
`cleartext` - `String`
The cleartext password for this Credential, with most characters masked for security.
`hash` - `String`
The password hash for this Credential, with most characters masked for security.
`has_proof` - `Boolean`

`true` if NodeZero obtained Proof demonstrating this Credential. Defaults to `false`.

`proofs` - `[Proof]`

The list of Proof objects demonstrating this Credential.

`is_validated` - `Boolean`

`true` if NodeZero successfully used and validated this Credential during the pentest, confirming it grants access.

`hosts` - `[Host]`

The list of Host endpoints accessed by this Credential during the pentest.

`is_injected` - `Boolean`

`true` if this Credential was injected (i.e., provided as an input to the pentest) rather than discovered autonomously by NodeZero.

`is_cracked` - `Boolean`

`true` if NodeZero successfully cracked the password hash for this Credential during the pentest.

`is_local_admin` - `Boolean`

`true` if this Credential has local administrator privileges on a host.

`is_domain_admin` - `Boolean`

`true` if this Credential has Domain Admin privileges in Active Directory.

`is_entra_global_admin` - `Boolean`

`true` if this Credential has Global Admin privileges in Microsoft Entra (formerly Azure AD).

`role_name` - `String`

Human-readable display name for the `role` (e.g., `Domain User`, `Local Admin`, `AWS Role`).

`context_score_description` - `String`

A human-readable explanation of how the Credential's risk score was determined, including context about downstream impacts.

`score` - `Float`

The risk score for this Credential (0.0 to 10.0). Higher scores indicate greater risk. Factors include the credential's privilege level and downstream impacts.

`severity` - `Severity`

The Severity level derived from the `score` (e.g., `CRITICAL`, `HIGH`, `MEDIUM`, `LOW`, `INFO`).

`weaknesses_count` - `Int!`

The number of Weakness findings associated with this Credential.

`weaknesses_page` - `WeaknessesPage!`

A paginated list of Weakness findings associated with this Credential.

Arguments

`page_input` - `PageInput`

`services_count` - `Int!`

The number of Service endpoints accessed using this Credential.

`data_stores_count` - `Int!`

The number of DataStore resources (e.g., file shares, databases) accessed using this Credential.

`data_stores_page` - `DataStoresPage!`

A paginated list of DataStore resources (e.g., file shares, databases) accessed using this Credential.

Arguments

`page_input` - `PageInput`

`op_id` - `String!`

The ID of the Op (pentest) in which this Credential was found.

`op` - `Op!`

The Op (pentest) in which this Credential was found.

`downstream_impact_types` - `[ImpactType!]`

The list of ImpactType values representing the business impacts found downstream of this Credential. Equivalent to the impact types in `attack_paths_page`.

`attack_paths_count` - `Int!`

The number of AttackPath records this Credential is included in. This equals the count of business impacts found downstream of this Credential.

Arguments

`page_input` - `PageInput`

`attack_paths_page` - `AttackPathsPage!`

A paginated list of AttackPath records this Credential is included in, representing the business impacts found downstream.

Arguments

`page_input` - `PageInput`

`time_to_finding_hms` - `String`

The elapsed time from pentest start until this Credential was found, formatted as `hh:mm:ss`.

`time_to_finding_s` - `Int`

The elapsed time from pentest start until this Credential was found, in seconds.

`display_name` - `String`

A computed display name for this Credential. Depending on the credential type, this may be the username, `username@ip`, or the AWS access key ID.

`portal_url` - `String`

A direct URL to view this Credential in the NodeZero Portal.

Request Response

```
fragment CredentialFragment on Credential {
  uuid
  created_at
  cred_type
  user_name
  cloud_role_name
  cloud_account_id
  ip
  ips
  sockets
  product
  permissions_list
  cleartext
  hash
  has_proof
  is_validated
  is_injected
  is_cracked
  is_local_admin
  is_domain_admin
  role
  role_name
  score
  severity
  weaknesses_count
  services_count
  data_stores_count
  attack_paths_count
  time_to_finding_hms
  display_name
  op_id
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.7 CloudRole

Description

Represents a security role in a cloud environment, such as an AWS IAM role, an Azure role, or another cloud provider role. CloudRoles are discovered when NodeZero enumerates cloud infrastructure during a pentest and may be associated with one or more Credential records.

Fields

`uuid` - `String!`
Unique identifier for this CloudRole.
`created_at` - `Datetime!`
When this CloudRole was discovered during the pentest (UTC).
`name` - `String!`
The name of the cloud role (e.g., an AWS IAM role name or Azure role name).
`cloud_account_id` - `String`
The ID of the cloud account (e.g., an AWS account ID) where this CloudRole is defined.
`cloud_provider_name` - `String`
The name of the cloud provider for this CloudRole (e.g., `AWS`, `Azure`).
`op_id` - `String!`
The ID of the Op (pentest) in which this CloudRole was discovered.

Request Response

```
fragment CloudRoleFragment on CloudRole {
  uuid
  created_at
  name
  cloud_account_id
  cloud_provider_name
  op_id
}
```

Examples may not include all available fields for a type.

Pentesting: Credentials

8.22.8 CredentialsPage

Description

A paginated list of Credential records. Returned by `Query.credentials_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`credentials` - `[Credential!]!`
The list of Credential records for the current page.

Request Response

```
fragment CredentialsPageFragment on CredentialsPage {
  page_info {
    page_num
    page_size
  }
  credentials {
    uuid
    cred_type
    user_name
    ip
    product
    role_name
    score
    severity
    display_name
    op_id
    portal_url
  }
}
```

Examples may not include all available fields for a type.

8.23 Pentesting: Data Stores

Query discovered data stores and databases identified during pentesting including file shares, databases, cloud storage, and sensitive resources. View accessibility, permissions, and interesting files or data discovered.

8.23.1 `data_stores_page`

Description

Returns a paginated list of DataStores discovered during the specified pentest. Supports filtering (e.g., by `data_store_type`, `severity`, `downstream_impact_types`), sorting, and text search via `page_input`.

Response

Returns a `DataStoresPage!`

Arguments

`input` - `OpInput!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query DataStoresPage($input: OpInput!, $page_input: PageInput) {
  data_stores_page(input: $input, page_input: $page_input) {
    data_stores {
      ...DataStoreFragment
    }
  }
}

fragment DataStoreFragment on DataStore {
  uuid
  data_store_type
```

```
  name
  title
  ip
  port
  service_type
  data_resources_count
  data_resources_label
  sensitive_resources_count
  is_authenticated
  score
  severity
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.2 `data_stores_count`

Description

Returns the count of DataStores discovered during the specified pentest. Supports filtering via `page_input` (e.g., by `data_store_type`, `severity`, or text search).

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query DataStoresCount($input: OpInput!, $page_input: PageInput) {
  data_stores_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.3 `data_store`

Description

Returns the DataStore with the given `uuid`. A DataStore represents a data repository (file share, database, S3 bucket, etc.) discovered during a pentest.

Response

Returns a `DataStore`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query DataStore($uuid: String!) {
  data_store(uuid: $uuid) {
    ...DataStoreFragment
  }
}

fragment DataStoreFragment on DataStore {
  uuid
  data_store_type
  created_at
  name
  title
  ip
  port
  host_name
  service_type
  address
  data_resources_count
  data_resources_label
  sensitive_resources_count
  is_authenticated
  permissions_list
  score
  severity
  op_id
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.4 `data_stores_csv_url`

Description

Returns a temporary presigned URL to a CSV export of all DataStores discovered during the specified pentest. The URL expires after a short time.

The CSV format is documented under ShareCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query DataStoresCSVUrl($input: OpInput!) {
  data_stores_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.5 DataStore

Description

A DataStore represents a data repository discovered by NodeZero during a pentest. DataStores include file shares (SMB, NFS), databases (PostgreSQL, MySQL, DynamoDB), AWS S3 buckets, Azure Storage accounts, Docker registries, Git repositories (GitHub, GitLab), and Microsoft 365 resources (Outlook, OneDrive, SharePoint).

The `data_store_type` field indicates which category of data store this record represents. Each DataStore is scored based on the sensitivity of the data it contains and the weaknesses that expose it.

Use `Query.data_stores_page` to list DataStores for a pentest, `Query.data_store` to fetch a single DataStore by UUID, or `Query.data_stores_csv_url` to export DataStores as CSV.

Fields

`uuid` - `String!`
Unique identifier for this DataStore.
`created_at` - `Datetime!`
When this DataStore was first observed during the pentest (UTC).
`name` - `String`
The name of this DataStore (e.g., share name, bucket name, repository name).
`name_and_account` - `String`
The DataStore name prefixed by the associated account when applicable, formatted as `<account_name> : <data_store_name>`. For Git repositories, this includes the Git account name.
`title` - `String`
A human-readable display title for this DataStore, e.g., `File Share ADMIN$ on 123.4.5.6 Port 445`.
`ip` - `String`
The IP address of the host where this DataStore resides. Falls back to the DNS FQDN if no IP is available.
`port` - `Int`
The network port on which this DataStore's service is accessible.
`host_name` - `String`
The hostname of the machine where this DataStore resides.
`service_type` - `String`

The protocol or service name associated with this DataStore. The value varies by `data_store_type`:

- **FileShare**: The SMB/NFS service name (e.g., `SMB`, `NFS`)
- **DockerRegistry**: `Docker Registry`
- **DatabaseRepo**: The database engine name (e.g., `PostgreSQL`, `MySQL`, `AWS DynamoDB`)
- **S3Bucket / AzureStorage**: The cloud service name
- **GitRepo**: The Git hosting provider (e.g., `GitHub`, `GitLab`)
- **Microsoft365**: `Microsoft Outlook`, `Microsoft OneDrive`, or `Microsoft SharePoint`

`account` - `String`

The account or owner associated with this DataStore. The value varies by `data_store_type`:

- **S3Bucket / AzureStorage / Microsoft365**: The cloud account ID
- **GitRepo**: The Git account name (e.g., the GitHub or GitLab organization)
- Other types: `null`

`address` - `String`

The network address or URL used to access this DataStore. The value varies by `data_store_type`:

- **FileShare / DockerRegistry / DatabaseRepo**: The IP address or DNS FQDN of the hosting service
- **S3Bucket**: The S3 URI (e.g., `s3://bucket-name`)
- **GitRepo**: The Git clone URL
- **Microsoft365 / AzureStorage**: The resource name or URL

`data_resources_count` - `Long`

The total number of data resources discovered in this DataStore. The meaning varies by `data_store_type`:

- **FileShare / S3Bucket**: Total file count
- **DockerRegistry**: Total image count
- **DatabaseRepo**: Total record count
- **GitRepo**: Sensitive findings count
- **Microsoft365 (Outlook)**: Total email count

`data_resources_label` - `String`

A human-readable label describing the unit of `data_resources_count`. Possible values:

- `Files` (FileShare, OneDrive, SharePoint)
- `Objects` (S3Bucket, AzureStorage)
- `Emails` (Outlook)
- `Sensitive Findings` (GitRepo)
- `Records` (DatabaseRepo)
- `Images` (DockerRegistry)

`sensitive_resources_count` - `Int!`
The number of sensitive resources found in this DataStore. For FileShares and S3Buckets, this is the count of sensitive files. For GitRepos, this is the count of sensitive findings. Not applicable for DatabaseRepo, DockerRegistry, or Microsoft365 types.
`is_authenticated` - `Boolean`
`true` if the DataStore was accessed using a non-anonymous credential during the pentest.
`cloud_resource_arn` - `String`
The Amazon Resource Name (ARN) of the cloud resource that hosts this DataStore, if applicable.
`cloud_service_name` - `String`
The name of the cloud service hosting this DataStore (e.g., `Amazon S3`, `Azure Blob Storage`). Applies to S3Buckets, AzureStorage, and cloud-hosted resources.
`cloud_provider_name` - `String`
The cloud provider name for this DataStore (e.g., `Amazon`, `Google`, `Microsoft`). Applies to S3Buckets, AzureStorage, and cloud-hosted resources.
`permissions_list` - `[String]`
Aggregated list of permissions across all credentials that accessed this DataStore. Common values include `read`, `write`, and `list`.
`aws_anonymous_user_permissions_list` - `[String]`
Permissions granted to the AWS anonymous user on this S3 bucket. Only populated for DataStores of type `S3Bucket`.
`aws_cross_account_user_permissions_list` - `[String]`
Permissions granted to AWS cross-account users on this S3 bucket. Only populated for DataStores of type `S3Bucket`.
`op_id` - `String!`
The ID of the Op in which this DataStore was discovered.
`score` - `Float`
The overall risk score for this DataStore, rounded to one decimal place. Equal to `context_score` when available, otherwise `base_score`.
`severity` - `Severity`

The Severity level corresponding to `score` (e.g., `CRITICAL`, `HIGH`, `MEDIUM`, `LOW`, `INFO`).

`base_score` - `Float`

The base risk score for this DataStore before contextual impact is factored in.

`base_severity` - `Severity`

The Severity level corresponding to `base_score`.

`context_score` - `Float`

The context-adjusted risk score that accounts for downstream impacts, such as weaknesses and sensitive data reachable through this DataStore.

`context_severity` - `Severity`

The Severity level corresponding to `context_score`.

`sensitive_data_item_count` - `Int`

Total count of sensitive data items found across all InterestingResources in this DataStore. Sensitive data items include PII and PCI data such as social security numbers, credit card numbers, and similar.

`sensitive_data_item_types_and_counts` - `[String!]`

List of distinct sensitive data types with their aggregate counts across all InterestingResources in this DataStore. Each element is formatted as `SensitiveDataType,count`, e.g. `["CREDIT_CARD,2", ...]`.

`sensitive_data_item_titles_and_counts` - `[String!]`

List of distinct sensitive data titles with their aggregate counts across all InterestingResources in this DataStore. Each element is formatted as `title,count`, e.g. `["Credit Card Number,2", ...]`.

`downstream_impact_types` - `[ImpactType!]`

List of ImpactType values representing the downstream business impacts reachable from this DataStore.

`downstream_impact_types_and_counts` - `[String!]`

List of downstream ImpactType values with their associated counts for this DataStore. Each element is formatted as `ImpactType,count`, e.g. `["RansomwareExposure,2", "SensitiveDataExposure,2"]`.

`impact_paths_count` - `Int!`

The number of impact paths (attack paths) this DataStore is included in. Alias of `attack_paths_count`.

`attack_paths_count` - `Int!`

The number of AttackPaths this DataStore is included in. Alias of `impact_paths_count`.

`weaknesses_count` - `Int!`

The number of Weaknesses affecting this DataStore.

`portal_url` - `String`

A direct URL to view this DataStore in the Horizon3.ai Portal.

Request Response

```
fragment DataStoreFragment on DataStore {
  uuid
  data_store_type
  created_at
  name
  name_and_account
  title
  ip
  port
  host_name
  service_type
  account
  address
  data_resources_count
  data_resources_label
  sensitive_resources_count
  is_authenticated
  permissions_list
  score
  severity
  base_score
  context_score
  sensitive_data_item_count
  downstream_impact_types
  attack_paths_count
  weaknesses_count
  op_id
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.6 DataStoresPage

Description

A paginated list of DataStore records. Returned by `Query.data_stores_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`data_stores` - `[DataStore!]!`
The list of DataStore records for the current page.

Request Response

```
fragment DataStoresPageFragment on DataStoresPage {
  page_info {
    page_num
    page_size
  }
  data_stores {
    uuid
    data_store_type
    name
    title
    ip
    port
    service_type
    data_resources_count
    sensitive_resources_count
    score
    severity
    op_id
    portal_url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Data Stores

8.23.7 InterestingResource

Description

An InterestingResource represents a file or web page flagged by NodeZero as potentially containing sensitive information during a pentest. Examples include configuration files with embedded credentials (AWS keys, SSH keys), exposed admin panels, and files containing PII or PCI data.

InterestingResources are typically found within a DataStore and may have associated Proof records (screenshots or command output) that document the finding.

Fields

`uuid` - `String!`
Unique identifier for this InterestingResource.
`created_at` - `Datetime!`
When this InterestingResource was discovered during the pentest (UTC).
`uri` - `String`
The URI or full file path of this InterestingResource (e.g., a web URL or file system path).
`op_id` - `String!`
The ID of the Op in which this InterestingResource was discovered.
`service_uuid` - `String`
The UUID of the Service where this InterestingResource was found.
`service` - `Service!`
The Service where this InterestingResource was found.
`data_store_uuid` - `String`
The UUID of the DataStore where this InterestingResource was found.
`data_store` - `DataStore`
The DataStore where this InterestingResource was found.
`proofs` - `[Proof]`
Proof records (screenshots or command output) documenting this InterestingResource as captured during the pentest.

Request Response

```
fragment InterestingResourceFragment on InterestingResource {
  uuid
  created_at
  uri
  service_uuid
  data_store_uuid
  op_id
}
```

Examples may not include all available fields for a type.

8.24 Pentesting: Hosts

Access information about discovered hosts including IP addresses, hostnames, operating systems, open ports, and their role in attack paths. Query the complete inventory of systems identified during pentesting.

8.24.1 `hosts_page`

Description

Returns a paginated list of in-scope Hosts discovered during the specified pentest. Supports filtering, sorting, and text search via `page_input`.

Response

Returns a `HostsPage!`

Arguments

`input` - `OpInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query HostsPage($input: OpInput!, $page_input: PageInput) {
  hosts_page(input: $input, page_input: $page_input) {
    hosts {
      ...HostPageFragment
    }
  }
}

fragment HostPageFragment on Host {
  uuid
  ip
  host_name
  is_in_scope
  is_domain_controller
  os_fingerprints
  score
  severity
  subnet
  is_public
  display_name
  weaknesses_count
  credentials_count
  services_count
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Hosts

8.24.2 `hosts_count`

Description

Returns the number of in-scope Hosts discovered during the specified pentest. Supports filtering via `page_input` (e.g., by `severity`, `is_domain_controller`, or text search).

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query HostsCount($input: OpInput!, $page_input: PageInput) {
  hosts_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Hosts

8.24.3 `host`

Description

Returns the Host with the given `uuid`. A Host represents a unique IP address (network endpoint) discovered during a pentest.

Response

Returns a `Host`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query Host($uuid: String!) {
  host(uuid: $uuid) {
    ...HostFragment
  }
}

fragment HostFragment on Host {
  uuid
  created_at
  ip
  mac
  host_name
  host_names
  is_in_scope
  is_domain_controller
  os_fingerprints
  op_id
  score
  severity
  subnet
  cloud_provider
  is_public
  display_name
  weaknesses_count
  credentials_count
  services_count
  attack_paths_count
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Hosts

8.24.4 `hosts_csv_url`

Description

Returns a temporary presigned URL to a CSV export of all Hosts discovered during the specified pentest. The URL expires after a short time. The CSV format is documented under HostCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query HostsCsvUrl($input: OpInput!) {
  hosts_csv_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Hosts

8.24.5 Host

Description

A Host represents a unique IP address discovered by NodeZero during a pentest. Each Host corresponds to a network endpoint such as a server, workstation, domain controller, cloud instance, printer, or IoT device.

Hosts serve as a central point for understanding the security posture of an individual machine -- including the Services it exposes, the Credentials that can access it, the DataStores it contains, and the Weaknesses that affect it.

Use `Query.hosts_page` to list Hosts for a pentest, `Query.host` to fetch a single Host by UUID, or `Query.hosts_csv_url` to export Hosts as CSV.

Fields

`uuid` - `String!`
Unique identifier for this Host.
`created_at` - `Datetime!`
When this Host was first discovered during the pentest (UTC).
`ip` - `String`
The IP address of this Host. Each Host record is uniquely identified by its IP address within a pentest.
`mac` - `String`
The MAC address of this Host, if detected.
`cname_chains` - `[String]`
CNAME chains associated with any DNS hostnames for this Host. Each element represents a chain of CNAME records.
`host_name` - `String`
The primary hostname for this Host. Resolved from LDAP or DNS hostnames, with preference given to LDAP when it matches a DNS hostname prefix.
`host_names` - `[String]`

All hostnames associated with this Host's IP address, aggregated from multiple sources:

- Internal and external DNS hostnames
- LDAP hostname
- Virtual host (VHost) names

`is_in_scope` - `Boolean`
`true` if this Host was included in the configured pentest scope.
`is_domain_controller` - `Boolean`
`true` if this Host is an Active Directory domain controller.
`is_database_server` - `Boolean`
`true` if this Host is a database server.
`is_load_balancer` - `Boolean`
`true` if this Host is a load balancer.
`is_mail_server` - `Boolean`
`true` if this Host is a mail server.
`is_vpn` - `Boolean`
`true` if this Host is a VPN gateway.
`is_waf` - `Boolean`
`true` if this Host is a Web Application Firewall (WAF).
`os_fingerprints` - `[String]`
Operating system names and fingerprints detected on this Host (e.g., `Windows Server 2019`, `Ubuntu 22.04`).
`hardware_fingerprints` - `[String]`
Hardware vendor and model fingerprints detected on this Host.
`device_fingerprints` - `[String]`
Device type fingerprints detected on this Host (e.g., printer, router, IoT device).

`op_id` - `String!`
The ID of the Op in which this Host was discovered.
`data_stores_count` - `Int!`
The number of DataStores (file shares, databases, buckets, etc.) found on this Host.
`data_stores_page` - `DataStoresPage!`
Paginated list of DataStores found on this Host.

Arguments

`page_input` - `PageInput`
`credentials_count` - `Int!`
The number of Credentials that accessed this Host during the pentest.
`credentials_page` - `CredentialsPage!`
Paginated list of Credentials that accessed this Host during the pentest.

Arguments

`page_input` - `PageInput`
`data_resources_count` - `Long`
The number of data resources (files and database records) found on this Host. Excludes web pages -- see `web_resources_count` for web page counts.
`web_resources_count` - `Long`
The number of web pages discovered on this Host.
`services_count` - `Int!`
The number of Services (open ports/protocols) found on this Host.
`context_score_description` - `String`
A human-readable explanation of the factors contributing to this Host's risk score.
`score` - `Float`
The overall risk score for this Host, rounded to one decimal place. Derived from the weaknesses, credentials, and data linked to this Host.
`severity` - `Severity`
The Severity level corresponding to `score` (e.g., `CRITICAL`, `HIGH`, `MEDIUM`, `LOW`, `INFO`).
`subnet` - `String`
The network subnet containing this Host's IP address (e.g., `192.168.1.0/24`).
`cloud_provider` - `String`
The cloud provider hosting this Host (e.g., `aws`, `gcp`, `azure`), if applicable.
`cloud_region` - `String`
The cloud region where this Host resides (e.g., `us-east-1`, `us-central1`), if applicable.
`cloud_arns` - `[String!]`
The Amazon Resource Names (ARNs) of any cloud resources associated with this Host.
`is_public` - `Boolean`
`true` if this Host has a publicly routable IP address.
`action_logs_count` - `Int!`
The number of ActionLogs (commands and actions run by NodeZero) recorded on this Host during the pentest.

Arguments

`page_input` - `PageInput`
`action_logs_page` - `ActionLogsPage!`
Paginated list of ActionLogs (commands and actions run by NodeZero) on this Host during the pentest.

Arguments

`page_input` - `PageInput`
`action_logs_csv_url` - `String`
A temporary presigned URL to a CSV export of all actions/commands run by NodeZero on this Host. The URL expires after a short time.
`downstream_impact_types` - `[ImpactType!]`
List of ImpactType values representing the downstream business impacts reachable from this Host.
`os_names` - `[String!]`

High-level operating system names detected on this Host (e.g., `Windows`, `Linux`, `macOS`). Derived from `os_fingerprints`.

`attack_paths_count` - `Int!`

The number of AttackPaths this Host is included in. Equivalent to the count of downstream business impacts reachable from this Host.

`attack_paths_page` - `AttackPathsPage!`

Paginated list of AttackPaths this Host is included in. Each attack path traces a sequence of exploits leading to a downstream business impact.

Arguments

`page_input` - `PageInput`

`display_name` - `String`

A human-friendly display name for this Host, typically combining the IP address and hostname.

`weaknesses_count` - `Int!`

The number of Weaknesses found on this Host.

Arguments

`page_input` - `PageInput`

`weaknesses_page` - `WeaknessesPage!`

Paginated list of Weaknesses found on this Host.

Arguments

`page_input` - `PageInput`

`op` - `Op!`

The Op in which this Host was discovered.

`portal_url` - `String`

A direct URL to view this Host in the Horizon3.ai Portal.

Request Response

```
fragment HostFragment on Host {
  uuid
  created_at
  ip
  mac
  cname_chains
  host_name
  host_names
  is_in_scope
  is_domain_controller
  is_database_server
  is_mail_server
  is_vpn
  is_waf
  os_fingerprints
  hardware_fingerprints
  device_fingerprints
  op_id
  data_stores_count
  credentials_count
  data_resources_count
  web_resources_count
  services_count
  context_score_description
  score
  severity
  subnet
  cloud_provider
  cloud_region
  cloud_arns
  is_public
  downstream_impact_types
  os_names
  attack_paths_count
  display_name
  weaknesses_count
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Hosts

8.24.6 HostsPage

Description

A paginated list of Host records. Returned by `Query.hosts_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`hosts` - `[Host!]!`
The list of Host records for the current page.

Request Response

```
fragment HostsPageFragment on HostsPage {
  page_info {
    page_num
    page_size
  }
  hosts {
    uuid
    ip
    host_name
    is_in_scope
    is_domain_controller
    os_fingerprints
    score
    severity
    subnet
    is_public
    display_name
    op_id
    portal_url
  }
}
```

Examples may not include all available fields for a type.

8.25 Pentesting: EDR

View EDR (Endpoint Detection and Response) events triggered during pentesting. Track which security controls detected NodeZero's activities to measure detection coverage and response effectiveness.

8.25.1 `edr_overview_hosts_csv_presigned_url`

Description

Generates a CSV export of hosts with endpoint detection and response (EDR) data from a pentest and returns a temporary presigned URL for downloading the file. Each row includes the host's risk score, IP address, detected EDR vendors, count of unblocked security control events, MITRE ATT&CK tactics observed, downstream impacts, and host classification flags (e.g., domain controller, database server, public-facing).

Use `page_input` to filter the exported hosts by any supported host field. Each unique filter combination produces a separately cached CSV file. The presigned URL expires after a short time.

The CSV format is documented under EDROverviewHostCSV.

Response

Returns a `String`

Arguments

`input` - `OpInput!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query EDROverviewHostsCSVPresignedUrl(
  $input: OpInput!
  $page_input: PageInput
) {
  edr_overview_hosts_csv_presigned_url(
    input: $input
    page_input: $page_input
  )
}
```

Examples may not include all available fields for a type.

8.26 Pentesting: External Domains

Query external domains discovered during pentesting including subdomains, DNS records, and associated IPs and services.

8.26.1 ExternalDomain

Description

Represents a DNS domain name discovered during an external attack surface assessment. Each ExternalDomain is associated with a pentest and captures the domain's DNS resolution data (IP addresses, CNAME records), cloud infrastructure details, certificate information, and relationships to hosts and web resources found during the operation. Domains are organized hierarchically: top-level domains are specified in the pentest configuration, and subdomains are discovered through DNS enumeration.

Fields

`uuid` - `String!`
Unique identifier for this ExternalDomain record.
`created_at` - `Datetime!`
Timestamp (UTC) when this domain was first discovered during the pentest.
`name` - `String!`
The fully qualified domain name (FQDN), such as `app.example.com`.
`cnames` - `[String]`
CNAME DNS records associated with this domain. Returns the full CNAME chain if available, otherwise the direct CNAME list.
`ips` - `[String]`
The IP addresses this domain resolves to via DNS.
`op_id` - `String!`
The ID of the pentest (Op) during which this domain was discovered.

Request Response

```
fragment ExternalDomainFragment on ExternalDomain {
  uuid
  created_at
  name
  cnames
  ips
  op_id
}
```

Examples may not include all available fields for a type.

8.27 Pentesting: LDAP Domains

Access Active Directory and LDAP domain information discovered during pentesting including domain controllers and configuration details.

8.27.1 LDAPDomain

Description

Represents an Active Directory LDAP domain discovered by NodeZero during a pentest. An LDAPDomain corresponds to a Windows Active Directory domain identified through LDAP enumeration. These domains are central to Active Directory attack paths -- they appear as affected entities on Weaknesses, are associated with Credentials and Users, and provide context for Active Directory password audit results.

LDAPDomains are not queryable directly. They are accessible as related objects on types such as Weakness (`affected_ldap_domain`) and User (`ldap_domain`).

Fields

`uuid` - `String!`
Unique identifier for this LDAPDomain.
`created_at` - `Datetime!`
When this LDAP domain was first discovered during the pentest (UTC).
`name` - `String!`
The fully qualified Active Directory domain name (e.g., `corp.example.com`).
`op_id` - `String!`
The ID of the pentest in which this LDAP domain was discovered.

Request Response

```
fragment LDAPDomainFragment on LDAPDomain {
  uuid
  created_at
  name
  op_id
}
```

Examples may not include all available fields for a type.

8.28 Pentesting: Cloud Resources

Query cloud resources discovered during pentesting including AWS, Azure, and GCP assets. View cloud infrastructure, permissions, misconfigurations, and their role in attack paths.

8.28.1 CloudResource

Description

Represents a cloud resource instance discovered by NodeZero during a pentest. A CloudResource is a specific infrastructure component within a cloud account, such as an AWS EC2 instance, an AWS RDS database, an AWS Lambda function, an Azure Virtual Machine, or a Google Cloud Compute Engine instance.

Each CloudResource belongs to a CloudAccount (the owning account) and is associated with a cloud service type (e.g., `aws_ec2` , `aws_rds`). CloudResources link to the underlying network Endpoint when an IP address is available, and may have associated data resources such as database records and files tracked via `data_resources_count` .

CloudResources are not queryable directly. They are accessible as related objects on Host (`cloud_resources`) and Weakness (`affected_cloud_resource`).

Fields

`uuid` - `String!`
Unique identifier for this CloudResource.
`cloud_service_name` - `String`
The cloud service type identifier for this CloudResource (e.g., `aws_ec2` , `aws_lambda` , `aws_rds`). See `cloud_service_h3_service_name` for the human-readable equivalent.
`ip` - `String`
The IP address of this CloudResource, resolved from the associated network endpoint.

`arn` - `String`
The provider-assigned unique identifier for this CloudResource. For AWS resources this is the ARN (Amazon Resource Name); for Azure resources this is the Azure Resource ID.
`data_resources_count` - `Long`
The number of data resources discovered on this CloudResource. Includes database records and files. Excludes web pages.
`op_id` - `String!`
The ID of the pentest in which this CloudResource was discovered.
`created_at` - `Datetime`
When this CloudResource was first discovered during the pentest (UTC).
`cloud_account_id` - `String`
The cloud provider's account identifier for the account that owns this CloudResource (e.g., an AWS Account ID). Resolved from the associated CloudAccount record.
`cloud_provider_name` - `String`
The cloud provider for this CloudResource. Possible values include `AWS`, `AZURE`, and `GCP`. Resolved from the associated CloudAccount record.

Request Response

```
fragment CloudResourceFragment on CloudResource {
  uuid
  cloud_service_name
  ip
  arn
  data_resources_count
  created_at
  cloud_account_id
  cloud_provider_name
  op_id
}
```

Examples may not include all available fields for a type.

8.29 Pentesting: Applications

Access information about applications discovered during pentesting including web applications, services, and software versions. View application vulnerabilities and exposure.

8.29.1 Application

Description

An Application represents a software application discovered by NodeZero running on a Service. Examples include web servers, Active Directory Certificate Services, database management interfaces, and other network-accessible applications.

Each Application is associated with a single Service and inherits the Service's IP address and port. The `name` field identifies the application type based on fingerprint data collected during the pentest.

Fields

`uuid` - `String!`
Unique identifier for this Application.
`created_at` - `Datetime!`
When this Application was first discovered during the pentest (UTC).
`service_uuid` - `String!`
The unique identifier of the Service this Application is running on.
`service` - `Service!`
The Service this Application is running on. Loaded from `service_uuid`.
`ip` - `String`
The IP address of the Service this Application is running on.
`port` - `Int!`
The port number of the Service this Application is running on.

`name` - `String!`
The name of this Application, derived from fingerprint data collected during the pentest.
`op_id` - `String!`
The identifier of the pentest (Op) in which this Application was discovered.

Request Response

```
fragment ApplicationFragment on Application {
  uuid
  created_at
  service_uuid
  ip
  port
  name
  op_id
}
```

Examples may not include all available fields for a type.

8.30 Pentesting: Services

Query network services discovered during pentesting including service types, ports, protocols, banners, and versions. Understand the service landscape of your tested environment.

8.30.1 Service

Description

A Service represents a listening network port discovered on a Host during a pentest. Each Service record corresponds to a unique combination of IP address and port.

Services expose the entry points that NodeZero probes during a pentest. Use the `fingerprints` field to identify what software is running on the port, and navigate to related Applications, DataStores, Credentials, and Weaknesses to understand the full security posture of each listening port.

Fields

`uuid` - `String!`
Unique identifier for this Service.
`created_at` - `Datetime!`
When this Service was first discovered during the pentest (UTC).
`ip` - `String!`
IP address of the Service. Falls back to the fully qualified domain name if the IP is unavailable.
`port` - `Int!`
The port number on which this Service is listening.
`protocol` - `String`
Transport protocol of this Service (e.g., `tcp`, `udp`).
`fingerprints` - `[String]`
Application fingerprints identifying the software running on this Service, derived from banner and probe analysis during the pentest.
`iana_service_name` - `String`
The IANA (Internet Assigned Numbers Authority) standard name for the protocol or application running on this port (e.g., `http`, `ssh`, `ms-sql-s`).
`host` - `Host`
The Host where this Service is running. Loaded from the underlying endpoint identifier.
`data_resources_count` - `Long`
The number of data resources (files and database records) found on this Service. Excludes web pages -- see `web_resources_count` for web page counts.
`web_resources_count` - `Long`
The number of web pages discovered on this Service.
`op_id` - `String!`
The identifier of the pentest (Op) in which this Service was discovered.

`context_score_description` - `String`
A human-readable explanation of how the context score was determined for this Service.
`score` - `Float`
The overall risk score for this Service, combining base and context factors. Rounded to one decimal place.
`severity` - `Severity`
The Severity level derived from the `score` (e.g., CRITICAL, HIGH, MEDIUM, LOW, INFO).

Request Response

```
fragment ServiceFragment on Service {
  uuid
  created_at
  ip
  port
  protocol
  fingerprints
  iana_service_name
  data_resources_count
  web_resources_count
  score
  severity
  context_score_description
  op_id
}
```

Examples may not include all available fields for a type.

8.31 Pentesting: Action Logs

Access detailed logs of every action NodeZero performed during pentesting including MITRE ATT&CK mappings, module execution details, timestamps, and outcomes. Action logs provide complete forensic traceability of all pentest activities.

8.31.1 `action_logs_page`

Description

Returns a paginated list of ActionLog records for a given pentest.

The action log is the complete record of all commands executed by NodeZero across all hosts during the pentest. Results are sorted by `start_time` in ascending order by default. Supports filtering by `module_id`, `endpoint_ip`, `exit_code`, `start_time`, and `end_time` via the `page_input` argument.

Response

Returns an `ActionLogsPage!`

Arguments

`input` - `OpInput!`
Pentest to get action logs for.
`page_input` - `PageInput`
Pagination of action logs.

Request Response Python
cURL h3-cli

```
query ActionLogsPage($input: OpInput!, $page_input: PageInput) {
  action_logs_page(input: $input, page_input: $page_input) {
    action_logs {
      ...ActionLogPageFragment
    }
  }
}

fragment ActionLogPageFragment on ActionLog {
  uuid
  start_time
  end_time
  endpoint_ip
  cmd
  module_id
  module_name
```

```
  exit_code
  target_h3_names
  op_id
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.2 `action_logs_count`

Description

Returns the total number of ActionLog records for a given pentest.

The action log is the complete record of all commands executed by NodeZero across all hosts during the pentest. Use the `page_input` argument to apply filters (e.g., by `module_id`, `endpoint_ip`, or time range) before counting.

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`
Pentest to get action logs for.
`page_input` - `PageInput`
Pagination of action logs.

Request	Response	Python
cURL	h3-cli	

```
query ActionLogsCount($input: OpInput!, $page_input: PageInput) {
  action_logs_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.3 `action_logs_csv_presigned_url`

Description

Returns a temporary, presigned URL for downloading the action log as a CSV file.

The action log is the complete record of all commands executed by NodeZero across all hosts during the pentest. The CSV is generated on demand and stored in S3. The presigned URL expires after a short time.

Response

Returns a `String`

Arguments

`input` - `OpInput!`

Request | Response | Python
cURL | h3-cli

```
query ActionLogsCsvPresignedUrl($input: OpInput!) {
  action_logs_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.4 ActionLog

Description

Represents a single action or command executed by NodeZero during a pentest. Each ActionLog entry records a specific command that was run against a target host, including timing information, the attack module that generated the command, and the associated MITRE ATT&CK mappings.

Use action_logs_page to retrieve a paginated list of action logs for a given pentest.

Fields

`uuid` - `String!`
Unique identifier for this action log entry.
`start_time` - `Datetime!`
Timestamp when the action started, in ISO 8601 format (UTC).
`endpoint_uuid` - `String`
Unique identifier of the host targeted by this action.
`endpoint_ip` - `String`
IP address of the host targeted by this action.
`end_time` - `Datetime!`
Timestamp when the action completed, in ISO 8601 format (UTC).
`cmd` - `String!`
The full command string executed by NodeZero on the target host.
`module_id` - `String`
Identifier of the attack module that executed this action. Resolves to the module name if available, otherwise falls back to the module ID.
`module_name` - `String`
Display name of the attack module that executed this action.
`module_description` - `String`
Human-readable description of the attack module that executed this action.
`module_meta` - `ModuleMeta`
Detailed metadata for the attack module that executed this action. See ModuleMeta for available fields.
`mitre_mappings` - `[MitreMapping]`
The MITRE ATT&CK tactic, technique, and subtechnique mappings associated with this action. See MitreMapping.
`target_h3_names` - `[String]`
List of asset and weakness names associated with this action. Includes the IP address, service, application, URL, or data store that was targeted by the command.
`exit_code` - `String!`
Exit code returned by the command. A value of `0` typically indicates success; non-zero values indicate an error or alternative outcome.
`op_id` - `String!`
Unique identifier of the pentest in which this action was executed.

Request Response

```
fragment ActionLogFragment on ActionLog {
  uuid
  start_time
  end_time
  endpoint_uuid
  endpoint_ip
  cmd
  module_id
  module_name
  module_description
  module_meta {
    id
    name
    description
  }
  mitre_mappings {
    mitre_tactic_id
    mitre_technique_id
    mitre_subtechnique_id
```

```
  }
  target_h3_names
  exit_code
  op_id
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.5 ActionLogsPage

Description

A paginated list of ActionLog records. Returned by action_logs_page.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`action_logs` - `[ActionLog!]!`
The list of ActionLog records for the current page.

Request Response

```
fragment ActionLogsPageFragment on ActionLogsPage {
  page_info {
    page_num
    page_size
  }
  action_logs {
    uuid
    start_time
    end_time
    endpoint_ip
    cmd
    module_id
    module_name
    exit_code
    target_h3_names
    op_id
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.6 MitreMapping

Description

Represents a specific combination of a MITRE ATT&CK Tactic, Technique, and optionally a Subtechnique. Each MitreMapping links a NodeZero attack module to the adversary behaviors it emulates, as defined by the MITRE ATT&CK framework. MitreMappings are associated with attack modules via ModuleMeta.

Fields

`mitre_tactic_id` - `String!`
The MITRE ATT&CK Tactic identifier (e.g., `TA0043`).
`mitre_tactic` - `MitreTactic`
The full MitreTactic object, including name, description, and documentation URL.
`mitre_technique_id` - `String!`
The MITRE ATT&CK Technique identifier (e.g., `T1595`).
`mitre_technique` - `MitreTechnique`
The full MitreTechnique object, including name, description, and documentation URL.
`mitre_subtechnique_id` - `String`
The MITRE ATT&CK Subtechnique identifier (e.g., `T1595.001`), if applicable. Null when the mapping is at the technique level only.
`mitre_subtechnique` - `MitreSubtechnique`
The full MitreSubtechnique object, if applicable. Null when the mapping is at the technique level only.

Request Response

```
fragment MitreMappingFragment on MitreMapping {
  mitre_tactic_id
  mitre_tactic {
    id
    name
    url
  }
  mitre_technique_id
  mitre_technique {
    id
    name
    url
  }
  mitre_subtechnique_id
  mitre_subtechnique {
    id
    name
    url
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.7 MitreSubtechnique

Description

Represents a MITRE ATT&CK Subtechnique. Subtechniques provide a more granular description of adversary behavior within a parent MitreTechnique. For example, "Scanning IP Blocks" is a subtechnique of "Active Scanning."

Fields

`id` - `String!`
The MITRE ATT&CK Subtechnique identifier.
`mitre_technique_id` - `String`
The MITRE ATT&CK Technique identifier of the parent technique for this subtechnique.
`name` - `String`
The display name of the MITRE ATT&CK Subtechnique.
`description` - `String`
Description of the MITRE ATT&CK Subtechnique explaining the specific adversary behavior.
`url` - `String`
URL to the official MITRE ATT&CK documentation page for this Subtechnique.

Request Response

```
fragment MitreSubtechniqueFragment on MitreSubtechnique {
  id
  mitre_technique_id
  name
  description
  url
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.8 MitreTactic

Description

Represents a MITRE ATT&CK Tactic. Tactics represent the adversary's tactical objective -- the reason for performing an action (e.g., Reconnaissance, Initial Access, Lateral Movement). Tactics are the columns in the MITRE ATT&CK Matrix.

Fields

`id` - `String!`
The MITRE ATT&CK Tactic identifier.
`name` - `String`
The display name of the MITRE ATT&CK Tactic.
`description` - `String`
Description of the MITRE ATT&CK Tactic explaining the adversary's objective.
`url` - `String`
URL to the official MITRE ATT&CK documentation page for this Tactic.

Request Response

```
fragment MitreTacticFragment on MitreTactic {
  id
  name
  description
  url
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.9 MitreTechnique

Description

Represents a MITRE ATT&CK Technique. Techniques describe the specific method an adversary uses to achieve a tactical objective (e.g., Active Scanning, Brute Force). A technique belongs to one or more MitreTactics and may have more granular MitreSubtechniques.

Fields

`id` - `String!`
The MITRE ATT&CK Technique identifier.
`name` - `String`
The display name of the MITRE ATT&CK Technique.
`description` - `String`
Description of the MITRE ATT&CK Technique explaining the adversary method.
`url` - `String`
URL to the official MITRE ATT&CK documentation page for this Technique.

Request Response

```
fragment MitreTechniqueFragment on MitreTechnique {
  id
  name
  description
  url
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.10 ModuleInstance

Description

Represents a specific execution of a NodeZero attack module during a pentest. Each ModuleInstance captures the lifecycle of one module run, including when it started and completed, whether it succeeded, and the commands it executed.

Use `ModuleInstance.action_logs` to retrieve the individual commands executed by this module instance.

Fields

`uuid` - `String!`
Unique identifier for this module instance.
`created_at` - `Datetime!`
Timestamp when the module instance was created. Note: this reflects when the module was queued, not when it began execution. See `started_at` for the actual execution start time.
`module_id` - `String!`
Identifier of the attack module. Corresponds to ModuleMeta.id.
`options` - `String`
Command-line options passed to this module instance at execution time.
`started_at` - `Datetime!`
Timestamp when the module began executing.
`ended_at` - `Datetime!`
Timestamp when the module finished executing.
`success_flag` - `Boolean`
Indicates whether the module completed execution successfully. `true` if the module ran to completion without errors.
`error_msg` - `String`
Error message describing why the module failed, if applicable. Null when the module succeeded.
`op_id` - `String!`
Unique identifier of the pentest in which this module instance executed.
`name` - `String`
Display name of the attack module. Equivalent to ModuleMeta.name.
`description` - `String`
Human-readable description of the attack module. Equivalent to ModuleMeta.description.
`action_logs_count` - `Int`
Total number of ActionLog entries (commands) executed by this module instance.
`action_logs` - `[ActionLog]`
List of ActionLog entries representing each command executed by this module instance.
`module_meta` - `ModuleMeta`
Detailed metadata for the attack module, including MITRE ATT&CK mappings. See ModuleMeta.

Request Response

```
fragment ModuleInstanceFragment on ModuleInstance {
  uuid
  module_id
  name
  description
  created_at
  started_at
  ended_at
  success_flag
  error_msg
  options
  op_id
  action_logs_count
  module_meta {
    id
    name
    description
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Action Logs

8.31.11 ModuleMeta

Description

Static metadata describing a NodeZero attack module. Each attack module performs a specific offensive technique (such as host discovery, credential brute-forcing, or privilege escalation) and is mapped to one or more MITRE ATT&CK tactics, techniques, and subtechniques.

ModuleMeta is referenced by ActionLog and ModuleInstance to provide context about which module generated a given action or execution.

Fields

`id` - `String!`
Unique identifier for the attack module.
`name` - `String`
Display name of the attack module.
`description` - `String`
Human-readable description of what the attack module does.
`mitre_mappings` - `[MitreMapping]`
List of MITRE ATT&CK tactic, technique, and subtechnique mappings associated with this attack module. See MitreMapping.

Request Response

```
fragment ModuleMetaFragment on ModuleMeta {
  id
  name
  description
  mitre_mappings {
    mitre_tactic_id
    mitre_technique_id
    mitre_subtechnique_id
  }
}
```

Examples may not include all available fields for a type.

8.32 Pentesting: Action Log V2

Access NodeZero pentest actions normalized into the OCSF (Open Cybersecurity Schema Framework) event schema. Query a paginated event stream and count events for a pentest.

8.32.1 `action_log_v2_events_page`

Description

Returns a paginated list of ActionLogV2Event records for a given pentest. Supports filtering via PageInput.

Response

Returns an `ActionLogV2EventsPage!`

Arguments

`input` - `OpInput!`
Pentest to get events for.
`page_input` - `PageInput`
Pagination and filtering.

Request Response Python
cURL h3-cli

```
query ActionLogV2EventsPage($input: OpInput!, $page_input: PageInput) {
  action_log_v2_events_page(input: $input, page_input: $page_input) {
```

```
    page_info {
      page_num
      page_size
    }
    events {
      ...ActionLogV2EventFragment
    }
  }
}

fragment ActionLogV2EventFragment on ActionLogV2Event {
  uid
  op_id
  time
  class_uid
  activity_id
  severity_id
  data
}
```

Examples may not include all available fields for a type.

Pentesting: Action Log V2

8.32.2 `action_log_v2_events_count`

Description

Returns the total number of ActionLogV2Event records for a given pentest. Supports filtering via PageInput.

Response

Returns an `Int!`

Arguments

`input` - `OpInput!`
Pentest to count events for.
`page_input` - `PageInput`
Pagination and filtering.

Request	Response	Python
cURL	h3-cli	

```
query ActionLogV2EventsCount($input: OpInput!, $page_input: PageInput) {
  action_log_v2_events_count(input: $input, page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Action Log V2

8.32.3 ActionLogV2Event

Description

An OCSF (Open Cybersecurity Schema Framework) event recorded during a pentest. Each event captures a single action taken by NodeZero, normalized into the OCSF event schema.

Use action_log_v2_events_page to retrieve a paginated list of events for a given pentest.

Fields

`uid` - `ID!`
Unique identifier for this event.
`op_id` - `ID!`
Unique identifier of the pentest associated with this event.
`time` - `Datetime!`
Time the event occurred.
`class_uid` - `Int!`
OCSF event class identifier.
`activity_id` - `Int!`
OCSF activity identifier within the event class.
`severity_id` - `Int`
OCSF severity level identifier. Null when the source omits it.
`data` - `JSONObject!`
The full OCSF event payload as a JSON object.

Request Response

```
fragment ActionLogV2EventFragment on ActionLogV2Event {
  uid
  op_id
  time
  class_uid
  activity_id
  severity_id
  data
}
```

Examples may not include all available fields for a type.

Pentesting: Action Log V2

8.32.4 ActionLogV2EventsPage

Description

A paginated list of ActionLogV2Event records. Returned by action_log_v2_events_page.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`events` - `[ActionLogV2Event!]!`
The list of ActionLogV2Event records for the current page.

Request Response

```
fragment ActionLogV2EventsPageFragment on ActionLogV2EventsPage {
  page_info {
    page_num
    page_size
  }
  events {
    uid
    op_id
    time
    class_uid
    activity_id
    severity_id
    data
  }
}
```

Examples may not include all available fields for a type.

8.33 Pentesting: Op Compare

Compare weaknesses between two pentests to identify new, resolved, and persistent vulnerabilities. Track security posture changes over time and measure remediation effectiveness.

8.33.1 `weaknesses_diff_page`

Description

Returns a paginated list of Weakness records that differ between two pentests.

Only weaknesses that are present in one pentest but not the other are included. Each Weakness record in the result includes a `diff_status` field set to either `ADDED` or `REMOVED`.

A weakness with `diff_status: REMOVED` was present in the first pentest (OpDiffInput.op_id_1) but not the second. A weakness with `diff_status: ADDED` was present in the second pentest (OpDiffInput.op_id_2) but not the first.

If the diff data has not yet been computed for the given pair of pentests, it will be generated on demand before returning results.

Response

Returns a `WeaknessesDiffPage!`

Arguments

`op_diff_input` - `OpDiffInput!`
`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query WeaknessesDiffPage(
  $op_diff_input: OpDiffInput!,
  $page_input: PageInput
```

```
) {
  weaknesses_diff_page(
    op_diff_input: $op_diff_input,
    page_input: $page_input
  ) {
    weaknesses {
      ...WeaknessDiffFragment
    }
  }
}

fragment WeaknessDiffFragment on Weakness {
  uuid
  vuln_id
  vuln_name
  vuln_short_name
  op_id
  ip
  severity
  score
  diff_status
  affected_asset_display_name
  has_proof
  portal_url
}
```

Examples may not include all available fields for a type.

Pentesting: Op Compare

8.33.2 `weaknesses_diff_stats`

Description

Returns aggregate statistics summarizing the differences in weaknesses between two pentests, including the count of added, removed, and unchanged weaknesses. See weaknesses_diff_page for the full paginated diff.

If the diff data has not yet been computed for the given pair of pentests, it will be generated on demand before returning results.

Response

Returns a `DiffStats!`

Arguments

`op_diff_input` - `OpDiffInput!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query WeaknessesDiffStats(
  $op_diff_input: OpDiffInput!,
  $page_input: PageInput
) {
  weaknesses_diff_stats(
    op_diff_input: $op_diff_input,
    page_input: $page_input
  ) {
    added_count
    removed_count
    unchanged_count
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Op Compare

8.33.3 `weaknesses_diff_csv_url`

Description

Returns a temporary, presigned URL for downloading a CSV file containing the diff of weaknesses between two pentests. The CSV is generated on demand and the presigned URL expires after a short time. See weaknesses_diff_page for more details on how the diff is computed.

Response

Returns a `String`

Arguments

`op_diff_input` - `OpDiffInput!`

Request | Response | Python
cURL | h3-cli

```
query WeaknessesDiffCsvUrl($op_diff_input: OpDiffInput!) {
  weaknesses_diff_csv_url(op_diff_input: $op_diff_input)
}
```

Examples may not include all available fields for a type.

Pentesting: Op Compare

8.33.4 DiffStats

Description

Aggregate statistics summarizing the differences between two pentests. Returned by weaknesses_diff_stats.

Fields

`added_count` - `Int`
The number of weaknesses that were newly discovered in the second pentest (present in `op_id_2` but not in `op_id_1`).
`removed_count` - `Int`
The number of weaknesses that were resolved or no longer detected in the second pentest (present in `op_id_1` but not in `op_id_2`).
`unchanged_count` - `Int`
The number of weaknesses that are identical between the two pentests being compared.

Request Response

```
fragment DiffStatsFragment on DiffStats {
  added_count
  removed_count
  unchanged_count
}
```

Examples may not include all available fields for a type.

Pentesting: Op Compare

8.33.5 DiffStatus

Description

Indicates whether a record was added or removed when comparing two pentests. Used in the weaknesses_diff_page results where each Weakness record includes a `diff_status` field set to one of these values.

Values

`ADDED`
The weakness was NOT present in the first pentest (`op_id_1`) but IS present in the second pentest (`op_id_2`), indicating it is a newly discovered weakness.
`REMOVED`
The weakness was present in the first pentest (`op_id_1`) but is NOT present in the second pentest (`op_id_2`), indicating it has been resolved or is no longer detected.

Example

```
"ADDED"
```

Pentesting: Op Compare

8.33.6 OpDiffInput

Description

Input type specifying the two pentests to compare. Used by weaknesses_diff_page, weaknesses_diff_stats, weaknesses_diff_csv_url, and other diff-related APIs.

Fields

`op_id_1` - `String!`
Unique identifier of the first (typically older) pentest in the comparison. Weaknesses present in this pentest but not in the second will have a DiffStatus of `REMOVED`.

`op_id_2` - `String!`
Unique identifier of the second (typically newer) pentest in the comparison. Weaknesses present in this pentest but not in the first will have a DiffStatus of `ADDED`.

Request

```
# Example 1: Compare weaknesses between two pentests
query WeaknessesDiff($op_diff_input: OpDiffInput! = {
    op_id_1: "1234abcd-1234-abcd-1234-abcd1234abcd",
    op_id_2: "5678abcd-5678-abcd-5678-abcd5678abcd"
}, $page_input: PageInput) {
  weaknesses_diff_page(
    op_diff_input: $op_diff_input,
    page_input: $page_input
  ) {
    weaknesses {
      uuid
      vuln_id
      vuln_name
      diff_status
      severity
      score
    }
  }
}

# Example 2: Get diff stats between two pentests
query WeaknessesDiffStats($op_diff_input: OpDiffInput! = {
    op_id_1: "1234abcd-1234-abcd-1234-abcd1234abcd",
    op_id_2: "5678abcd-5678-abcd-5678-abcd5678abcd"
}) {
  weaknesses_diff_stats(op_diff_input: $op_diff_input) {
    added_count
    removed_count
    unchanged_count
  }
}
```

Examples may not include all available fields for a type.

Pentesting: Op Compare

8.33.7 WeaknessesDiffPage

Description

A paginated list of Weakness records representing differences between two pentests. Returned by weaknesses_diff_page. Only weaknesses that differ between the two pentests are included; weaknesses common to both are excluded.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`weaknesses` - `[Weakness!]!`
The list of Weakness records for the current page. Each record includes a `diff_status` field set to ADDED or REMOVED indicating whether the weakness appeared in only the second or first pentest, respectively.

Request Response

```
fragment WeaknessesDiffPageFragment on WeaknessesDiffPage {
  page_info {
    page_num
    page_size
  }
  weaknesses {
    uuid
    vuln_id
    vuln_name
    vuln_short_name
    op_id
    ip
    severity
    score
    diff_status
    affected_asset_display_name
    has_proof
    portal_url
  }
}
```

Examples may not include all available fields for a type.

8.34 Vuln Management Hub

Track vulnerability trends and manage remediation over time. Access weakness series for historical vulnerability data, query assets across your environment, organize findings through custom perspectives, and run focused validation campaigns. The Vuln Management Hub provides a comprehensive view of your security posture evolution.

8.35 Vuln Management Hub: Weakness Series

Track vulnerability trends over time with weakness series that aggregate findings across multiple pentests. View historical data, apply annotations and tags for tracking, manage tickets, and analyze remediation progress for persistent vulnerabilities.

8.35.1 `weakness_series_page`

Description

Returns a paginated list of WeaknessSeries records for the current client account.

A weakness series tracks a single weakness on a single asset across multiple pentests over time, enabling vulnerability management workflows such as tracking remediation progress and identifying regressions.

Use `page_input` to filter by severity, status, vulnerability type, and other fields. Use `as_of_date` to view the state of weakness series as of a specific date.

Response

Returns a `WeaknessSeriesPage!`

Arguments

`page_input` - `PageInput!`

`as_of_date` - `Date`

Optional date in YYYY-MM-DD format specifying the point-in-time view for the weakness series data. Defaults to today.

Request | Response | Python
cURL | h3-cli

```
query WeaknessSeriesPage($page_input: PageInput!, $as_of_date: Date) {
  weakness_series_page(
    page_input: $page_input
    as_of_date: $as_of_date
  ) {
    weakness_series {
      ...WeaknessSeriesFragment
    }
  }
}

fragment WeaknessSeriesFragment on WeaknessSeries {
  name
  vuln_id
  vuln_name
  vuln_short_name
  vuln_category
  severity
  status
  score
  annotation_status
  is_one_click_verify
  affected_asset_name
  ip
  first_seen_date
  last_seen_date
  weakness_series_found_count
  attack_paths_count
  total_impacts_count
  business_risks
  last_found_op_types
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.2 `weakness_series_count`

Description

Returns the total number of WeaknessSeries records matching the given filters for the current client account.

Use `page_input` to apply filters before counting.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput!`

`as_of_date` - `Date`

Optional date in YYYY-MM-DD format specifying the point-in-time view for the weakness series data. Defaults to today.

Request | Response | Python

cURL | h3-cli

```
query WeaknessSeriesCount($page_input: PageInput!, $as_of_date: Date) {
  weakness_series_count(
    page_input: $page_input
    as_of_date: $as_of_date
  )
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.3 `weakness_series_annotations_page`

Description

Returns a paginated list of WeaknessSeriesAnnotations for the current client account. Annotations capture user-applied status updates, notes, and tags on weakness series records.

Response

Returns a `WeaknessSeriesAnnotationsPage!`

Arguments

`page_input` - `PageInput!`

Request | Response | Python
cURL | h3-cli

```
query WeaknessSeriesAnnotationsPage($page_input: PageInput!) {
  weakness_series_annotations_page(page_input: $page_input) {
    annotations {
      ...WeaknessSeriesAnnotationFragment
    }
  }
}

fragment WeaknessSeriesAnnotationFragment on WeaknessSeriesAnnotation {
  uuid
  row_created_at
  row_updated_at
  weakness_series_uuid
  annotation_uuid
  annotation_status
  annotation_text
  affected_asset_short_name
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.4 `weakness_series_annotations_count`

Description

Returns the total number of WeaknessSeriesAnnotations for the current client account.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput!`

Request | Response | Python
cURL | h3-cli

```
query WeaknessSeriesAnnotationsCount($page_input: PageInput!) {
  weakness_series_annotations_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.5 `apply_annotations_to_weakness_series`

Description

Applies one or more Annotations to WeaknessSeries records. Each AnnotationInput specifies the target weakness series UUID, status, text, and optional tags.

If an `annotation_uuid` is provided in the input, the existing annotation is updated; otherwise, a new annotation is created. Each weakness series can have at most one annotation. Returns the count and list of annotations that were applied.

Response

Returns an `ApplyAnnotationOutput!`

Arguments

`inputs` - `[AnnotationInput!]!`

Request | Response | Python

cURL | h3-cli

```
mutation ApplyAnnotationsToWeaknessSeries(
  $inputs: [AnnotationInput!]!
) {
  apply_annotations_to_weakness_series(inputs: $inputs) {
    annotations_count
    annotations {
      uuid
      target_entity_uuid
      target_entity_type
      text
      status
      risk_accepted
      is_suppressed
    }
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.6 `remove_annotations_from_weakness_series`

Description

Removes all annotations from the specified WeaknessSeries records. Returns `true` if at least one annotation was successfully removed, `false` if no annotations were found for the given UUIDs.

Response

Returns a `Boolean!`

Arguments

`weakness_series_uuids` - `[String!]!`

Request | Response | Python
cURL | h3-cli

```
mutation RemoveAnnotationsFromWeaknessSeries(
  $weakness_series_uuids: [String!]!
) {
  remove_annotations_from_weakness_series(
    weakness_series_uuids: $weakness_series_uuids
  )
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.7 `tickets_csv_presigned_url`

Description

Returns a temporary, presigned URL for downloading a CSV file containing all tickets for the current client account across all pentests. The presigned URL expires after a short time.

The CSV format is documented under TicketCSV.

Response

Returns a `String`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query TicketsCsvPresignedUrl($page_input: PageInput) {
  tickets_csv_presigned_url(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.8 `weakness_series_metadata`

Description

Returns a WeaknessSeriesMetadata object with metadata for the weakness series data.

Response

Returns a `WeaknessSeriesMetadata!`

Request | Response | Python
cURL | h3-cli

```
query WeaknessSeriesMetadata {
  weakness_series_metadata {
    unprocessed_ops_count
    last_processed_op_id
    unprocessed_ops {
      op_id
      op_name
      op_type
      op_state
      completed_at
    }
    last_processed_op {
      op_id
      op_name
      op_type
      op_state
      completed_at
    }
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.9 `weakness_series_facets`

Description

Returns the available filter facets for WeaknessSeries queries. Each facet provides a list of distinct values and their counts in the current dataset, enabling construction of filtered list UIs.

Response

Returns a `WeaknessSeriesFacets!`

Arguments

`page_input` - `PageInput`

`as_of_date` - `Date`

Optional date in YYYY-MM-DD format specifying the point-in-time view for the facet data. Defaults to today.

Request | Response | Python

cURL | h3-cli

```
query WeaknessSeriesFacets($page_input: PageInput, $as_of_date: Date) {
  weakness_series_facets(
    page_input: $page_input
    as_of_date: $as_of_date
  ) {
    vuln_short_name {
      label
      value
      value_count
    }
    op_template_name {
      label
      value
      value_count
    }
    business_risks {
      label
      value
      value_count
    }
    vuln_threat_actor_names {
      label
      value
      value_count
    }
    downstream_impact_types {
      label
      value
      value_count
    }
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.10 `weakness_series_csv_presigned_url`

Description

Returns a temporary, presigned URL for downloading the WeaknessSeries data as a CSV file. The CSV is generated on demand and the presigned URL expires after a short time. Use `page_input` to apply filters to the exported data.

Response

Returns a `String`

Arguments

`page_input` - `PageInput!`

`as_of_date` - `Date`

Optional date in YYYY-MM-DD format specifying the point-in-time view for the exported data. Defaults to today.

Request | Response | Python

cURL | h3-cli

```
query WeaknessSeriesCsvPresignedUrl(
  $page_input: PageInput!
  $as_of_date: Date
) {
  weakness_series_csv_presigned_url(
    page_input: $page_input
    as_of_date: $as_of_date
  )
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.11 `weakness_series_downstream_impacts_count`

Description

Returns the total number of downstream impacts for a given set of WeaknessSeries UUIDs. Downstream impacts represent the systems and data that could be compromised if this weakness is exploited.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput!`

`weakness_series_uuids` - `[String!]`

Request | Response | Python

cURL | h3-cli

```
query WeaknessSeriesDownstreamImpactsCount(
  $page_input: PageInput!
  $weakness_series_uuids: [String!]
) {
  weakness_series_downstream_impacts_count(
    page_input: $page_input
    weakness_series_uuids: $weakness_series_uuids
  )
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.12 `schedule_weakness_series_1cv_ops`

Description

Creates one or more 1CV (One-Click-Verify) pentests for a set of WeaknessSeries UUIDs.

A 1CV pentest is a targeted re-pentest that verifies whether a specific weakness has been remediated. The pentest configuration is derived automatically from the most recent pentest in which each weakness was discovered — including its scope, runner, and attack parameters — so no manual setup is required. Weaknesses from the same original pentest are grouped into a single 1CV op.

Returns the scheduled pentests and any failures via ScheduleWeaknessSeries1CVOpsOutput.

Response

Returns a `ScheduleWeaknessSeries1CVOpsOutput!`

Arguments

`input` - `ScheduleWeaknessSeries1CVOpsInput!`

Request | Response | Python
cURL | h3-cli

```
mutation ScheduleWeaknessSeries1CVOps($input: ScheduleWeaknessSeries1CVOpsInput!) {
  schedule_weakness_series_1cv_ops(input: $input) {
    ops_count
    ops {
      op {
        ...OpFragment
      }
    }
    failures {
      reason
      for_weakness_series_uuids
    }
  }
}

fragment OpFragment on Op {
  op_id
  op_name
  op_type
  op_state
  op_param_max_scope
  runner_name
  nodezero_script_url
  one_click_verify_data {
    op_id
    vuln_ids
    weakness_uuids
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.13 WeaknessSeries

Description

A WeaknessSeries tracks a single weakness on a single asset across multiple pentests over time.

It aggregates weakness findings across an entire series of pentests (an OpSeries), enabling vulnerability management workflows such as monitoring remediation progress, identifying regressions, and applying Annotations to track status.

Use weakness_series_page to retrieve a paginated list of weakness series records.

Fields

`name` - `String!`
Display name for this weakness series, typically derived from the vulnerability name and affected asset.
`last_weakness_uuid` - `String!`
Unique identifier of the most recent weakness instance in this series. Can be used to retrieve the latest weakness details.
`first_found_at` - `Date!`
Timestamp (date and time) when the weakness was first discovered in this series. Deprecated in favor of `first_seen_at`. Use `first_seen_at`.
`first_seen_at` - `Datetime!`
Timestamp (date and time, in ISO 8601 format) when the weakness was first discovered in this series.
`first_seen_date` - `Date!`
Date (without time, in YYYY-MM-DD format) when the weakness was first discovered in this series.
`first_found_op_id` - `String!`
Unique identifier of the pentest in which this weakness was first discovered.
`first_found_op_type` - `String!`
The type of pentest (e.g., Internal, External) in which this weakness was first discovered.
`last_found_at` - `Date!`
Timestamp (date and time) when the weakness was most recently detected. Deprecated in favor of `last_seen_at`. Use `last_seen_at`.
`last_seen_at` - `Datetime!`
Timestamp (date and time, in ISO 8601 format) when the weakness was most recently detected in a pentest.
`last_seen_date` - `Date!`
Date (without time, in YYYY-MM-DD format) when the weakness was most recently detected.
`vuln_id` - `String`
The vulnerability identifier associated with this weakness series (e.g., CVE ID or internal vulnerability ID).
`vuln_name` - `String`
The full name of the vulnerability associated with this weakness series.
`vuln_short_name` - `String`
The abbreviated name of the vulnerability associated with this weakness series.
`vuln_category` - `String`
The category of the vulnerability (e.g., credential, configuration, software) associated with this weakness series.
`vuln_aliases` - `[String!]`
Well-known aliases for this vulnerability (e.g., alternate CVE IDs or common names).
`weakness_series_found_count` - `Int!`
The number of pentests in which this weakness was detected.
`status` - `WeaknessSeriesOpStatus`
Current status of this weakness series. See WeaknessSeriesOpStatus for possible values (OPEN, MITIGATED, REGRESSED, NO_LONGER_FOUND).
`severity` - `Severity`
The severity level of this weakness series as determined by the most recent pentest (e.g., CRITICAL, HIGH, MEDIUM, LOW, INFO).
`is_one_click_verify` - `Boolean`

Indicates whether this weakness series is eligible for one-click verification, which allows running a targeted re-pentest to confirm remediation.

`affected_asset_name` - `String`

Full name of the asset directly affected by this weakness series (e.g., IP address with port and service name).

`ip` - `String`

IP address of the asset directly affected by this weakness series, if available.

`port` - `Int`

Port number of the affected service, if available.

`protocol` - `String`

Network protocol of the affected service (e.g., `tcp`, `udp`), if available.

`last_found_op_id` - `String!`

Unique identifier of the pentest in which this weakness was most recently detected.

`last_pentested_op_id` - `String!`

Unique identifier of the most recent pentest that included the affected asset in its scope, regardless of whether the weakness was found.

`annotation` - `Annotation`

The user-applied Annotation for this weakness series, if one has been set. Annotations track remediation status, risk acceptance, and notes.

`annotation_status` - `AnnotationStatus`

The AnnotationStatus of this weakness series (e.g., FIXED, TO_DO, RISK_ACCEPTED), if an annotation has been applied.

`min_time_to_finding_s` - `Int`

The minimum time (in seconds) it took NodeZero to discover this weakness during any pentest in the series.

`min_time_to_finding_hms` - `String`

The minimum time to discovery formatted as `HH:MM:SS`.

`attack_paths_count` - `Int!`

The number of distinct attack paths that lead to this weakness.

`credentials_count` - `Int!`

The number of compromised credentials associated with this weakness.

`impacts_count` - `Int!`

The number of downstream impacts for this weakness series. No longer supported

`critical_impacts_count` - `Int!`

The number of critical downstream impacts reachable from this weakness (e.g., ransomware exposure, sensitive data exposure).

`critical_impacts_percentage` - `Float!`

The percentage of downstream impacts that are classified as critical.

`total_impacts_count` - `Int!`

The total number of downstream impacts reachable from this weakness, including both critical and non-critical impacts.

`severity_ranking` - `Int!`

Numeric ranking used for sorting by severity. Lower values indicate higher severity.

`status_ranking` - `Int!`

Numeric ranking used for sorting by WeaknessSeriesOpStatus. Lower values indicate higher priority statuses.

`annotation_status_ranking` - `Int`

Numeric ranking used for sorting by AnnotationStatus. Null if no annotation has been applied.

`score` - `Float`

Composite risk score for this weakness series, incorporating severity and downstream impact. Rounded to one decimal place.

`vuln_threat_actor_names` - `[String!]!`

List of known threat actor names associated with the vulnerability, derived from threat intelligence data.

`business_risks` - `[String!]!`

List of business risk categories associated with this weakness (e.g., `Data Breach`, `Ransomware`).

`downstream_impact_types_and_counts` - `[String!]`

List of downstream impact types and their counts reachable from this weakness. Each array element is formatted as `ImpactType,count` (e.g., `["RansomwareExposure,2", "SensitiveDataExposure,2"]`). The downstream impacts and counts are from the most recent pentest containing this weakness.

`last_found_op_types` - `[OpType!]`

The pentest types in which this weakness was most recently found. Returns a single value for ungrouped queries or multiple unique values for grouped queries.

`last_mitigated_at` - `Datetime`

Timestamp (date and time) when the weakness series was last mitigated.

`last_mitigated_op_id` - `String`

Unique ID for the Op when the weakness series was last mitigated.

Request Response

```
fragment WeaknessSeriesFragment on WeaknessSeries {
  name
  vuln_id
  vuln_name
  vuln_short_name
  vuln_category
  severity
  status
  score
  annotation_status
  is_one_click_verify
  affected_asset_name
  ip
  first_found_at
  last_found_at
  first_seen_date
  last_seen_date
  first_found_op_id
  first_found_op_type
  last_found_op_id
  last_pentested_op_id
  last_weakness_uuid
  weakness_series_found_count
  attack_paths_count
  credentials_count
  total_impacts_count
  critical_impacts_count
  critical_impacts_percentage
  min_time_to_finding_hms
  business_risks
  last_found_op_types
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.14 Annotation

Description

Represents a user-applied annotation on a WeaknessSeries record. Annotations track the remediation lifecycle of a weakness, including status (e.g., FIXED, TO_DO, RISK_ACCEPTED), free-text notes, risk suppression dates, and user-defined Tags.

Use apply_annotations_to_weakness_series to create or update annotations.

Fields

`uuid` - `ID!`
Unique identifier of this annotation.
`target_entity_uuid` - `StringNotEmpty!`
Unique identifier of the target entity this annotation is applied to (typically a WeaknessSeries UUID).
`text` - `StringNotEmpty`
Free-text note providing additional context about the remediation status or risk decision.
`status` - `AnnotationStatus`
The current remediation or review status of this annotation. See AnnotationStatus for possible values.
`risk_accepted` - `Boolean!`
Indicates whether this annotation marks the associated risk as explicitly accepted.
`suppress_until` - `Date`
Date until which risk notifications are suppressed for this weakness. Null if no suppression is active.
`is_suppressed` - `Boolean!`
Indicates whether risk notifications are currently suppressed for this weakness based on the `suppress_until` date.
`tags` - `[Tag!]!`
List of user-defined Tags applied to this annotation for categorization and filtering.

Request Response

```
fragment AnnotationFragment on Annotation {
  uuid
  target_entity_uuid
  target_entity_type
  text
  status
  risk_accepted
  suppress_until
  is_suppressed
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.15 AnnotationInput

Fields

`annotation_uuid` - `StringNotEmpty`
Unique identifier of an existing annotation to update. If omitted, a new annotation will be created.
`target_entity_uuid` - `StringNotEmpty!`
Unique identifier of the WeaknessSeries to annotate.
`text` - `StringNotEmpty`
Free-text note to attach to the annotation. Required when `status` is RISK_ACCEPTED.
`status` - `AnnotationStatus`
The remediation or review status to assign. See AnnotationStatus for possible values.
`suppress_until_epoch` - `Float`
Unix epoch timestamp (in UTC milliseconds) until which risk notifications should be suppressed. Set to suppress alerts for a defined period.
`remove_suppression` - `Boolean`
Set to `true` to clear any existing risk suppression on this annotation.
`tag_uuids` - `[StringNotEmpty]`
List of Tag UUIDs to associate with this annotation for categorization.

Request

```
# Example 1: Create a new annotation
mutation CreateAnnotation($inputs: [AnnotationInput!]! = [
    {
        target_entity_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
        target_entity_type: WeaknessSeries,
        text: "Patched in maintenance window",
        status: FIXED
    }
]) {
  apply_annotations_to_weakness_series(inputs: $inputs) {
    annotations_count
    annotations {
      uuid
      target_entity_uuid
      target_entity_type
      text
      status
      risk_accepted
      is_suppressed
    }
  }
}

# Example 2: Suppress a weakness series with tags
mutation SuppressWithTags($inputs: [AnnotationInput!]! = [
    {
        target_entity_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd",
        target_entity_type: WeaknessSeries,
        status: RISK_ACCEPTED,
        tag_uuids: ["b2c3d4e5-b2c3-d4e5-b2c3-d4e5b2c3d4e5"]
    }
]) {
  apply_annotations_to_weakness_series(inputs: $inputs) {
    annotations_count
    annotations {
      uuid
      status
      risk_accepted
      is_suppressed
    }
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.16 AnnotationStatus

Description

Represents the user-assigned remediation or review status for a WeaknessSeries. Used to track the lifecycle of weakness management from discovery through resolution.

Values

`FIXED`
The weakness has been remediated and the underlying issue is no longer present.
`POTENTIAL_FALSE_POSITIVE`
The weakness may not be valid; an investigation is underway to determine whether the issue actually exists.
`TO_DO`
The weakness has been acknowledged and remediation is planned, but action has not yet been taken.
`RISK_ACCEPTED`
The weakness has been reviewed and the associated risk has been explicitly accepted without remediation.
`COMPENSATING_CONTROL`
The weakness has been addressed through alternative security controls rather than fixing the underlying issue directly.
`VERIFY_COMPENSATING_CONTROL`
The weakness is awaiting verification that alternative security controls have sufficiently mitigated the risk.
`MUTE`
The weakness has been hidden from active views without resolving or accepting the risk.
`CLOSED`
The weakness has been removed from the active remediation queue.

Example

```
"FIXED"
```

Vuln Management Hub: Weakness Series

8.35.17 ApplyAnnotationOutput

Description

Output returned by apply_annotations_to_weakness_series.

Fields

`annotations_count` - `Int!`
The total number of Annotations that were created or updated.
`annotations` - `[Annotation!]`
The list of Annotation records that were created or updated.

Request Response

```
fragment ApplyAnnotationOutputFragment on ApplyAnnotationOutput {
  annotations_count
  annotations {
    uuid
    target_entity_uuid
    target_entity_type
    text
    status
    risk_accepted
    is_suppressed
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.18 ScheduleWeaknessSeries1CVOpsFailureOutput

Description

Describes a 1CV pentest that could not be scheduled. Common reasons include the assigned runner being unavailable, no valid pentest configuration being found for the weakness, or a configuration error.

Fields

`reason` - `String!`
A human-readable explanation of why the 1CV pentest could not be scheduled.
`op_form` - `ScheduleOpForm`
The ScheduleOpForm configuration that was built for the failed op, if one was available before the error occurred. May be null if the configuration could not be determined.
`for_weakness_series_uuids` - `[String!]`
The WeaknessSeries UUIDs that were associated with the op that failed to schedule.

Request Response

```
fragment ScheduleWeaknessSeries1CVOpsFailureOutputFragment on ScheduleWeaknessSeries1CVOpsFailureOutput {
  reason
  for_weakness_series_uuids
  op_form {
    op_param_max_scope
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.19 ScheduleWeaknessSeries1CVOpsInput

Description

Input for schedule_weakness_series_1cv_ops. Provide the UUIDs of the WeaknessSeries records you want to re-verify.

Fields

`weakness_series_uuids` - `[StringNotEmpty]`
UUIDs of the WeaknessSeries records to re-verify. Pentest configurations are derived automatically from the most recent pentest in which each weakness was found. Weaknesses from the same original pentest are batched into a single 1CV op.
`skip_if_active_or_queued` - `Boolean`
When set to `true`, any weakness series that already has an active or queued 1CV op will be skipped rather than scheduled. Skipped ops appear in `failures` with a reason that identifies the existing op. When omitted or `false`, the default behavior is preserved and no skip check is performed.
`runner_uuid` - `StringNotEmpty`
ID of the NodeZero Runner to use for all 1CV ops scheduled by this call. When provided, all ops use this runner instead of the runner derived from each weakness series's most recent pentest. The runner must belong to the same client account associated with the `weakness_series_uuids` specified.
`run_immediately` - `Boolean`
When `true`, the first op scheduled on each runner will pause any currently running action on that runner and move to the top of the queue. If multiple ops target the same runner (either by coincidence or via `runner_uuid`), only the first can run immediately — the rest queue normally behind it. Requires runner queueing to be enabled for the account and the runner to support queueing.

Request

```
mutation Schedule1CVOps($input: ScheduleWeaknessSeries1CVOpsInput! = {
  weakness_series_uuids: ["1234abcd-1234-abcd-1234-abcd1234abcd"]
  skip_if_active_or_queued: false
}) {
  schedule_weakness_series_1cv_ops(input: $input) {
    ops_count
    ops {
      op {
        op_id
        op_name
      }
    }
    failures {
      reason
      for_weakness_series_uuids
    }
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.20 ScheduleWeaknessSeries1CVOpsOutput

Description

Output from schedule_weakness_series_1cv_ops. Contains the successfully scheduled 1CV pentests and any that could not be scheduled.

Fields

`ops_count` - `Int!`
The number of 1CV pentests successfully scheduled.
`ops` - `[ScheduleOpOutput!]`
The 1CV pentests that were successfully scheduled. Each entry is a ScheduleOpOutput containing the newly created op.
`failures` - `[ScheduleWeaknessSeries1CVOpsFailureOutput!]`
Pentests that could not be scheduled, along with the reason for each failure. See ScheduleWeaknessSeries1CVOpsFailureOutput for details.

Request Response

```
fragment ScheduleWeaknessSeries1CVOpsOutputFragment on ScheduleWeaknessSeries1CVOpsOutput {
  ops_count
  ops {
    op {
      op_id
      op_name
    }
  }
  failures {
    reason
    for_weakness_series_uuids
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.21 Tag

Description

A user-defined or system-defined label that can be applied to Annotations for categorization and filtering. Tags support a hierarchical structure through the `parent_tag` relationship.

Fields

`uuid` - `ID!`
Unique identifier of this tag.
`text` - `String!`
The display text of the tag (e.g., `Network`, `Critical Infrastructure`).
`parent_tag_uuid` - `String`
Unique identifier of the parent tag, if this tag is part of a hierarchy. Null for top-level tags.
`parent_tag` - `Tag`
The parent Tag in the hierarchy, if applicable. Null for top-level tags.
`client_account_uuid` - `String`
Unique identifier of the client account that created this tag. Null if the tag was created by the platform (system-defined).

Request Response

```
fragment TagFragment on Tag {
  uuid
  text
  parent_tag_uuid
  client_account_uuid
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.22 WeaknessSeriesAnnotation

Description

Represents the association between an Annotation and a WeaknessSeries. This bridge record links user-applied annotations (status, notes, tags) to their corresponding weakness series records. Returned by weakness_series_annotations_page.

Fields

`uuid` - `String!`
Unique identifier for this weakness series annotation record.
`row_created_at` - `Datetime!`
Timestamp when this annotation record was created, in ISO 8601 format (UTC).
`row_updated_at` - `Datetime`
Timestamp when this annotation record was last updated, in ISO 8601 format (UTC). Null if never updated after creation.
`weakness_series_uuid` - `String!`
Unique identifier of the WeaknessSeries this annotation is associated with.
`annotation_uuid` - `String!`
Unique identifier of the underlying Annotation record containing the status, text, and tags.
`annotation_status` - `AnnotationStatus!`
The current AnnotationStatus of this annotation (e.g., FIXED, TO_DO, RISK_ACCEPTED).
`annotation_text` - `String`
Free-text note associated with this annotation, providing additional context about the remediation status or risk decision.
`affected_asset_short_name` - `String!`
Abbreviated name of the asset affected by this weakness series (e.g., IP address with port).

Request Response

```
fragment WeaknessSeriesAnnotationFragment on WeaknessSeriesAnnotation {
  uuid
  row_created_at
  row_updated_at
  weakness_series_uuid
  annotation_uuid
  annotation_status
  annotation_text
  affected_asset_short_name
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.23 WeaknessSeriesAnnotationsPage

Description

A paginated list of WeaknessSeriesAnnotation records. Returned by weakness_series_annotations_page.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`annotations` - `[WeaknessSeriesAnnotation!]!`
The list of WeaknessSeriesAnnotation records for the current page.

Request Response

```
fragment WeaknessSeriesAnnotationsPageFragment on WeaknessSeriesAnnotationsPage {
  annotations {
    uuid
    row_created_at
    row_updated_at
    weakness_series_uuid
    annotation_uuid
    annotation_status
    annotation_text
    affected_asset_short_name
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.24 WeaknessSeriesFacets

Description

Available filter facets for WeaknessSeries queries. Each facet provides a list of unique values and their counts in the current dataset, enabling construction of filtered list UIs. Returned by weakness_series_facets.

Fields

`page_info` - `PageInfo`
Pagination metadata for the facet query.
`vuln_short_name` - `[FacetValue]`
Distinct vulnerability short names present in the current dataset, with counts. Useful for filtering weakness series by vulnerability type.
`op_template_name` - `[FacetValue]`
Distinct pentest template names present in the current dataset, with counts. Useful for filtering weakness series by the pentest configuration that discovered them.
`business_risks` - `[FacetValue]`
Distinct business risk categories present in the current dataset, with counts. Useful for filtering weakness series by business impact category.
`vuln_threat_actor_names` - `[FacetValue]`
Distinct threat actor names associated with vulnerabilities in the current dataset, with counts. Useful for filtering weakness series by known threat actors.
`downstream_impact_types` - `[FacetValue]`
Distinct downstream impact types present in the current dataset, with counts. Useful for filtering weakness series by the type of downstream impact (e.g., ransomware exposure).

Request Response

```
fragment WeaknessSeriesFacetsFragment on WeaknessSeriesFacets {
  vuln_short_name {
    label
    value
    value_count
  }
  op_template_name {
    label
    value
    value_count
  }
  business_risks {
    label
    value
    value_count
  }
  vuln_threat_actor_names {
    label
    value
    value_count
  }
  downstream_impact_types {
    label
    value
    value_count
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.25 WeaknessSeriesMetadata

Description

Output from weakness_series_metadata. Contains metadata for the weakness series data.

Fields

`unprocessed_ops_count` - `Int!`
The number of pentests not yet processed into weakness series data.
`unprocessed_ops` - `[Op!]!`
The list of Op objects not yet processed into weakness series data.
`last_processed_op_id` - `String`
The unique identifier of the last processed op in weakness series data.
`last_processed_op` - `Op`
The last processed Op object in weakness series data.

Request Response

```
fragment WeaknessSeriesMetadataFragment on WeaknessSeriesMetadata {
  unprocessed_ops_count
  last_processed_op_id
  unprocessed_ops {
    op_id
    op_name
    op_type
    op_state
    completed_at
  }
  last_processed_op {
    op_id
    op_name
    op_type
    op_state
    completed_at
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Weakness Series

8.35.26 WeaknessSeriesOpStatus

Description

Represents the lifecycle status of a WeaknessSeries as determined by pentest results over time.

Values

`OPEN`
The weakness is actively exploitable and was detected in the most recent pentest.
`MITIGATED`
The weakness was previously detected but a subsequent one-click verify pentest confirmed it has been remediated.
`REGRESSED`
The weakness was previously marked as MITIGATED but has reappeared in a more recent pentest, indicating the remediation was incomplete or has been undone.
`NO_LONGER_FOUND`
The affected asset was scanned in the most recent pentest but the weakness was not detected, suggesting it may have been resolved.

Example

```
"OPEN"
```

Vuln Management Hub: Weakness Series

8.35.27 WeaknessSeriesPage

Description

A paginated list of WeaknessSeries records. Returned by weakness_series_page.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`weakness_series` - `[WeaknessSeries!]!`
The list of WeaknessSeries records for the current page.

Request Response

```
fragment WeaknessSeriesPageFragment on WeaknessSeriesPage {
  weakness_series {
    name
    vuln_id
    vuln_name
    vuln_short_name
    severity
    status
    score
    annotation_status
    is_one_click_verify
    affected_asset_name
    ip
    first_seen_date
    last_seen_date
    weakness_series_found_count
    attack_paths_count
    total_impacts_count
    business_risks
    last_found_op_types
  }
}
```

Examples may not include all available fields for a type.

8.36 Vuln Management Hub: Assets

Query your complete asset inventory across all pentests. An asset is a host discovered during pentesting and tracked over time, often identified by its domain name, IP address, cloud resource ID, and other attributes.

8.36.1 `assets_page`

Description

Returns a paginated list of unique Asset records discovered across all operations for the current client account. Assets are deduplicated across ops using a proprietary identification algorithm. Supports sorting by fields such as `name`, `last_seen_at`, `first_seen_at`, `ops_count`, and `op_series_type`. Supports filtering and text search across asset name, IPs, hostnames, OS names, cloud ARNs, and tags. For MSSP accounts, use `client_account_uuid` to query a subclient's assets.

Response

Returns an `AssetsPage!`

Arguments

`client_account_uuid` - `String`
Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.
`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query AssetsPage($page_input: PageInput) {
  assets_page(page_input: $page_input) {
    assets {
      ...AssetPageFragment
    }
  }
}

fragment AssetPageFragment on Asset {
  uuid
  name
  op_series_type
  first_seen_at
  last_seen_at
  ops_count
  last_op_id
  first_op_scheduled_at
  last_op_scheduled_at
  finding_series_uuid
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Assets

8.36.2 `assets_count`

Description

Returns the total number of unique Asset records discovered across all operations for the current client account. Accepts the same filtering and text search parameters as `assets_page`. For MSSP accounts, use `client_account_uuid` to count a subclient's assets.

Response

Returns an `Int!`

Arguments

`client_account_uuid` - `String`
Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

`page_input` - `PageInput`

Request	Response	Python
cURL	h3-cli	

```
query AssetsCount($page_input: PageInput) {
  assets_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Assets

8.36.3 `assets_csv_url`

Description

Returns a temporary presigned URL to download the full asset inventory as a CSV file. The CSV is pre-generated and stored in S3; this query returns a time-limited download link. Raises an error if no CSV has been generated for the account yet.

Response

Returns a `String`

Arguments

`client_account_uuid` - `String`
Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request	Response	Python
cURL	h3-cli	

```
query AssetsCsvUrl {
  assets_csv_url
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Assets

8.36.4 Asset

Description

Represents a unique asset tracked across multiple penetration testing operations (e.g., a Host).

Assets are identified using a proprietary algorithm that correlates observed attributes such as IP addresses, hostnames, operating systems, and cloud resource ARNs. This cross-operation tracking enables you to see how an asset has been observed over time, including which operations discovered it and when it was first and last seen. Due to the inherent challenges of scan-based asset tracking in dynamic environments, 100% accuracy in asset identification cannot be guaranteed.

Fields

`uuid` - `String!`
Unique identifier for this asset.
`finding_series_uuid` - `String!`
The finding series identifier that links this asset to its underlying cross-operation observation records.
`host_tabs_count` - `Int!`
Returns the total number of Host observations of this asset across operations.

Arguments

`page_input` - `PageInput`
`last_op_id` - `String!`
The operation identifier of the most recent op in which this asset was observed.
`last_op` - `OpTab!`
The most recent operation in which this asset was observed.
`name` - `String!`
The display name of the asset, typically derived from its primary hostname or IP address.
`first_seen_at` - `Datetime!`
Timestamp of when this asset was first observed across all operations.
`last_seen_at` - `Datetime!`
Timestamp of when this asset was most recently observed across all operations.
`ops_count` - `Int!`
The total number of operations in which this asset has been observed.
`first_op_scheduled_at` - `Datetime!`
The scheduled start time of the first operation in which this asset was observed.
`last_op_scheduled_at` - `Datetime!`
The scheduled start time of the most recent operation in which this asset was observed.

Request Response

```
fragment AssetFragment on Asset {
  uuid
  finding_series_uuid
  name
  op_series_type
  first_seen_at
  last_seen_at
  ops_count
  last_op_id
  first_op_scheduled_at
  last_op_scheduled_at
  host_tabs_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Assets

8.36.5 AssetsPage

Description

A paginated list of Asset records. Returned by `assets_page`.

Fields

`page_info` - `PageInfo`
Pagination metadata for the current result set. See PageInfo.
`assets` - `[Asset!]!`
The list of Asset records in the current page.

Request Response

```
fragment AssetsPageFragment on AssetsPage {
  page_info {
    page_num
    page_size
  }
  assets {
    uuid
    name
    op_series_type
    first_seen_at
    last_seen_at
    ops_count
    last_op_id
    first_op_scheduled_at
    last_op_scheduled_at
  }
}
```

Examples may not include all available fields for a type.

8.37 Vuln Management Hub: Perspectives

Create custom attacker perspectives and conduct continuous pentesting of specific segments of your network. Inject credentials to simulate insider threats and compromised accounts. Use perspectives to focus remediation efforts and continuously defend against real-world threat scenarios.

8.37.1 `perspectives_page`

Description

Returns a paginated list of Perspective records in the client's account. Only active (non-deleted) targeted perspectives are returned. Supports sorting by fields such as `name`, `last_exposure_score`, and `last_op_scheduled_at`. Supports filtering by `last_exposure_severity`, `op_type`, and `scheduled` status, as well as text search by perspective name and UUID. For MSSP accounts, use `client_account_uuid` to query a subclient's perspectives.

Response

Returns a `PerspectivesPage!`

Arguments

`page_input` - `PageInput`
`client_account_uuid` - `String`
Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request Response Python
cURL h3-cli

```
query PerspectivesPage($page_input: PageInput = {
  page_num: 1,
  page_size: 5
}) {
  perspectives_page(page_input: $page_input) {
    perspectives {
```

```
      ...PerspectiveFragment
    }
  }
}

fragment PerspectiveFragment on Perspective {
  uuid
  name
  op_template {
    uuid
    name
  }
  schedule {
    uuid
    name
  }
  ops_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.2 `perspectives_count`

Description

Returns the total number of Perspective records in the client's account. Only active (non-deleted) targeted perspectives are counted. Accepts the same filtering parameters as `perspectives_page`. For MSSP accounts, use `client_account_uuid` to count a subclient's perspectives.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

`client_account_uuid` - `String`

Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request	Response	Python
cURL	h3-cli	

```
query PerspectivesCount($page_input: PageInput) {
  perspectives_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.3 `perspective`

Description

Returns a single Perspective by its unique identifier. Returns `null` if the perspective does not exist or has been deleted.

Response

Returns a `Perspective`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query Perspective($uuid: String!) {
  perspective(uuid: $uuid) {
    uuid
    name
    op_template {
      uuid
      name
    }
    schedule {
      uuid
      name
    }
    ops_count
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.4 `org_perspective`

Description

Returns the account-wide Org Perspective for the current client account.

The Org Perspective automatically aggregates data from all targeted perspectives in the account, covering operations from the past 12 months. This provides a holistic view of your organization's security posture trends without requiring manual perspective selection.

Response

Returns a `Perspective`

Arguments

`client_account_uuid` - `String`
Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request | Response | Python
cURL | h3-cli

```
query OrgPerspective {
  org_perspective {
    uuid
    name
    op_template {
      uuid
      name
    }
    schedule {
      uuid
      name
    }
    ops_count
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.5 `perspectives_up_next`

Description

Returns Perspective records ordered by their next scheduled execution time (soonest first). Only perspectives with an active schedule are included. Defaults to returning 3 perspectives. Also includes scheduled operations that are not yet associated with a Perspective.

Response

Returns a `PerspectivesPage!`

Arguments

`count` - `Int`

`client_account_uuid` - `String`

Optional subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request | Response | Python

cURL | h3-cli

```
query PerspectivesUpNext($count: Int = 3) {
  perspectives_up_next(count: $count) {
    perspectives {
      ...PerspectiveFragment
    }
  }
}

fragment PerspectiveFragment on Perspective {
  uuid
  name
  op_template {
    uuid
    name
  }
  schedule {
    uuid
    name
  }
  ops_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.6 Perspective

Description

Represents a recurring pentesting configuration, also referred to as an attacker's perspective.

A Perspective groups together the key components of a repeatable pentest:

1. An asset scope and attack profile defined by an OpTemplate.
2. Optionally, injected credentials to simulate specific attack scenarios.
3. A consistent NodeZero Docker host (runner) for executing the pentest.
4. A Schedule for automated recurring execution.

By running multiple pentests under the same Perspective, you can track trends over time and measure the effectiveness of remediation efforts.

Visit Campaigns in the Horizon3.ai documentation for more details on using Campaigns and Perspectives to organize and manage your pentesting efforts.

Fields

`uuid` - `String!`
Unique identifier for this Perspective.
`name` - `String!`
The user-defined name for this Perspective.
`op_template` - `OpTemplate`
The OpTemplate that defines the asset scope and attack profile used when launching new pentests under this Perspective.
`schedule` - `Schedule`
The Schedule that controls automated pentest execution for this Perspective. Null if no schedule is configured.
`ops_count` - `Int!`
Returns the total number of pentesting operations associated with this Perspective's underlying op series.

Arguments

`page_input` - `PageInput`

Pagination and filtering options. See PageInput.

`ops_page` - `OpsPage!`
Returns a paginated list of pentesting operations associated with this Perspective's underlying op series. See OpsPage.

Arguments

`page_input` - `PageInput`

Pagination and filtering options. See PageInput.

Request Response

```
fragment PerspectiveFragment on Perspective {
  uuid
  name
  op_template {
    uuid
    name
  }
  schedule {
    uuid
    name
  }
  ops_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Perspectives

8.37.7 PerspectivesPage

Description

A paginated list of Perspective records. Returned by queries such as `perspectives_page` and `perspectives_up_next`.

Fields

`page_info` - `PageInfo`
Pagination metadata for the current result set. See PageInfo.
`perspectives` - `[Perspective!]!`
The list of Perspective records in the current page.

Request Response

```
fragment PerspectivesPageFragment on PerspectivesPage {
  perspectives {
    uuid
    name
    op_template {
      uuid
      name
    }
    schedule {
      uuid
      name
    }
    ops_count
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

8.38 Vuln Management Hub: Campaigns

Create and manage focused cybersecurity campaigns across a set of perspectives. Use campaigns to coordinate large-scale pentesting efforts and track remediation progress across your network.

8.38.1 `campaigns`

Description

Returns all active (non-deleted) Campaigns for the authenticated user's client account. Results are ordered with the `ORG`-scoped default campaign first, followed by `TARGETED` campaigns sorted alphabetically by name.

Response

Returns `[Campaign!]!`

Request Response Python
cURL h3-cli

```
query Campaigns {
  campaigns {
    ...CampaignFragment
  }
}

fragment CampaignFragment on Campaign {
  uuid
  name
  description
  scope
  client_account_uuid
  is_deleted
  deleted_at
  row_created_at
  row_updated_at
```

```
  perspective_uuids
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.2 `campaign`

Description

Retrieves a single Campaign by its UUID. Returns `null` if the campaign does not exist or has been soft-deleted. Only returns campaigns belonging to the authenticated user's client account.

Response

Returns a `Campaign`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query Campaign($uuid: String!) {
  campaign(uuid: $uuid) {
    ...CampaignFragment
  }
}

fragment CampaignFragment on Campaign {
  uuid
  name
  description
  scope
  client_account_uuid
  is_deleted
  deleted_at
  row_created_at
  row_updated_at
  perspective_uuids
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.3 `create_campaign`

Description

Creates a new Campaign for the authenticated user's client account. Campaign names must be unique within a client account. Only one `ORG`-scoped campaign may exist per client account. Returns the newly created campaign, including system-generated fields such as `uuid` and `row_created_at`.

Response

Returns a `Campaign!`

Arguments

`input` - `CreateCampaignInput!`

Request | Response | Python
cURL | h3-cli

```
mutation CreateCampaign($input: CreateCampaignInput!) {
  create_campaign(input: $input) {
    ...CampaignFragment
  }
}

fragment CampaignFragment on Campaign {
  uuid
  name
  description
  scope
  client_account_uuid
  is_deleted
  row_created_at
  row_updated_at
  perspective_uuids
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.4 `update_campaign`

Description

Updates an existing Campaign. Only provided fields in the UpdateCampaignInput are modified. Perspective associations cannot be modified on `ORG`-scoped campaigns. Returns the updated campaign with all current field values.

Response

Returns a `Campaign!`

Arguments

`input` - `UpdateCampaignInput!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateCampaign($input: UpdateCampaignInput!) {
  update_campaign(input: $input) {
    ...CampaignFragment
  }
}

fragment CampaignFragment on Campaign {
  uuid
  name
  description
  scope
  client_account_uuid
  is_deleted
  row_created_at
  row_updated_at
  perspective_uuids
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.5 `delete_campaign`

Description

Soft-deletes a Campaign by UUID. The default `ORG`-scoped campaign cannot be deleted. Returns `true` on success.

Response

Returns a `Boolean!`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteCampaign($uuid: String!) {
  delete_campaign(uuid: $uuid)
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.6 `campaign_available_perspectives_page`

Description

Returns a paginated list of Perspectives that are available to be added to a campaign. When a `campaign_uuid` is provided, perspectives already associated with that campaign are excluded from the results.

Response

Returns a `PerspectivesPage!`

Arguments

`page_input` - `PageInput`

`campaign_uuid` - `String`
When provided, excludes perspectives that are already associated with the specified campaign.

`client_account_uuid` - `String`
Subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request | Response | Python
cURL | h3-cli

```
query CampaignAvailablePerspectivesPage(
  $page_input: PageInput = { page_num: 1, page_size: 5 },
  $campaign_uuid: String
) {
  campaign_available_perspectives_page(
    page_input: $page_input
    campaign_uuid: $campaign_uuid
  ) {
    perspectives {
      ...PerspectiveFragment
    }
  }
}

fragment PerspectiveFragment on Perspective {
  uuid
  name
  op_template {
    uuid
    name
  }
  schedule {
    uuid
    name
  }
  ops_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.7 `campaign_available_perspectives_count`

Description

Returns the total count of Perspectives available to be added to a campaign. When a `campaign_uuid` is provided, perspectives already associated with that campaign are excluded from the count.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

`campaign_uuid` - `String`
When provided, excludes perspectives that are already associated with the specified campaign.

`client_account_uuid` - `String`
Subclient UUID for MSSP accounts. When omitted, defaults to the authenticated user's client account.

Request | Response | Python
cURL | h3-cli

```
query CampaignAvailablePerspectivesCount(
  $campaign_uuid: String
) {
  campaign_available_perspectives_count(
    campaign_uuid: $campaign_uuid
  )
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.8 Campaign

Description

A Campaign is an organizational grouping of Perspectives that allows users to track and manage related pentesting configurations together. Each client account has one system-generated `ORG`-scoped campaign that automatically includes all perspectives, plus any number of user-created `TARGETED` campaigns containing curated subsets. Campaign names must be unique within a client account. Campaigns support soft deletion -- the default `ORG` campaign cannot be deleted.

Visit Campaigns in the Horizon3.ai documentation for more details on using Campaigns and Perspectives to organize and manage your pentesting efforts.

Fields

`uuid` - `String!`
Unique identifier for the campaign.
`name` - `String!`
Display name of the campaign. Unique within the client account.
`description` - `String!`
User-provided description of the campaign's purpose.
`scope` - `CampaignScope!`
Scope of the campaign. See CampaignScope for details.
`client_account_uuid` - `String!`
UUID of the client account that owns this campaign.
`is_deleted` - `Boolean!`
Indicates whether the campaign has been soft-deleted. Soft-deleted campaigns are excluded from list queries.
`deleted_at` - `String`
Timestamp when the campaign was soft-deleted, or `null` if the campaign is active.
`row_created_at` - `String!`
Timestamp when the campaign was created.
`row_updated_at` - `String`
Timestamp when the campaign was last updated, or `null` if the campaign has never been modified.
`perspective_uuids` - `[String!]`
List of perspective UUIDs associated with this campaign. For `ORG` campaigns, this includes all perspectives in the client account.
`perspectives_page` - `PerspectivesPage!`
Returns a paginated list of Perspectives included in this campaign. Accepts standard pagination, filtering, and sorting via `page_input`.

Arguments

`page_input` - `PageInput`
`perspectives_count` - `Int!`
Returns the total number of Perspectives included in this campaign. Accepts filtering via `page_input`.

Arguments

`page_input` - `PageInput`
`up_next` - `PerspectivesPage!`
Returns the next upcoming scheduled pentests for perspectives in this campaign, ordered by next execution time. Defaults to 3 results. For `ORG` campaigns, this also includes standalone schedules not attached to any perspective.

Arguments

`count` - `Int`
`recommended_perspectives` - `[RecommendedPerspective!]!`
Returns a list of RecommendedPerspective records summarizing perspective coverage by operation type for this campaign, sorted so that uncovered operation types appear first.

Request Response

```
fragment CampaignFragment on Campaign {
  uuid
  name
  description
  scope
  client_account_uuid
  is_deleted
  deleted_at
  row_created_at
  row_updated_at
  perspective_uuids
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.9 CampaignScope

Description

Defines the scope of a Campaign, distinguishing between the system-managed organization-wide default campaign and user-created targeted campaigns.

Values

`ORG`
Organization-wide default campaign that automatically includes all perspectives for the client account. Each client account has exactly one ORG campaign, which cannot be deleted.
`TARGETED`
User-created campaign containing a curated subset of perspectives. TARGETED campaigns can be created, updated, and deleted by authorized users.

Example

```
"ORG"
```

Vuln Management Hub: Campaigns

8.38.10 CreateCampaignInput

Description

Input for the create_campaign mutation. Defines the attributes of the new Campaign to create.

Fields

`name` - `String!`
Name for the campaign. Required, must be unique within the client account, and cannot exceed 128 characters.
`description` - `String!`
Description for the campaign. Required and cannot exceed 256 characters.
`scope` - `CampaignScope`
Scope of the campaign. Defaults to `TARGETED` if not specified. Only one `ORG`-scoped campaign may exist per client account. See CampaignScope for details.
`perspective_uuids` - `[String!]`
List of perspective UUIDs to associate with the campaign at creation time. Only applicable for `TARGETED` campaigns; `ORG` campaigns automatically include all perspectives.

Request

```
mutation CreateCampaignExample(
  $input: CreateCampaignInput = {
    name: "Q1 Security Campaign"
    description: "Quarterly security assessment campaign"
    scope: TARGETED
    perspective_uuids: [
      "1234abcd-1234-abcd-1234-abcd1234abcd"
    ]
  }
) {
  create_campaign(input: $input) {
    uuid
    name
    description
    scope
  }
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.11 RecommendedPerspective

Description

Represents a recommended operation type for a Campaign, along with counts of how many perspectives of that type are currently included. Results are sorted so that operation types with no perspectives appear first, making it easy to identify coverage gaps. The recommended operation types are: NodeZero, ExternalAttack, ADPasswordAudit, K8sPentest, Phishing, InsiderThreatAttack, AWSPentest, and AzurePentest.

Fields

`id` - `String!`
Unique identifier for the recommendation. Currently matches the `op_type` value.
`op_type` - `OpType!`
The operation type this recommendation covers (e.g., `NodeZero`, `ExternalAttack`, `ADPasswordAudit`).
`total_count` - `Int!`
Total number of perspectives of this operation type currently included in the campaign.
`scheduled_count` - `Int!`
Number of perspectives of this operation type that have an active schedule configured.

Request Response

```
fragment RecommendedPerspectiveFragment on RecommendedPerspective {
  id
  op_type
  total_count
  scheduled_count
}
```

Examples may not include all available fields for a type.

Vuln Management Hub: Campaigns

8.38.12 UpdateCampaignInput

Description

Input for the update_campaign mutation. All fields except `uuid` are optional; only provided fields will be modified. Perspective associations cannot be modified on `ORG`-scoped campaigns.

Fields

`uuid` - `String!`
UUID of the Campaign to update. Required.
`name` - `String`
Updated campaign name. Must be unique within the client account and cannot exceed 128 characters.
`description` - `String`
Updated campaign description. Cannot exceed 256 characters.
`add_perspective_uuids` - `[String!]`
List of perspective UUIDs to add to the campaign. Not applicable for `ORG`-scoped campaigns.
`delete_perspective_uuids` - `[String!]`
List of perspective UUIDs to remove from the campaign. Not applicable for `ORG`-scoped campaigns.

Request

```
mutation UpdateCampaignExample(
  $input: UpdateCampaignInput = {
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    name: "Updated Campaign Name"
    description: "Updated campaign description"
    add_perspective_uuids: [
      "5678abcd-5678-abcd-5678-abcd5678abcd"
    ]
    delete_perspective_uuids: [
      "9999aaaa-9999-aaaa-9999-aaaa9999aaaa"
    ]
  }
) {
  update_campaign(input: $input) {
    uuid
    name
    description
  }
}
```

Examples may not include all available fields for a type.

8.39 Platform Features

Advanced platform capabilities to enhance your security testing. Leverage Rapid Response for immediate threat validation, deploy Tripwires for breach detection and monitoring, and integrate vulnerability scanner data with Vuln Risk Intelligence to prioritize findings based on NodeZero's exploitation validation.

8.40 Platform Features: Rapid Response

Be notified of emerging threats and zero-day vulnerabilities and launch targeted tests to assess if your environment is vulnerable to newly disclosed CVEs or attack techniques.

8.40.1 `notifications_page`

Description

Returns a paginated list of Notifications for the authenticated user. Includes notifications from both Rapid Response and Tripwire activity. Use the `unread` parameter to filter for unread notifications only.

Response

Returns a `NotificationsPage!`

Arguments

`unread` - `Boolean`

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query NotificationsPage($unread: Boolean, $page_input: PageInput) {
  notifications_page(unread: $unread, page_input: $page_input) {
    notifications {
      type
      data {
        tripwire {
          uuid
          external_id
          type
          state
        }
        trip {
          tripped_at
          detected_at
          is_test
        }
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.2 `notifications_count`

Description

Returns the total count of Notifications for the authenticated user. Use the `unread` parameter to count only unread notifications.

Response

Returns an `Int!`

Arguments

`unread` - `Boolean`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query NotificationsCount($unread: Boolean) {
  notifications_count(unread: $unread)
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.3 `notification`

Description

Retrieves a single Notification by its UUID. Returns `null` if the notification does not exist or does not belong to the authenticated user. For `TripwireTripped` notifications, accessing this query also records a view event.

Response

Returns a `Notification`

Arguments

`notification_uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query GetNotification($notification_uuid: String!) {
  notification(notification_uuid: $notification_uuid) {
    type
    data {
      tripwire {
        uuid
        external_id
        type
        state
      }
      trip {
        tripped_at
        detected_at
        is_test
      }
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.4 `apply_annotations_to_rapid_response_assets`

Description

Applies an Annotation to one or more RapidResponseAsset records within a single RapidResponse. The annotation is attached to the underlying RapidResponseEntity records for each asset; for FEE-backed entities the annotation also influences the corresponding WeaknessSeries status rollup.

A single new RapidResponseEvent of type `ANNOTATION_CREATED` is created per API call, capturing the user's action for audit and timeline purposes. The asset's `last_annotation_status` is updated, and the projected `status` is recomputed per the conditional mitigation rule.

Returns the count and list of Annotation records created via ApplyAnnotationOutput.

Response

Returns an `ApplyAnnotationOutput!`

Arguments

`input` - `ApplyAnnotationsToRapidResponseAssetsInput!`

Request | Response | Python
cURL | h3-cli

```
mutation ApplyAnnotationsToRapidResponseAssets(
  $input: ApplyAnnotationsToRapidResponseAssetsInput!
) {
  apply_annotations_to_rapid_response_assets(input: $input) {
    annotations_count
    annotations {
      uuid
      target_entity_uuid
      target_entity_type
      text
      status
      risk_accepted
      is_suppressed
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.5 ApplyAnnotationsToRapidResponseAssetsInput

Description

Input for apply_annotations_to_rapid_response_assets.

Carries both the asset selector (via RapidResponseAssetsInput) and the annotation body to apply to every selected asset.

Fields

`status` - `AnnotationStatus!`
The remediation or review status to assign. See AnnotationStatus for possible values.
`text` - `StringNotEmpty`
Free-text note to attach to the annotation. Required when `status` is `RISK_ACCEPTED`.

Request

```
# Example 1: Apply FIXED annotation to specific asset UUIDs
mutation ApplyFixedToAssets(
  $input: ApplyAnnotationsToRapidResponseAssetsInput = {
    assets: {
      rapid_response_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
      rapid_response_asset_uuids: ["aaaa1111-bbbb-2222-cccc-dddd3333eeee"]
    }
    status: FIXED
    text: "Patched in maintenance window"
  }
) {
  apply_annotations_to_rapid_response_assets(input: $input) {
    annotations_count
  }
}

# Example 2: Risk-accept all assets on a RapidResponse
mutation RiskAcceptAllAssets(
  $input: ApplyAnnotationsToRapidResponseAssetsInput = {
    assets: {
      rapid_response_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    }
    status: RISK_ACCEPTED
    text: "Reviewed and accepted by security team"
  }
) {
  apply_annotations_to_rapid_response_assets(input: $input) {
    annotations_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.6 Notification

Description

Represents an alert generated by Rapid Response exploit testing or Tripwire activity detection. Notifications are scoped to a specific user (by email) within a client account and can be queried via notifications_page.

Fields

`data` - `NotificationData!`
Detailed payload of the notification. The populated fields vary based on the notification `type`. See NotificationData.
`type` - `NotificationType!`
Category of the notification, indicating whether it relates to a Rapid Response finding or tripwire activity. See NotificationType.

Request Response

```
fragment NotificationFragment on Notification {
  type
  data {
    tripwire {
      uuid
      external_id
      type
      state
    }
    trip {
      tripped_at
      detected_at
      is_test
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.7 NotificationData

Description

Contains the detailed payload of a Notification. The populated fields vary depending on the NotificationType: Rapid Response notifications include vulnerability context and affected entities, while Tripwire notifications include the associated tripwire and trip activity records.

Fields

`tripwire` - `Tripwire`
The Tripwire associated with this notification. Present when the notification type is `TripwireTripped`.

`trip` - `Trip`
The Trip activity record associated with this notification. Present when the notification type is `TripwireTripped`, containing details about the detected activity.

Request Response

```
fragment NotificationDataFragment on NotificationData {
  tripwire {
    uuid
    external_id
    type
    state
    deployment_asset_name
  }
  trip {
    tripped_at
    detected_at
    is_test
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Rapid Response

8.40.8 NotificationType

Description

Classifies the type of a Notification, indicating the event category that triggered the alert. Public values cover confirmed Rapid Response findings and tripwire activity detections.

Values

`EvidenceOfWeakness`
Indicates that confirmed evidence of a breaking vulnerability was found during Rapid Response exploit testing. Recipients should take immediate action to assess whether the vulnerability is still present, such as by running a targeted Rapid Response test if one is available.

`PossibleEvidenceOfWeakness`
Possible Evidence of Weakness Found, but not confirmable

`NoEvidenceOfWeakness`
No Evidence of Weakness Found

`TripwireTripped`
Indicates that activity was detected on an active Tripwire deployed in the recipient's environment. Recipients should urgently review the associated Trip details and take appropriate remedial action, such as quarantining the affected asset or deactivating the alerting tripwire if warranted.

Example

```
"EvidenceOfWeakness"
```

Platform Features: Rapid Response

8.40.9 NotificationsPage

Description

Contains a page of Notification records returned by notifications_page.

Fields

`page_info` - `PageInfo`
Pagination metadata for the current page of results.
`notifications` - `[Notification!]!`
The list of Notification records in the current page.

Request Response

```
fragment NotificationsPageFragment on NotificationsPage {
  page_info {
    page_num
    page_size
  }
  notifications {
    type
    data {
      tripwire {
        uuid
        external_id
        type
        state
      }
      trip {
        tripped_at
        detected_at
        is_test
      }
    }
  }
}
```

Examples may not include all available fields for a type.

8.41 Platform Features: Tripwires

Deploy honeypot credentials and deception assets (tripwires) to detect unauthorized access in your environment. Monitor tripwire triggers to identify active attackers and insider threats.

8.41.1 `tripwires_page`

Description

Returns a paginated list of Tripwires deployed for the authenticated user's client account. Includes tripwires in all lifecycle states (`ACTIVE` , `PENDING` , `INACTIVE` , `ERROR`). Supports standard pagination, filtering, and sorting via `page_input` .

Response

Returns a `TripwiresPage!`

Arguments

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query TripwiresPage($page_input: PageInput) {
  tripwires_page(page_input: $page_input) {
    tripwires {
      ...TripwireFragment
    }
  }
}
```

```
fragment TripwireFragment on Tripwire {
  uuid
  external_id
  type
  op_id
  deployment_asset_name
  file_attributes {
    name
    path
    sha1
  }
  processes
  state
  active_at
  inactive_at
  last_tested_at
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.2 `tripwires_csv_presigned_url`

Description

Returns a presigned S3 URL to download a CSV export of all Tripwires deployed for the current client account, including their name, type, deployment location, file details, monitored processes, status, and deployment date. The CSV is regenerated on each request to reflect the latest tripwire data. Optionally pass a list of Tripwire UUIDs via the `uuids` parameter to export only selected tripwires.

The CSV column format is documented under TripwireCSV.

Response

Returns a `String`

Arguments

`uuids` - `[ID!]`

Request | Response | Python

cURL | h3-cli

```
query TripwiresCsvPresignedUrl($uuids: [ID!]) {
  tripwires_csv_presigned_url(uuids: $uuids)
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.3 `tripwires_count`

Description

Returns the total count of Tripwires for the authenticated user's client account. Supports filtering via `page_input` to match the criteria used with tripwires_page.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query TripwiresCount($page_input: PageInput) {
  tripwires_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.4 `tripwire_jobs_page`

Description

Returns a paginated list of tripwire jobs (TripwireInternal operations) for the authenticated user's client account. Each job represents an operation that deployed one or more Tripwires. Results are returned as OpTab records. Supports standard pagination, filtering, and sorting via `page_input`.

Response

Returns an `OpTabsPage!`

Arguments

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query TripwireJobsPage($page_input: PageInput) {
  tripwire_jobs_page(page_input: $page_input) {
    op_tabs {
      uuid
      op_id
      op_name
      op_state
      portal_op_state
      op_type
      scheduled_at
      launched_at
      completed_at
      duration_hms
      tripwires_count
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.5 `tripwire_jobs_count`

Description

Returns the total count of tripwire jobs (TripwireInternal operations) for the authenticated user's client account. Supports filtering via `page_input` to match the criteria used with tripwire_jobs_page.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query TripwireJobsCount($page_input: PageInput) {
  tripwire_jobs_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.6 `test_tripwire`

Description

Triggers an on-demand test of the specified Tripwire to verify that its detection mechanism is operational. The resulting Trip event will have `is_test` set to `true`. Returns the tested tripwire with updated state.

Response

Returns a `Tripwire`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
mutation TestTripwire($uuid: String!) {
  test_tripwire(uuid: $uuid) {
    uuid
    external_id
    type
    state
    deployment_asset_name
    last_tested_at
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.7 Tripwire

Description

A honeytoken-style decoy deployed during an Internal, External, Phishing Impact, or Rapid Response pentest to detect unauthorized access or lateral movement within the target environment. During a pentest, NodeZero automatically places tripwires on compromised hosts, data stores, and within Active Directory along critical attack paths. Tripwires are lightweight and remain silent until triggered — only when an attacker interacts with a tripwire does it generate a Trip event and alert. Each tripwire has a type that determines the detection channel (e.g., cloud credential monitoring, DNS callbacks, or AD authentication event monitoring) and a lifecycle state tracking its operational status. Tripwires can be tested on demand via test_tripwire to validate alerting infrastructure without impacting the target environment.

Visit Tripwires in the Horizon3.ai documentation for more details on how tripwires work, best practices for using them effectively, and how to interpret tripwire alerts.

Fields

`uuid` - `ID!`
Unique identifier for the tripwire.
`external_id` - `String!`
External identifier for the tripwire, typically used as the display name. The format varies by tripwire type: for `AWS_ACCESS_KEY` tripwires this is the AWS access key ID; for `WINDOWS_SUSPICIOUS_PROCESS` and `MYSQL_DUMP` tripwires this is the DNS subdomain embedded within the tripwire; and for Active Directory tripwires (KERBEROAST, AS_REP_ROAST, EXPOSED_PASSWORD) this is the fully qualified user name of the tripwire account.
`type` - `TripwireType!`
The tripwire type. See TripwireType for the full list of supported types.
`op_id` - `String`
Op ID of the pentest or tripwire job during which this tripwire was deployed, if applicable.
`op` - `OpTab`
Operation details for the pentest or tripwire job during which this tripwire was deployed, if applicable. See OpTab.
`deployment_asset_name` - `String!`
Human-readable name describing the host and/or data store where the tripwire was deployed, as determined at deployment time.
`file_attributes` - `[FileAttributes!]`
List of FileAttributes for each file deployed as part of the tripwire, including the file name, file system path, and SHA1 hash.
`processes` - `[String!]`
List of process names monitored by the tripwire. Only populated for `WINDOWS_SUSPICIOUS_PROCESS` tripwires.
`state` - `TripwireState!`
Current lifecycle state of the tripwire. Transitions from `PENDING` to `ACTIVE` once deployed, and may be set to `INACTIVE` when decommissioned. Deactivation is permanent — once a tripwire is deactivated, it cannot be reactivated. For Active Directory tripwires, the state may reflect `ERROR` if the underlying AD account or agent session is in an error state. See TripwireState.
`active_at` - `Date`
Timestamp when the tripwire transitioned to the `ACTIVE` state, or `null` if not yet activated.
`inactive_at` - `Date`
Timestamp when the tripwire transitioned to the `INACTIVE` state, or `null` if still active.
`last_tested_at` - `Date`
Timestamp when the tripwire was last tested via test_tripwire, or `null` if never tested.
`data_store` - `DataStore`
The DataStore in which the tripwire was deployed, if applicable. Contains details collected during the deploying operation.

Request Response

```
fragment TripwireFragment on Tripwire {
  uuid
```

```
  external_id
  type
  op_id
  deployment_asset_name
  file_attributes {
    name
    path
    sha1
  }
  processes
  state
  active_at
  inactive_at
  last_tested_at
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.8 FileAttributes

Description

Metadata about a file deployed as part of a Tripwire. Each entry describes one file placed on the target system during tripwire deployment, including its location and integrity hash.

Fields

`name` - `String!`
Name of the deployed file.
`path` - `String!`
Full file system path where the file was deployed.
`sha1` - `String`
SHA1 hash of the file contents, used for integrity verification.

Request Response

```
fragment FileAttributesFragment on FileAttributes {
  name
  path
  sha1
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.9 Trip

Description

Represents a single activity detection event recorded by a Tripwire. Each Trip captures when unauthorized or suspicious activity triggered the tripwire, when the event was detected, and detailed forensic attributes about the source of the activity.

Fields

`tripped_at` - `Date!`
Timestamp when the trip event occurred (i.e., when the unauthorized or suspicious activity took place).
`detected_at` - `Date!`
Timestamp when the trip event was detected by the monitoring system.
`is_test` - `Boolean!`
Indicates whether this trip event was caused by an on-demand test of the tripwire (via test_tripwire) rather than real activity.
`attributes` - `TripAttributes!`
Detailed forensic attributes of the trip event. The populated fields vary by tripwire type. See TripAttributes.

Request Response

```
fragment TripFragment on Trip {
  tripped_at
  detected_at
  is_test
  attributes {
    trip_time
    geoip
    source_hostname
    source_ip_address
    source_edns_client_subnet
    source_process_name
    source_username
    event_id
    source_ip_port
    dc_hostname
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.10 TripAttributes

Description

Detailed forensic attributes of a Trip event. The populated fields vary depending on the TripwireType: DNS callback-based tripwires (WINDOWS_SUSPICIOUS_PROCESS, MYSQL_DUMP) populate DNS and subnet fields; AWS_ACCESS_KEY tripwires populate CloudTrail-derived fields; and Active Directory tripwires (KERBEROAST, AS_REP_ROAST, EXPOSED_PASSWORD) populate Windows security event fields.

Fields

`trip_time` - `String`
Timestamp when the trip event occurred.
`geoip` - `JSONObject`
Geographic location information derived from the source IP address of the trip event, returned as a JSON object.
`source_hostname` - `String`
The hostname of the source machine where the triggering activity occurred. For WINDOWS_SUSPICIOUS_PROCESS tripwires, this is the computer where the process was run. For MYSQL_DUMP tripwires, this is the machine where the dump file was loaded. For Active Directory tripwires, this is the machine that generated the security event.
`source_ip_address` - `String`
Source IP address from which the trip event originated.
`source_edns_client_subnet` - `String`
EDNS Client Subnet (ECS) from the DNS query that triggered the trip event. Applicable to DNS callback-based tripwires (WINDOWS_SUSPICIOUS_PROCESS, MYSQL_DUMP). Often more accurate than `source_ip_address` for identifying the originating network.
`source_process_name` - `String`
Name of the process that triggered the trip event. Applicable to WINDOWS_SUSPICIOUS_PROCESS tripwires.
`source_username` - `String`
Username of the account that ran the process triggering the trip event. Applicable to WINDOWS_SUSPICIOUS_PROCESS tripwires.
`event_id` - `Int`
Windows Security Event ID associated with the trip event. Applicable to Active Directory tripwires (KERBEROAST, AS_REP_ROAST, EXPOSED_PASSWORD).
`source_ip_port` - `String`
Source IP port from which the authentication attempt originated. Applicable to Active Directory tripwires.
`dc_hostname` - `String`
Hostname of the domain controller that logged the security event. Applicable to Active Directory tripwires.

Request Response

```
fragment TripAttributesFragment on TripAttributes {
  trip_time
  geoip
  source_hostname
  source_ip_address
  source_edns_client_subnet
  source_process_name
  source_username
  event_id
  source_ip_port
  dc_hostname
}
```

Examples may not include all available fields for a type.

Platform Features: Tripwires

8.41.11 TripwireState

Description

Represents the lifecycle state of a Tripwire. A tripwire transitions from `PENDING` to `ACTIVE` once deployed, and may be set to `INACTIVE` when decommissioned. The `ERROR` state indicates a deployment or operational issue requiring attention.

Values

`PENDING`
The tripwire has been created but deployment has not yet completed.
`ACTIVE`
The tripwire is deployed and actively monitoring for unauthorized activity. Only `ACTIVE` tripwires generate Trip alerts and can be tested via test_tripwire.
`INACTIVE`
The tripwire has been permanently deactivated and will no longer generate alerts. Deactivation cannot be reversed.
`ERROR`
The tripwire encountered a deployment or operational issue. For Active Directory tripwires, this may indicate a problem with the underlying AD account or agent session.

Example

```
"PENDING"
```

Platform Features: Tripwires

8.41.12 TripwireType

Description

The set of Tripwire types. Each type corresponds to a different deployment method and detection channel. During a pentest, NodeZero selects the appropriate tripwire type based on the compromised asset and deploys decoys that blend into the target environment. The detection channels range from cloud credential monitoring (AWS, Azure, GCP) to DNS callback-based detection and Active Directory authentication event monitoring.

Values

`AWS_ACCESS_KEY`
A decoy AWS credentials file placed on network shares or compromised hosts. Tripped when an actor attempts to use the embedded AWS access key ID, which is a real credential generated by Horizon3.ai for detection purposes.
`MYSQL_DUMP`
A deceptive MySQL dump file placed on network shares or Linux servers. Tripped when an actor attempts to import or load the contents, which triggers a DNS callback to Horizon3.ai detection systems.
`WINDOWS_SUSPICIOUS_PROCESS`
A process monitor deployed on Windows 7+ machines that watches for execution of commonly used attacker reconnaissance tools (e.g., `tasklist.exe`, `certutil.exe`, `systeminfo.exe`, `netstat.exe`, `at.exe`). Tripped when a monitored process is executed, triggering a DNS callback to Horizon3.ai detection systems.
`KERBEROAST`
A decoy Active Directory account configured with a Service Principal Name (SPN). Tripped when an attacker requests a Kerberos TGS (Ticket Granting Service) ticket for the account, a technique used to obtain service tickets for offline password cracking.
`AS_REP_ROAST`
A decoy Active Directory account configured with Kerberos pre-authentication disabled. Tripped when an attacker requests an AS-REP (Authentication Server Response) for the account, a technique used to obtain authentication responses for offline password cracking.
`EXPOSED_PASSWORD`
A decoy Active Directory account with a credential hint embedded in the user description. Tripped when an attacker scrapes the exposed credential and attempts to authenticate to Active Directory with it.

Example

```
"AWS_ACCESS_KEY"
```

Platform Features: Tripwires

8.41.13 TripwiresPage

Description

Contains a page of Tripwire records returned by tripwires_page.

Fields

`page_info` - `PageInfo`
Pagination metadata for the current page of results.
`tripwires` - `[Tripwire!]!`
The list of Tripwire records in the current page.

Request Response

```
fragment TripwiresPageFragment on TripwiresPage {
  tripwires {
    uuid
    external_id
    type
    state
    deployment_asset_name
    op_id
    active_at
    inactive_at
    last_tested_at
    file_attributes {
      name
      path
      sha1
    }
    processes
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

8.42 Platform Features: Vuln Risk Intelligence

Import vulnerability scan data from third-party scanners and enrich findings with NodeZero's exploit validation and risk intelligence. Prioritize scanner findings based on actual exploitability demonstrated by autonomous pentesting.

8.42.1 `scanner_files_page`

Description

Returns a paginated list of ScannerFile records for the current user's client account. Use PageInput to control pagination, sorting, and filtering.

Response

Returns a `ScannerFilesPage!`

Arguments

`page_input` - `PageInput`

Request Response Python
cURL h3-cli

```
query ScannerFilesPage($page_input: PageInput) {
  scanner_files_page(page_input: $page_input) {
    scanner_files {
      ...ScannerFileFragment
    }
  }
}

fragment ScannerFileFragment on ScannerFile {
```

```
  uuid
  vendor
  file_name
  file_size_bytes
  file_size_formatted
  format
  status
  status_message
  status_progress
  uploading_started_at
  uploading_ended_at
  processing_started_at
  processing_ended_at
  expires_at
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.2 `scanner_files_count`

Description

Returns the total count of vulnerability scanner files for the current user's client account. Accepts an optional PageInput to apply filters before counting.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query ScannerFilesCount($page_input: PageInput) {
  scanner_files_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.3 `scanner_file`

Description

Returns a single ScannerFile by its unique identifier. The file must belong to the current user's client account.

Response

Returns a `ScannerFile`

Arguments

`uuid` - `String!`

Request | Response | Python
cURL | h3-cli

```
query ScannerFile($uuid: String!) {
  scanner_file(uuid: $uuid) {
    ...ScannerFileFragment
  }
}

fragment ScannerFileFragment on ScannerFile {
  uuid
  vendor
  file_name
  file_size_bytes
  file_size_formatted
  format
  status
  status_message
  status_progress
  uploading_started_at
  uploading_ended_at
  processing_started_at
  processing_ended_at
  expires_at
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.4 `scanner_classifications_csv_presigned_url`

Description

Generates or retrieves a CSV export of scanner classifications for the specified scanner file and returns a presigned URL to download it. The CSV contains risk intelligence data produced by correlating scanner findings with NodeZero pentest results. If a pre-generated CSV exists from the ETL pipeline, its URL is returned directly; otherwise the CSV is generated on demand.

Response

Returns a `String`

Arguments

`scanner_file_uuid` - `String!`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query ScannerClassificationsCsvPresignedUrl($scanner_file_uuid: String!) {
  scanner_classifications_csv_presigned_url(scanner_file_uuid: $scanner_file_uuid)
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.5 `upload_scanner_file`

Description

Initiates the upload of a vulnerability scanner file. Returns a UploadScannerFileOutput containing a presigned URL. Use the presigned URL with an HTTP `PUT` request to upload the file content. After upload completes, call process_scanner_file to begin risk intelligence analysis.

Response

Returns an `UploadScannerFileOutput!`

Arguments

`input` - `UploadScannerFileInput!`

Request | Response | Python
cURL | h3-cli

```
mutation UploadScannerFile($input: UploadScannerFileInput!) {
  upload_scanner_file(input: $input) {
    scanner_file_uuid
    upload_url
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.6 `delete_scanner_files`

Description

Permanently deletes one or more vulnerability scanner files and their associated data from both storage and the database. Returns the count of files successfully deleted.

Response

Returns a `DeleteScannerFilesOutput!`

Arguments

`input` - `DeleteScannerFilesInput!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteScannerFiles($input: DeleteScannerFilesInput!) {
  delete_scanner_files(input: $input) {
    deleted_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.7 `process_scanner_file`

Description

Triggers risk intelligence processing for a vulnerability scanner file. Processing imports the scanner data and cross-references it with your NodeZero pentest results to produce ScannerClassification records. The file must be in `PENDING` or `FAILED` status, or in `COMPLETED` status with the `force` flag set to `true` for reprocessing. Returns an error if the file is currently being processed or has expired.

Response

Returns a `ProcessScannerFileOutput!`

Arguments

`input` - `ProcessScannerFileInput!`

Request	Response	Python
cURL	h3-cli	

```
mutation ProcessScannerFile($input: ProcessScannerFileInput!) {
  process_scanner_file(input: $input) {
    success
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.8 `update_scanner_file`

Description

Updates the metadata of a vulnerability scanner file, such as its display name. Returns the updated ScannerFile.

Response

Returns an `UpdateScannerFileOutput!`

Arguments

`input` - `UpdateScannerFileInput!`

Request | Response | Python

cURL | h3-cli

```
mutation UpdateScannerFile($input: UpdateScannerFileInput!) {
  update_scanner_file(input: $input) {
    success
    scanner_file {
      ...ScannerFileFragment
    }
  }
}

fragment ScannerFileFragment on ScannerFile {
  uuid
  vendor
  file_name
  file_size_bytes
  file_size_formatted
  format
  status
  status_message
  uploading_started_at
  processing_started_at
  processing_ended_at
  expires_at
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.9 DeleteScannerFilesInput

Description

Input for the delete_scanner_files mutation.

Fields

`scanner_file_uuids` - `[String!]!`
List of scanner file UUIDs to delete. Files that are currently processing or belong to another client account are skipped.

Request

```
mutation DeleteScannerFilesExample(
  $input: DeleteScannerFilesInput = {
    scanner_file_uuids: [
      "1234abcd-1234-abcd-1234-abcd1234abcd"
      "5678abcd-5678-abcd-5678-abcd5678abcd"
    ]
  }
) {
  delete_scanner_files(input: $input) {
    deleted_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.10 DeleteScannerFilesOutput

Description

Output from the delete_scanner_files mutation.

Fields

`deleted_count` - `Int!`
The number of scanner files that were successfully deleted.

Request Response

```
fragment DeleteScannerFilesOutputFragment on DeleteScannerFilesOutput {
  deleted_count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.11 ProcessScannerFileInput

Description

Input for the process_scanner_file mutation.

Fields

`scanner_file_uuid` - `String!`
UUID of the scanner file to process.
`force` - `Boolean`
Set to `true` to reprocess a file that has already completed processing. Required when the file status is `COMPLETED`.

Request

```
mutation ProcessScannerFileExample(
  $input: ProcessScannerFileInput = {
    scanner_file_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    force: false
  }
) {
  process_scanner_file(input: $input) {
    success
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.12 ProcessScannerFileOutput

Description

Output from the process_scanner_file mutation.

Fields

`success` - `Boolean!`
True if the processing request was successfully submitted. The file transitions to `PROCESSING` status while risk intelligence analysis runs asynchronously.

Request Response

```
fragment ProcessScannerFileOutputFragment on ProcessScannerFileOutput {
  success
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.13 ScannerCategory

Description

Raw classification categories assigned to vulnerability scanner findings. A single classification may have multiple categories. See ScannerDisplayCategory for the prioritized display version used in UI presentation.

Values

`CONFIRMED_EXPLOITABLE`
NodeZero confirmed exploitation of this vulnerability during a pentest.
`CONTEXTUALLY_EXPLOITABLE`
NodeZero did not directly exploit this vulnerability, but achieved remote code execution on the target asset. Informational-severity vulnerabilities are excluded.
`DETECTED_ON_HIGH_VALUE_TARGET`
The vulnerability was detected on a high-value target asset (e.g. domain controller, database server).
`THREAT_ACTOR_DETECTED`
The vulnerability is associated with known threat actors.
`VULNERABLE_NOT_EXPLOITED`
The scanner reported this vulnerability but NodeZero did not exploit it during pentesting.
`ASSET_NOT_DISCOVERED`
The scanner found this vulnerability on an asset that NodeZero did not discover.
`MITIGATED`
The vulnerability was previously exploited by NodeZero but was found to be remediated in a subsequent pentest.

Example

```
"CONFIRMED_EXPLOITABLE"
```

Platform Features: Vuln Risk Intelligence

8.42.14 ScannerCategoryBreakdown

Description

Aggregate classification counts grouped by composite risk categories. These categories combine multiple ScannerDisplayCategory values for higher-level analysis.

Fields

`exploitable` - `Int!`
Total classifications that are confirmed or contextually exploitable.
`not_exploitable` - `Int!`
Total classifications that are not confirmed or contextually exploitable.
`hvt_or_threat_actor_not_exploitable` - `Int!`
Classifications on high-value targets or with threat actor association that are not exploitable (`NOT_EXPLOITED_ON_HVT` or `NOT_EXPLOITED_THREAT_ACTOR`).
`scanner_confirmed_exploitable` - `Int!`
Classifications confirmed exploitable that were also reported by the scanner. Excludes vulnerabilities found only by NodeZero.
`priority` - `Int!`
Total priority classifications: exploitable, on a high-value target, or associated with a threat actor.

Request Response

```
fragment ScannerCategoryBreakdownFragment on ScannerCategoryBreakdown {
  exploitable
  not_exploitable
  hvt_or_threat_actor_not_exploitable
  scanner_confirmed_exploitable
  priority
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.15 ScannerCategoryCount

Description

The number of classifications assigned to a specific ScannerCategory. Since classifications may belong to multiple categories, these counts may not sum to the total.

Fields

`category` - `ScannerCategory!`
The raw category being counted.
`count` - `Int!`
Number of classifications that include this category.

Request Response

```
fragment ScannerCategoryCountFragment on ScannerCategoryCount {
  category
  count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.16 ScannerClassification

Description

Represents a single vulnerability classification produced by cross-referencing a third-party scanner finding with NodeZero pentest results. Each classification captures the correlation between what the scanner reported and what NodeZero discovered, enabling prioritized remediation based on confirmed exploitability, high-value target exposure, and threat actor association.

Fields

`uuid` - `ID!`
Unique identifier for this classification.
`vuln_scanner_file_uuid` - `ID!`
UUID of the ScannerFile this classification belongs to.
`vuln_scanner_file_data_uuid` - `ID`
UUID of the underlying scanner file data record. Null if this vulnerability was only found by NodeZero.
`vuln_scanner_asset_uuid` - `ID`
UUID of the scanner-reported asset. Null if this vulnerability was only found by NodeZero.
`finding_evidence_edge_tab_uuid` - `ID`
UUID of the matched NodeZero finding. Null if not found by NodeZero.
`categories` - `[ScannerCategory!]!`
All applicable ScannerCategory values for this classification. A classification may belong to multiple categories.
`display_category` - `ScannerDisplayCategory!`
The single highest-priority ScannerDisplayCategory for this classification, suitable for UI display.
`weakness_name` - `String`
Name of the vulnerability, sourced from NodeZero findings when available, otherwise from the scanner data.
`weakness_id` - `String`
Vulnerability identifier such as a CVE ID, NodeZero vulnerability ID, or vendor-specific ID.
`risk_score` - `Float`
Risk score on a 0-10 scale. Uses the NodeZero context score when available; falls back to the scanner-reported CVSS score.
`risk_severity` - `String`
Severity label derived from the `risk_score`: `CRITICAL`, `HIGH`, `MEDIUM`, `LOW`, or `INFO`.
`is_reported_by_scanner` - `Boolean!`
True if the third-party vulnerability scanner reported this finding.
`is_reported_by_nodezero` - `Boolean!`
True if NodeZero discovered this vulnerability during a pentest.
`weakness_series_uuid` - `ID`
UUID of the associated weakness series for cross-operation tracking. Null if not matched to a NodeZero finding.
`host_tab_uuid` - `ID`
UUID of the host where this vulnerability was identified.
`asset_uuid` - `ID`
UUID of the asset for cross-operation tracking.
`scanned_ip` - `String!`
IP address of the asset as reported by the vulnerability scanner.
`scanned_host` - `String`
Hostname of the asset as reported by the vulnerability scanner.
`is_high_value_target` - `Boolean!`
True if this vulnerability was detected on a high-value target asset (e.g. domain controller, database server).
`threat_actor_ids` - `[String!]`
IDs of threat actors known to exploit this vulnerability.
`threat_actors` - `[ThreatActor!]`
Threat actor records associated with this vulnerability.
`downstream_impact_types` - `[String!]`

Types of downstream impacts that could result from this vulnerability (e.g. data exfiltration, lateral movement).

`downstream_impact_types_and_counts` - `[String!]`

Downstream impact types with associated counts, formatted as `Type: N`.

`business_risks` - `[String!]`

Business risk categories associated with this vulnerability.

`nodezero_asset` - `Asset`

The NodeZero asset (tracked across operations) matched to this scanner-reported asset.

`weakness_series` - `WeaknessSeries`

The NodeZero weakness series this classification was correlated with, or that was found only by NodeZero on the scanned asset.

Request Response

```
fragment ScannerClassificationFragment on ScannerClassification {
  uuid
  vuln_scanner_file_uuid
  categories
  display_category
  weakness_name
  weakness_id
  risk_score
  risk_severity
  is_reported_by_scanner
  is_reported_by_nodezero
  scanned_ip
  scanned_host
  is_high_value_target
  host_tab_uuid
  asset_uuid
  weakness_series_uuid
  downstream_impact_types
  business_risks
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.17 ScannerClassificationFacets

Description

Faceted filter values for scanner classifications. Each facet provides the distinct values present in the current (optionally filtered) dataset along with counts, enabling dynamic filter UIs. See FacetValue.

Fields

`page_info` - `PageInfo`
Pagination metadata reflecting any filters applied to the facet query. See PageInfo.
`threat_actor_names` - `[FacetValue]`
Distinct threat actor names present in the current dataset, with counts. Each FacetValue includes a prebuilt filter for direct use with PageInput.
`business_risk_names` - `[FacetValue]`
Distinct business risk names present in the current dataset, with counts. Each FacetValue includes a prebuilt filter for direct use with PageInput.

Request Response

```
fragment ScannerClassificationFacetsFragment on ScannerClassificationFacets {
  threat_actor_names {
    label
    value
    value_count
  }
  business_risk_names {
    label
    value
    value_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.18 ScannerClassificationPage

Description

Paginated response containing ScannerClassification records.

Fields

`classifications` - `[ScannerClassification!]!`
The list of scanner classification records for the current page.
`page_info` - `PageInfo!`
Pagination metadata for the current page. See PageInfo.

Request Response

```
fragment ScannerClassificationPageFragment on ScannerClassificationPage {
  classifications {
    uuid
    display_category
    weakness_name
    weakness_id
    risk_score
    risk_severity
    is_reported_by_scanner
    is_reported_by_nodezero
    scanned_ip
    scanned_host
    is_high_value_target
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.19 ScannerClassificationsSummary

Description

Aggregate statistics and breakdowns for scanner classifications within a ScannerFile. Provides multiple views into the data for building dashboards and summary reports.

Fields

`counts_by_display_category` - `[ScannerDisplayCategoryCount!]!`
Classification counts grouped by ScannerDisplayCategory. These counts are mutually exclusive and sum to the total.
`counts_by_category` - `[ScannerCategoryCount!]!`
Classification counts grouped by raw ScannerCategory. Since a classification can have multiple categories, these counts may exceed the total.
`counts_by_severity` - `[ScannerSeverityCount!]!`
Classification counts grouped by ScannerSeverity. These counts are mutually exclusive and sum to the total.
`source_breakdown` - `ScannerSourceBreakdown!`
Breakdown showing overlap between scanner and NodeZero findings. See ScannerSourceBreakdown.
`category_breakdown` - `ScannerCategoryBreakdown!`
Breakdown by composite risk categories (exploitable, priority, etc.). See ScannerCategoryBreakdown.
`counts_by_downstream_impact` - `[ScannerDownstreamImpactCount!]!`
Classification counts grouped by downstream impact type.

Request Response

```
fragment ScannerClassificationsSummaryFragment on ScannerClassificationsSummary {
  counts_by_display_category {
    display_category
    count
  }
  counts_by_severity {
    severity
    count
  }
  source_breakdown {
    found_by_scanner
    found_by_nodezero
    found_by_both
    found_by_scanner_only
    found_by_nodezero_only
  }
  category_breakdown {
    exploitable
    not_exploitable
    priority
  }
  counts_by_downstream_impact {
    impact_type
    count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.20 ScannerDisplayCategory

Description

Priority-ordered display category for scanner classifications. Each classification is assigned exactly one display category (the highest-priority one that applies), making this suitable for mutually exclusive UI presentation. Ordered from highest to lowest severity.

Values

`CONFIRMED_EXPLOITABLE`
NodeZero confirmed exploitation of this vulnerability during a pentest.
`CONTEXTUALLY_EXPLOITABLE`
NodeZero achieved remote code execution on the target asset without directly exploiting this vulnerability. Informational-severity vulnerabilities are excluded.
`NOT_EXPLOITED_ON_HVT`
Not exploited by NodeZero, but detected on a high-value target asset.
`NOT_EXPLOITED_THREAT_ACTOR`
Not exploited by NodeZero, but associated with known threat actor activity.
`ASSET_NOT_DISCOVERED`
The scanner found this vulnerability on an asset that NodeZero did not discover.
`VULNERABLE_NOT_EXPLOITED`
The scanner reported this vulnerability but NodeZero did not exploit it.
`MITIGATED`
The vulnerability was previously exploited by NodeZero but was found to be remediated in a subsequent pentest.

Example

```
"CONFIRMED_EXPLOITABLE"
```

Platform Features: Vuln Risk Intelligence

8.42.21 ScannerDisplayCategoryCount

Description

The number of classifications assigned to a specific ScannerDisplayCategory.

Fields

`display_category` - `ScannerDisplayCategory!`
The display category being counted.

`count` - `Int!`
Number of classifications in this display category.

Request Response

```
fragment ScannerDisplayCategoryCountFragment on ScannerDisplayCategoryCount {
  display_category
  count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.22 ScannerDownstreamImpactCount

Description

The number of distinct classifications associated with a specific downstream impact type (e.g. data exfiltration, lateral movement).

Fields

`impact_type` - `String!`
The downstream impact type name.
`count` - `Int!`
Number of distinct classifications associated with this downstream impact type.

Request Response

```
fragment ScannerDownstreamImpactCountFragment on ScannerDownstreamImpactCount {
  impact_type
  count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.23 ScannerFile

Description

Represents an uploaded vulnerability scanner file from a third-party vendor (Qualys, Tenable, or Rapid7). After upload and processing, the file's scanner findings are cross-referenced with NodeZero pentest results to produce ScannerClassification records for risk intelligence analysis.

Fields

`uuid` - `String!`
Unique identifier for this scanner file.
`vendor` - `String!`
The vulnerability scanner vendor that produced this file (e.g. `Qualys`, `Tenable`, `Rapid7`). See ScannerFileVendor.
`file_name` - `String!`
The display name of the scanner file. Defaults to a generated name based on vendor and upload timestamp if not specified during upload.
`file_size_bytes` - `Float`
The raw file size in bytes.
`file_size_formatted` - `String`
Human-readable file size (e.g. `2.4 MB`).
`format` - `ScannerFileFormat`
The detected file format. See ScannerFileFormat.
`status` - `ScannerFileStatus!`
Current processing status of the scanner file. See ScannerFileStatus.
`status_message` - `String`
A detailed message providing additional context about the current `status`, such as error details for `FAILED` files.
`status_progress` - `Float`
Processing progress indicator from 0 to 100. Only populated when the file is in `PROCESSING` status.
`uploading_started_at` - `Datetime!`
Timestamp when the file upload was initiated.
`uploading_ended_at` - `Datetime`
Timestamp when the file upload completed.
`processing_started_at` - `Datetime`
Timestamp when risk intelligence processing began.
`processing_ended_at` - `Datetime`
Timestamp when risk intelligence processing completed.
`expires_at` - `Datetime`
Timestamp when this scanner file expires and can no longer be processed.
`classifications_count` - `Int!`
Total count of ScannerClassification records for this file. Accepts an optional PageInput to apply filters before counting.

Arguments

`page_input` - `PageInput`
`classifications_page` - `ScannerClassificationPage!`
Paginated list of ScannerClassification records for this file.

Arguments

`page_input` - `PageInput`
`classifications_summary` - `ScannerClassificationsSummary!`
Aggregate statistics and breakdowns for the classifications in this file. See ScannerClassificationsSummary.

Arguments

`page_input` - `PageInput`

`classifications_facets` - `ScannerClassificationFacets!`

Available filter values with counts for classification faceted navigation. See ScannerClassificationFacets.

Arguments

`page_input` - `PageInput`

Request Response

```
fragment ScannerFileFragment on ScannerFile {
  uuid
  vendor
  file_name
  file_size_bytes
  file_size_formatted
  format
  status
  status_message
  status_progress
  uploading_started_at
  uploading_ended_at
  processing_started_at
  processing_ended_at
  expires_at
  classifications_count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.24 ScannerFileFormat

Description

Supported file formats for uploaded vulnerability scanner files.

Values

`CSV`
Comma-separated values format.
`XML`
Extensible Markup Language format.

Example

```
"CSV"
```

Platform Features: Vuln Risk Intelligence

8.42.25 ScannerFileStatus

Description

Lifecycle status of a vulnerability scanner file as it progresses through upload and risk intelligence processing.

Values

`PENDING`
The file has been uploaded but risk intelligence processing has not yet been initiated. Call process_scanner_file to begin.
`PROCESSING`
Risk intelligence processing is currently in progress.
`FAILED`
Risk intelligence processing encountered an error. The file can be reprocessed.
`COMPLETED`
Risk intelligence processing completed successfully. Classifications are available.
`EXPIRED`
The file has passed its expiration date and can no longer be processed.
`DUPLICATE`
The file was identified as a duplicate of an existing upload.

Example

```
"PENDING"
```

Platform Features: Vuln Risk Intelligence

8.42.26 ScannerFileVendor

Description

Supported third-party vulnerability scanner vendors for BYO Scanner file uploads.

Values

`Qualys`
Qualys vulnerability scanner.
`Tenable`
Tenable vulnerability scanner.
`Rapid7`
Rapid7 vulnerability scanner.

Example

```
"Qualys"
```

Platform Features: Vuln Risk Intelligence

8.42.27 ScannerFilesPage

Description

Paginated response containing ScannerFile records.

Fields

`page_info` - `PageInfo`
Pagination metadata for the current page. See PageInfo.
`scanner_files` - `[ScannerFile!]!`
The list of scanner files for the current page.

Request Response

```
fragment ScannerFilesPageFragment on ScannerFilesPage {
  scanner_files {
    uuid
    vendor
    file_name
    file_size_formatted
    format
    status
    status_message
    uploading_started_at
    processing_ended_at
    expires_at
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.28 ScannerSeverity

Description

Severity levels for scanner classification risk scores, based on standard CVSS score ranges.

Values

`CRITICAL`
Critical severity. Risk score is 9.0 or higher.
`HIGH`
High severity. Risk score is between 7.0 and 8.9.
`MEDIUM`
Medium severity. Risk score is between 4.0 and 6.9.
`LOW`
Low severity. Risk score is between 0.1 and 3.9.
`INFO`
Informational. Risk score is 0 or below.
`UNKNOWN`
Unknown severity. No risk score is available.

Example

```
"CRITICAL"
```

Platform Features: Vuln Risk Intelligence

8.42.29 ScannerSeverityCount

Description

The number of classifications at a specific ScannerSeverity level.

Fields

`severity` - `ScannerSeverity!`
The severity level being counted.
`count` - `Int!`
Number of classifications at this severity level.

Request Response

```
fragment ScannerSeverityCountFragment on ScannerSeverityCount {
  severity
  count
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.30 ScannerSourceBreakdown

Description

Breakdown of classification counts by detection source, showing the overlap between the third-party vulnerability scanner and NodeZero pentest findings.

Fields

`found_by_scanner` - `Int!`
Total classifications found by the scanner, including those also found by NodeZero.
`found_by_nodezero` - `Int!`
Total classifications found by NodeZero, including those also found by the scanner.
`found_by_both` - `Int!`
Classifications found by both the scanner and NodeZero.
`found_by_scanner_only` - `Int!`
Classifications found exclusively by the scanner and not by NodeZero.
`found_by_nodezero_only` - `Int!`
Classifications found exclusively by NodeZero and not by the scanner.

Request Response

```
fragment ScannerSourceBreakdownFragment on ScannerSourceBreakdown {
  found_by_scanner
  found_by_nodezero
  found_by_both
  found_by_scanner_only
  found_by_nodezero_only
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.31 UpdateScannerFileInput

Description

Input for the update_scanner_file mutation.

Fields

`scanner_file_uuid` - `String!`
UUID of the scanner file to update.
`file_name` - `String`
New display name to assign to the scanner file.

Request

```
mutation UpdateScannerFileExample(
  $input: UpdateScannerFileInput = {
    scanner_file_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    file_name: "qualys-scan-2024-01-renamed.csv"
  }
) {
  update_scanner_file(input: $input) {
    success
    scanner_file {
      uuid
      file_name
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.32 UpdateScannerFileOutput

Description

Output from the update_scanner_file mutation.

Fields

`success` - `Boolean!`
True if the scanner file was successfully updated.
`scanner_file` - `ScannerFile!`
The updated ScannerFile with the new metadata applied.

Request Response

```
fragment UpdateScannerFileOutputFragment on UpdateScannerFileOutput {
  success
  scanner_file {
    uuid
    vendor
    file_name
    status
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.33 UploadScannerFileInput

Description

Input for the upload_scanner_file mutation.

Fields

`vendor` - `ScannerFileVendor!`
The vendor that produced the scanner file. See ScannerFileVendor.

`file_name` - `String`
Optional display name for the file. If omitted, a name is generated from the vendor and upload timestamp.

`file_size_in_bytes` - `Float`
The file size in bytes. Used for display purposes.

Request

```
mutation UploadScannerFileExample(
  $input: UploadScannerFileInput = {
    vendor: Qualys
    file_name: "qualys-scan-2024-01.csv"
    file_size_in_bytes: 2048576
  }
) {
  upload_scanner_file(input: $input) {
    scanner_file_uuid
    upload_url
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Vuln Risk Intelligence

8.42.34 UploadScannerFileOutput

Description

Output from the upload_scanner_file mutation.

Fields

`scanner_file_uuid` - `String!`
Unique identifier assigned to the newly created scanner file record.
`upload_url` - `String!`
Presigned URL for uploading the file content. Issue an HTTP `PUT` request to this URL with the file data in the request body.

Request Response

```
fragment UploadScannerFileOutputFragment on UploadScannerFileOutput {
  scanner_file_uuid
  upload_url
}
```

Examples may not include all available fields for a type.

8.43 Platform Features: CSV Format

These types document the format of CSV exports available from various queries in the API.

8.43.1 ActionLogCSV

Description

`String` scalar type representing a ActionLogCSV row with columns:

- IP: String
- ModuleName: String
- Cmd: String
- StartTime: String
- EndTime: String
- Targets: String
- MitreTactics: String
- OpID: String

Example

```
ActionLogCSV
```

Platform Features: CSV Format

8.43.2 ActiveDirPasswordCSV

Description

`String` scalar type representing a ActiveDirPasswordCSV row with columns:

- Score: Float
- Severity: String
- Username: String
- UserStatus: String
- HashMasked: String
- CleartextMasked: String
- PasswordLength: String
- Cracked: Boolean
- CrackMethod: String
- CrackMethodLabel: String
- PasswordRank: String
- PasswordRankLabel: String
- UsersWithSimilarPassword: String
- NumSimilarPassword: Int
- PwdLastSet: String
- LDAPDomainName: String
- Hostname: String
- IP: String
- OpID: String

Example

```
ActiveDirPasswordCSV
```

Platform Features: CSV Format

8.43.3 CertificateCSV

Description

`String` scalar type representing a CertificateCSV row with columns:

- FirstSeen: Datetime
- Vhost: String
- IP: String
- Port: Int
- Expiration: String
- Issuer: String
- CommonName: String
- AltNames: String
- AltNamesList: String
- Signed: String
- OpID: String

Example

```
CertificateCSV
```

Platform Features: CSV Format

8.43.4 CredentialCSV

Description

`String` scalar type representing a CredentialCSV row with columns:

- FirstSeen: Datetime
- CredType: String
- CredSource: String
- UserName: String
- Cleartext: String
- Role: String
- Confirmed: Boolean
- Hostname: String
- VhostName: String
- IP: String
- Port: Int
- Protocol: String
- IPAddresses: String
- ServiceType: String
- Service: String
- LDAPDomain: String
- AzureDomain: String
- UserGroups: String
- UserCloudRoles: String
- LoginFormUrl: String
- OS: String
- Hardware: String
- Device: String
- Product: String
- Permissions: String
- Injected: String
- Cracked: Boolean
- Anon: Boolean
- Hash: String
- HashType: String
- RiskScore: Float
- RiskScoreDescription: String
- OpID: String
- ProvenEntityEid: String
- BusinessRisks: String

Example

```
CredentialCSV
```

Platform Features: CSV Format

8.43.5 DatabaseRepoCSV

Description

`String` scalar type representing a DatabaseRepoCSV row with columns:

- UUID: String
- EntityType: String
- FirstSeen: Datetime
- Name: String
- Service: String
- ServiceType: String
- Product: String
- IP: String
- Uri: String
- Hostname: String
- ProtocolPort: String
- NumDataResources: Int
- PermissionsList: String
- Permissions: String
- Authenticated: Boolean
- Score: Float
- Severity: String
- ContextScoreDescription: String
- OpID: String

Example

```
DatabaseRepoCSV
```

Platform Features: CSV Format

8.43.6 DockerRegistryCSV

Description

`String` scalar type representing a DockerRegistryCSV row with columns:

- UUID: String
- EntityType: String
- FirstSeen: Datetime
- Name: String
- Service: String
- ServiceType: String
- Product: String
- IP: String
- Hostname: String
- ProtocolPort: String
- NumDataResources: Int
- PermissionsList: String
- Permissions: String
- Authenticated: Boolean
- Score: Float
- Severity: String
- ContextScoreDescription: String
- OpID: String

Example

```
DockerRegistryCSV
```

Platform Features: CSV Format

8.43.7 EDROverviewHostCSV

Description

`String` scalar type representing a EDROverviewHostCSV row with columns:

- UUID: String
- Score: Float
- Severity: String
- IP: String
- HostNames: String
- EDRVendors: String
- UnblockedEvents: Int
- MitreAttackTactics: String
- DownstreamImpacts: String
- IsDomainController: Boolean
- IsPublic: Boolean
- IsDatabaseServer: Boolean
- IsLoadBalancer: Boolean
- IsMailServer: Boolean
- IsVPN: Boolean
- IsWAF: Boolean
- IsK8sNode: Boolean
- IsK8sService: Boolean
- IsK8sPod: Boolean

Example

```
EDROverviewHostCSV
```

Platform Features: CSV Format

8.43.8 ExternalDomainCSV

Description

`String` scalar type representing a ExternalDomainCSV row with columns:

- Name: String
- TldName: String
- Reachable: Boolean
- InScope: String
- CNAMEs: String
- CnamesList: String
- Ips: String
- IpsList: String
- CloudProviders: String
- CloudProvidersList: String
- CloudServices: String
- CloudServicesList: String
- CloudRegions: String
- CloudRegionsList: String
- Organizations: String
- OrganizationsList: String
- Isps: String
- IspsList: String
- Asns: String
- AsnsList: String
- Countries: String
- CountriesList: String
- DNSHostnames: String
- DNSHostnamesList: String
- CertificateIssuers: String
- CertificateIssuersList: String
- OpID: String

Example

```
ExternalDomainCSV
```

Platform Features: CSV Format

8.43.9 ExternalDomainXopCSV

Description

`String` scalar type representing a ExternalDomainXopCSV row with columns:

- Name: String
- IP: String
- Status: String
- IpType: String
- Reachable: Boolean
- Isp: String
- Organization: String
- LastSeen: String
- FirstSeen: String
- AssetType: String
- Aliases: String
- Ips: String
- TldInScope: Boolean

Example

```
ExternalDomainXopCSV
```

Platform Features: CSV Format

8.43.10 FileShareCSV

Description

`String` scalar type representing a FileShareCSV row with columns:

- UUID: String
- EntityType: String
- FirstSeen: Datetime
- Name: String
- Service: String
- ServiceType: String
- Product: String
- IP: String
- Hostname: String
- ProtocolPort: String
- NumDataResources: Int
- PermissionsList: String
- Permissions: String
- Authenticated: Boolean
- Score: Float
- Severity: String
- ContextScoreDescription: String
- OpID: String

Example

```
FileShareCSV
```

Platform Features: CSV Format

8.43.11 GitRepoCSV

Description

`String` scalar type representing a GitRepoCSV row with columns:

- UUID: String
- EntityType: String
- Name: String
- GitAccountName: String
- GitAccountType: String
- GitAccountSource: String
- NumGitRepoSensitiveFindings: Int
- Score: Float
- Severity: String
- OpID: String

Example

```
GitRepoCSV
```

Platform Features: CSV Format

8.43.12 HostCSV

Description

`String` scalar type representing a HostCSV row with columns:

- FirstSeen: Datetime
- Subnet: String
- SubnetSource: String
- IP: String
- Hostname: String
- DNSHostnames: String
- LDAPHostname: String
- InScope: Boolean
- OS: String
- Hardware: String
- Device: String
- NumWeaknesses: Int
- NumConfirmedWeaknesses: Int
- NumDataResources: Int
- NumCredentials: Int
- NumConfirmedCredentials: Int
- NumServices: Int
- NumWebShares: Int
- RiskScore: Float
- RiskScoreDescription: String
- OpID: String
- BusinessRisks: String

Example

```
HostCSV
```

Platform Features: CSV Format

8.43.13 HostXopCSV

Description

`String` scalar type representing a HostXopCSV row with columns:

- IP: String
- Status: String
- Domains: String
- ThirdPartyAliases: String
- AssetType: String
- Isp: String
- Organization: String
- LastSeen: Datetime
- FirstSeen: String
- Asn: String
- OpID: String

Example

```
HostXopCSV
```

Platform Features: CSV Format

8.43.14 ImpactCSV

Description

`String` scalar type representing a ImpactCSV row with columns:

- Score: Float
- Severity: String
- ImpactType: String
- Name: String
- AffectedAsset: String
- IP: String
- Hostname: String
- TimeToCompromise: String
- BusinessRisks: String

Example

```
ImpactCSV
```

Platform Features: CSV Format

8.43.15 RapidResponseAssetCSV

Description

`String` scalar type representing a RapidResponseAssetCSV row with columns:

- RapidResponseName: String
- RapidResponseShortName: String
- RapidResponsePublishedAt: Datetime
- Hostname: String
- IPs: String
- DNSHostnames: String
- OSNames: String
- Status: String
- ExposureOpenedAt: Datetime
- WeaknessIDs: String
- AffectedServices: String
- ServiceTypes: String
- Products: String
- URLs: String
- AffectedApplications: String
- FixActions: String
- References: String

The grain is one row per affected host on the RapidResponse.

- `WeaknessIDs` is the host's weakness identifier list with the H3 → CVE alias swap baked in: each id in `RapidResponseAsset.vuln_ids` is replaced by its first `CVE-` alias from `vuln.aliases` when one exists, and falls back to the raw id otherwise.
- `AffectedServices` / `ServiceTypes` / `Products` / `URLs` / `AffectedApplications` are distinct values rolled up across the host's status-bearing RapidResponseEntities (every observation entity with a non-null status — today that's CONFIRMED_EXPLOITABLE / NOT_EXPLOITABLE; tracking entities with status=NULL never contribute to the rollups). `URLs` is only populated for HTTP/HTTPS services.
- `FixActions` / `References` are distinct `vuln.mitigations[].description` and `vuln.references[].url` values rolled up across the host's `vuln_ids`, mirroring the WeaknessCSV `Mitigations` / `References` columns.
- List-shaped cells use `'; '` (semicolon-space) as the within-cell separator so they never collide with the CSV's own comma field separator.

Example

```
RapidResponseAssetCSV
```

Platform Features: CSV Format

8.43.16 S3BucketCSV

Description

`String` scalar type representing a S3BucketCSV row with columns:

- UUID: String
- EntityType: String
- Name: String
- AwsAccountID: String
- CloudServiceName: String
- CloudProviderName: String
- ServiceType: String
- NumDataResources: Int
- PermissionsList: String
- AwsAnonymousUserPermissionsList: String
- AwsCrossAccountUserPermissionsList: String
- Score: Float
- Severity: String
- OpID: String

Example

```
S3BucketCSV
```

Platform Features: CSV Format

8.43.17 ScannerClassificationCSV

Description

`String` scalar type representing a ScannerClassificationCSV row with columns:

- UUID: String
- WeaknessName: String
- WeaknessID: String
- RiskScore: Float
- RiskSeverity: String
- Categories: String
- IP: String
- Host: String
- Status: String
- MarkedAs: String
- DownstreamImpactTypes: String
- BusinessRisks: String
- AttackPath: Int
- ThreatActors: String
- Mitigations: String
- References: String

Example

```
ScannerClassificationCSV
```

Platform Features: CSV Format

8.43.18 ServiceCSV

Description

`String` scalar type representing a ServiceCSV row with columns:

- FirstSeen: Datetime
- Hostname: String
- IP: String
- Port: Int
- Protocol: String
- Service: String
- ServiceType: String
- Product: String
- OS: String
- Hardware: String
- Device: String
- LDAPHostname: String
- VhostNames: String
- NumWeaknesses: Int
- NumCredentials: Int
- NumDataResources: Int
- RiskScore: Float
- RiskScoreDescription: String
- NonStandardPort: Boolean
- Enumerated: Boolean
- DNSServer: Boolean
- DomainController: Boolean
- MailServer: Boolean
- OpID: String

Example

```
ServiceCSV
```

Platform Features: CSV Format

8.43.19 ShareCSV

Description

`String` scalar type representing a ShareCSV row with columns:

- Score: Float
- Severity: String
- DataStoreType: String
- ServiceType: String
- Address: String
- Account: String
- IP: String
- Port: Int
- Protocol: String
- Hostname: String
- DNSAddress: String
- ARN: String
- Permissions: String
- NumDataResources: Int
- NumSensitiveResources: Int
- NumSensitiveDataItem: Int
- SensitiveDataItemTypes: String
- Authenticated: Boolean
- CreatedAt: Datetime
- OpID: String
- BusinessRisks: String

Example

```
ShareCSV
```

Platform Features: CSV Format

8.43.20 TicketCSV

Description

`String` scalar type representing a TicketCSV row with columns:

- UUID: String
- AssignedBy: String
- Status: String
- Severity: String
- TicketName: String
- WeaknessID: String
- WeaknessName: String
- AssetCount: Int
- CreatedAt: Datetime
- LastUpdated: Datetime
- SyncSystem: String
- SyncDate: Datetime
- ExternalID: String

Example

```
TicketCSV
```

Platform Features: CSV Format

8.43.21 TripwireCSV

Description

`String` scalar type representing a TripwireCSV row with columns:

- TripwireName: String
- TripwireType: String
- FileName: String
- MonitoredProcesses: String
- Location: String
- Path: String
- Status: String
- DateDropped: String

Example

```
TripwireCSV
```

Platform Features: CSV Format

8.43.22 UserCSV

Description

`String` scalar type representing a UserCSV row with columns:

- Name: String
- Compromised: Boolean
- Source: String
- Hostname: String
- VhostName: String
- IP: String
- Role: String
- LocalAdmin: Boolean
- DomainAdmin: Boolean
- LDAPDomainName: String
- AzureDomainName: String
- GroupNames: String
- CloudRoleNames: String
- OpID: String

Example

```
UserCSV
```

Platform Features: CSV Format

8.43.23 WeaknessCSV

Description

`String` scalar type representing a WeaknessCSV row with columns:

- WeaknessID: String
- FirstSeen: Datetime
- Name: String
- RootCause: String
- Severity: String
- ContextScore: Float
- ContextScoreDescription: String
- Confirmed: Boolean
- Hostname: String
- CNAME: String
- OS: String
- IP: String
- Port: Int
- Protocol: String
- ProtocolPort: String
- Service: String
- ServiceType: String
- Product: String
- OpID: String
- Description: String
- AssetID: String
- AllHostnames: String
- Repo: String
- Vhost: String
- ResourceUri: String
- UserName: String
- UserDomainName: String
- ProvenEntityEid: String
- Impact: String
- Mitigations: String
- References: String
- ThreatActors: String
- BusinessRisks: String

Example

```
WeaknessCSV
```

Platform Features: CSV Format

8.43.24 WeaknessSeriesCSV

Description

`String` scalar type representing a WeaknessSeriesCSV row with columns:

- UUID: String
- WeaknessName: String
- WeaknessID: String
- AffectedAsset: String
- WeaknessCategory: String
- Status: String
- NoteStatus: String
- NoteText: String
- Severity: String
- Verifiable: Boolean
- LatestOpID: String
- IP: String
- FirstSeen: Datetime
- LastSeen: String

Example

```
WeaknessSeriesCSV
```

Platform Features: CSV Format

8.43.25 WebShareCSV

Description

`String` scalar type representing a WebShareCSV row with columns:

- FirstSeen: Datetime
- Vhost: String
- IP: String
- Port: Int
- Protocol: String
- Hostname: String
- OS: String
- Service: String
- ServiceType: String
- Product: String
- NumDataResources: Int
- OpID: String

Example

```
WebShareCSV
```

8.44 Platform Features: Inventory

8.44.1 `inventory_assets_csv_presigned_url`

Times out at the 30s API GW boundary at scale. UI / user-bound callers should switch to `request_inventory_assets_csv_export` for async builds with email delivery. Headless / API-key integrations have no recipient for the email and should continue using this query until a dedicated async-poll variant exists.

Description

Generates a CSV export of inventory assets and returns a temporary presigned URL for downloading the file. Supports exporting domains, IPs, or both.

Use `page_input` to export all rows matching active filters ("Download All"), or `uuids` to export specific selected assets ("Download Selected"). Optionally scope the export to a single `InventoryOpScope`.

Each unique input combination produces a separately cached CSV file. The presigned URL expires after a short time.

Response

Returns a `String`

Arguments

`input` - `InventoryAssetsCsvInput!`

Request | Response | Python
cURL | h3-cli

```
query InventoryAssetsCsvPresignedUrl($input: InventoryAssetsCsvInput!) {
  inventory_assets_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.2 `inventory_ip_ranges_csv_presigned_url`

Times out at the 30s API GW boundary at scale. UI / user-bound callers should switch to `request_inventory_ip_ranges_csv_export` for async builds with email delivery. Headless / API-key integrations have no recipient for the email and should continue using this query until a dedicated async-poll variant exists.

Description

Generates a CSV export of `InventoryCidr` records (the IP Ranges tab) and returns a temporary presigned URL for downloading the file.

Use `page_input` to export all rows matching active filters ("Download All"), or `uuids` to export specific selected IP ranges ("Download Selected"). Optionally scope the export to a single `InventoryOpScope`.

Each unique input combination produces a separately cached CSV file. The presigned URL expires after a short time.

Response

Returns a `String`

Arguments

`input` - `InventoryIpRangesCsvInput!`

Request	Response	Python
cURL	h3-cli	

```
query InventoryIpRangesCsvPresignedUrl(
  $input: InventoryIpRangesCsvInput!
) {
  inventory_ip_ranges_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.3 `inventory_scopes_csv_presigned_url`

Description

Generates a CSV export of `InventoryOpScope` records (the Scopes tab) and returns a temporary presigned URL for downloading the file.

Use `page_input` to export all rows matching active filters ("Download All"), or `uuids` to export specific selected scopes ("Download Selected").

Each unique input combination produces a separately cached CSV file. The presigned URL expires after a short time.

Response

Returns a `String`

Arguments

`input` - `InventoryScopesCsvInput!`

Request | Response | Python
cURL | h3-cli

```
query InventoryScopesCsvPresignedUrl(
  $input: InventoryScopesCsvInput!
) {
  inventory_scopes_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.4 `initiate_bulk_inventory_upload`

Description

Initiates a bulk inventory upload. Returns a presigned URL for uploading the file. After uploading the file via HTTP PUT, call `process_bulk_inventory_upload` to begin processing.

Response

Returns an `InitiateBulkInventoryUploadOutput!`

Arguments

`input` - `InitiateBulkInventoryUploadInput!`

Request | Response | Python
cURL | h3-cli

```
mutation InitiateBulkInventoryUpload(
  $input: InitiateBulkInventoryUploadInput!
) {
  initiate_bulk_inventory_upload(input: $input) {
    upload_uuid
    presigned_url
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.5 `process_bulk_inventory_upload`

Description

Triggers processing for a previously uploaded bulk inventory file. The upload must be in PENDING or FAILED status.

Response

Returns a `ProcessBulkInventoryUploadOutput!`

Arguments

`input` - `ProcessBulkInventoryUploadInput!`

Request | Response | Python
cURL | h3-cli

```
mutation ProcessBulkInventoryUpload(
  $input: ProcessBulkInventoryUploadInput!
) {
  process_bulk_inventory_upload(input: $input) {
    upload_uuid
    status
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.6 `create_inventory_cidr`

Description

Creates a single inventory CIDR from an input object. Returns the created `InventoryCidr` record.

Response

Returns an `InventoryCidr!`

Arguments

`input` - `CreateInventoryCidrInput!`

Request | Response | Python
cURL | h3-cli

```
mutation CreateInventoryCidr(
  $input: CreateInventoryCidrInput!
) {
  create_inventory_cidr(input: $input) {
    uuid
    cidr
    ownership_status
    asset_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.7 `update_inventory_cidr`

Description

Updates fields on a single inventory CIDR identified by UUID. Returns the updated `InventoryCidr`.

Response

Returns an `InventoryCidr!`

Arguments

`uuid` - `ID!`

`input` - `UpdateInventoryCidrInput!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateInventoryCidr(
  $uuid: ID!
  $input: UpdateInventoryCidrInput!
) {
  update_inventory_cidr(uuid: $uuid, input: $input) {
    uuid
    cidr
    ownership_status
    asset_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.8 `delete_inventory_cidr`

Description

Deletes a single inventory CIDR by UUID. Returns the count and details of deleted CIDRs in an `InventoryCidrDeleteOutput`.

Response

Returns an `InventoryCidrDeleteOutput!`

Arguments

`uuid` - `ID!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryCidr(
  $uuid: ID!
) {
  delete_inventory_cidr(uuid: $uuid) {
    deleted_count
    deleted_inventory_cidrs
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.9 `delete_inventory_cidrs`

Description

Deletes one or more inventory CIDRs by UUID. Returns the count and details of deleted CIDRs in an `InventoryCidrDeleteOutput`.

Response

Returns an `InventoryCidrDeleteOutput!`

Arguments

`uuids` - `[ID!]!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryCidrs(
  $uuids: [ID!]!
) {
  delete_inventory_cidrs(uuids: $uuids) {
    deleted_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.10 `request_inventory_assets_csv_export`

Description

Request an asynchronous CSV export of inventory assets (domains, IPs, or both). Returns the ETL job instance UUID immediately. The user receives an email with the download link when the build completes (or a failure notice if it does not). Identical in-flight requests from the same user are coalesced onto the existing job.

Replaces the synchronous `inventory_assets_csv_presigned_url` query, which times out at the 30s API Gateway boundary for large tenants.

Response

Returns an `ID!`

Arguments

`input` - `InventoryAssetsCsvInput!`

EXAMPLE

Query

```
mutation request_inventory_assets_csv_export($input: InventoryAssetsCsvInput!) {
  request_inventory_assets_csv_export(input: $input)
}
```

Variables

```
{"input": InventoryAssetsCsvInput}
```

Response

```
{
  "data": {
    "request_inventory_assets_csv_export": "4"
  }
}
```

Platform Features: Inventory

8.44.11 `request_inventory_ip_ranges_csv_export`

Description

Request an asynchronous CSV export of inventory IP ranges (CIDRs). Returns the ETL job instance UUID immediately. The user receives an email when the build completes (or a failure notice if it does not). Identical in-flight requests from the same user are coalesced onto the existing job.

Replaces the synchronous `inventory_ip_ranges_csv_presigned_url` query, which times out at the 30s API Gateway boundary for large tenants.

Response

Returns an `ID!`

Arguments

`input` - `InventoryIpRangesCsvInput!`

EXAMPLE

Query

```
mutation request_inventory_ip_ranges_csv_export($input: InventoryIpRangesCsvInput!) {
  request_inventory_ip_ranges_csv_export(input: $input)
}
```

Variables

```
{"input": InventoryIpRangesCsvInput}
```

Response

```
{
  "data": {
    "request_inventory_ip_ranges_csv_export": "4"
  }
}
```

Platform Features: Inventory

8.44.12 `update_inventory_domains`

Description

Updates fields on one or more inventory domains. Returns the updated `InventoryDomainDetail` records.

Response

Returns `[InventoryDomainDetail!]!`

Arguments

`input` - `[UpdateInventoryDomainBatchInput!]!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateInventoryDomains(
  $input: [UpdateInventoryDomainBatchInput!]!
) {
  update_inventory_domains(input: $input) {
    uuid
    domain_name
    authorization_status
    ownership_status
    ip_assignment_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.13 `delete_inventory_domain`

Description

Deletes a single inventory domain by UUID. Returns the count and details of deleted domains in an `InventoryDomainDeleteOutput`.

Response

Returns an `InventoryDomainDeleteOutput!`

Arguments

`uuid` - `ID!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryDomain(
  $uuid: ID!
) {
  delete_inventory_domain(uuid: $uuid) {
    deleted_count
    deleted_inventory_domains
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.14 `delete_inventory_domains`

Description

Deletes one or more inventory domains by UUID. Returns the count and details of deleted domains in an `InventoryDomainDeleteOutput`.

Response

Returns an `InventoryDomainDeleteOutput!`

Arguments

`uuids` - `[ID!]!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryDomains(
  $uuids: [ID!]!
) {
  delete_inventory_domains(uuids: $uuids) {
    deleted_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.15 `authorize_all_owned_inventory_domains`

Description

Authorizes all inventory domains with `ownership_status = 'Owned'` for the client account. When `page_input` is provided, only domains matching its `filter_by_inputs` and `text_search` are authorized — the FE can pair this with `inventory_domains_count` to preview the affected count. `page_num`, `page_size`, and `sort_inputs` on `page_input` are ignored. The hard rules `ownership_status = 'Owned'` and `is_authorizable = true` are always enforced and cannot be overridden by filters. Sets `authorization_status` to `Authorized`. Returns the count of updated domains.

Response

Returns a `BulkUpdateResult!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
mutation AuthorizeAllOwnedInventoryDomains(
  $page_input: PageInput
) {
  authorize_all_owned_inventory_domains(
    page_input: $page_input
  ) {
    updated_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.16 `update_inventory_ips`

Description

Updates fields on one or more inventory IPs. Returns the updated `InventoryIP` records.

Response

Returns `[InventoryIP!]!`

Arguments

`input` - `[UpdateInventoryIpInput!]!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateInventoryIps(
  $input: [UpdateInventoryIpInput!]!
) {
  update_inventory_ips(input: $input) {
    uuid
    ip
    authorization_status
    ownership_status
    assignment_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.17 `delete_inventory_ips`

Description

Deletes one or more inventory IPs by UUID. Returns the count of deleted IPs in an `InventoryIpDeleteOutput`.

Response

Returns an `InventoryIpDeleteOutput!`

Arguments

`uuids` - `[ID!]!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryIps(
  $uuids: [ID!]!
) {
  delete_inventory_ips(uuids: $uuids) {
    deleted_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.18 `authorize_all_owned_inventory_ips`

Description

Authorizes all inventory IPs with `ownership_status = 'Owned'` for the client account. When `page_input` is provided, only IPs matching its `filter_by_inputs` and `text_search` are authorized — the FE can pair this with `inventory_ips_count` to preview the affected count. `page_num`, `page_size`, and `sort_inputs` on `page_input` are ignored. The hard rules `ownership_status = 'Owned'` and `is_authorizable = true` are always enforced and cannot be overridden by filters. Sets `authorization_status` to `Authorized`. Returns the count of updated IPs.

Response

Returns a `BulkUpdateResult!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
mutation AuthorizeAllOwnedInventoryIps(
  $page_input: PageInput
) {
  authorize_all_owned_inventory_ips(
    page_input: $page_input
  ) {
    updated_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.19 `assign_assets_to_inventory_op_scope`

Description

Assigns existing inventory domains and/or CIDRs to an InventoryOpScope. Returns the updated `InventoryOpScope` with refreshed asset counts.

Response

Returns an `InventoryOpScope!`

Arguments

`inventory_op_scope_uuid` - `ID!`

`domain_uuids` - `[ID!]`

`cidr_uuids` - `[ID!]`

`ip_uuids` - `[ID!]`

IP UUIDs to assign. Each IP is wrapped in a /32 CIDR and the CIDR is linked to the scope.

Request Response Python

cURL h3-cli

```
mutation AssignAssetsToInventoryOpScope(
  $inventory_op_scope_uuid: ID!
  $domain_uuids: [ID!]
  $cidr_uuids: [ID!]
) {
  assign_assets_to_inventory_op_scope(
    inventory_op_scope_uuid: $inventory_op_scope_uuid
    domain_uuids: $domain_uuids
    cidr_uuids: $cidr_uuids
  ) {
    uuid
    name
    asset_type
    domain_count
    cidr_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.20 `unassign_assets_from_inventory_op_scope`

Description

Unassigns inventory domains and/or CIDRs from an InventoryOpScope. For IPs, the associated /32 CIDR is unassigned from the scope. Returns the updated `InventoryOpScope` with refreshed asset counts.

Response

Returns an `InventoryOpScope!`

Arguments

`inventory_op_scope_uuid` - `ID!`

`domain_uuids` - `[ID!]`

`cidr_uuids` - `[ID!]`

`ip_uuids` - `[ID!]`

IP UUIDs to unassign. The associated /32 CIDR is unlinked from the scope.

Request Response Python

cURL h3-cli

```
mutation UnassignAssetsFromInventoryOpScope(
  $inventory_op_scope_uuid: ID!
  $domain_uuids: [ID!]
  $cidr_uuids: [ID!]
) {
  unassign_assets_from_inventory_op_scope(
    inventory_op_scope_uuid: $inventory_op_scope_uuid
    domain_uuids: $domain_uuids
    cidr_uuids: $cidr_uuids
  ) {
    uuid
    name
    asset_type
    domain_count
    cidr_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.21 `create_inventory_op_scope`

Description

Creates a new inventory op scope for organizing external attack surface assets. Returns the created `InventoryOpScope`.

Response

Returns an `InventoryOpScope!`

Arguments

`input` - `CreateInventoryOpScopeInput!`

Request | Response | Python
cURL | h3-cli

```
mutation CreateInventoryOpScope(
  $input: CreateInventoryOpScopeInput!
) {
  create_inventory_op_scope(input: $input) {
    uuid
    name
    asset_type
    domain_count
    cidr_count
    created_at
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.22 `update_inventory_op_scope`

Description

Updates an inventory op scope — name, schedule config, and/or EAD schedule. Returns the updated `InventoryOpScope`.

Response

Returns an `InventoryOpScope!`

Arguments

`input` - `UpdateInventoryOpScopeInput!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateInventoryOpScope(
  $input: UpdateInventoryOpScopeInput!
) {
  update_inventory_op_scope(input: $input) {
    uuid
    name
    asset_type
    last_updated
    ead_schedule {
      uuid
      schedule_name
      cron_expression
      max_triggers
      is_disabled
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.23 `update_inventory_op_scopes`

Use update_inventory_op_scope (singular) instead.

Description

Updates one or more inventory op scopes. Each item in the input list must include the scope UUID and the fields to update. Returns the updated `InventoryOpScope` records.

Response

Returns `[InventoryOpScope!]!`

Arguments

`input` - `[UpdateInventoryOpScopeInput!]!`

Request | Response | Python
cURL | h3-cli

```
mutation UpdateInventoryOpScopes(
  $input: [UpdateInventoryOpScopeInput!]!
) {
  update_inventory_op_scopes(input: $input) {
    uuid
    name
    asset_type
    last_updated
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.24 `delete_inventory_op_scopes`

Description

Deletes one or more inventory op scopes by UUID. Returns the count and details of deleted scopes in an `InventoryOpScopeDeleteOutput`.

Response

Returns an `InventoryOpScopeDeleteOutput!`

Arguments

`uuids` - `[ID!]!`

Request | Response | Python
cURL | h3-cli

```
mutation DeleteInventoryOpScopes(
  $uuids: [ID!]!
) {
  delete_inventory_op_scopes(uuids: $uuids) {
    deleted_count
    deleted_inventory_op_scopes
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.25 BulkInventoryUploadStatus

Description

Lifecycle status of a bulk inventory upload.

Values

`PENDING`
The file has been uploaded but processing has not yet been initiated.
`PROCESSING`
The file is currently being processed.
`COMPLETED`
Processing completed successfully.
`FAILED`
Processing encountered an error.

Example

```
"PENDING"
```

Platform Features: Inventory

8.44.26 BulkUpdateResult

Description

Output returned from bulk update operations.

Fields

`updated_count` - `Int!`
Number of records updated.

Request Response

```
fragment BulkUpdateResultFragment on BulkUpdateResult {
  updated_count
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.27 CreateInventoryCidrInput

Description

Fields for creating a new `InventoryCidr`.

Fields

`cidr` - `String!`
CIDR notation (e.g. 203.0.113.0/24).

Request

```
mutation CreateInventoryCidrExample(
  $input: CreateInventoryCidrInput = {
    cidr: "192.168.1.0/24"
    ownership_status: "owned"
    asset_type: "external"
  }
) {
  create_inventory_cidr(input: $input) {
    uuid
    cidr
    ownership_status
    asset_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.28 CreateInventoryOpScopeInput

Description

Input for creating a new `InventoryOpScope`.

Fields

`name` - `String!`
Display name for the new scope. Must be unique within the client account.
`schedule_op_form` - `ScheduleOpFormInput`
Schedule op form config for the scope's auto-created External Asset Discovery OpTemplate. Domains and CIDRs are extracted from the form and linked to the scope.
`ead_cron_expression` - `String`
Cron expression for automatically scheduling EADs against this scope. When provided, a ScheduledAction is created and linked to the scope.
`ead_max_triggers` - `Int`
Maximum number of times the EAD schedule should trigger. When omitted, the schedule runs indefinitely.

Request

```
mutation CreateInventoryOpScopeExample(
  $input: CreateInventoryOpScopeInput = {
    name: "External Attack Surface"
    asset_type: "external"
  }
) {
  create_inventory_op_scope(input: $input) {
    uuid
    name
    asset_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.29 InitiateBulkInventoryUploadInput

Description

Input for the `initiate_bulk_inventory_upload` mutation.

Fields

`file_name` - `String`
Optional display name for the upload.
`file_size_in_bytes` - `Float`
File size in bytes (for display purposes).

Request

```
mutation InitiateBulkInventoryUploadExample(
  $input: InitiateBulkInventoryUploadInput = {
    file_name: "inventory_assets.csv"
    file_type: "text/csv"
  }
) {
  initiate_bulk_inventory_upload(input: $input) {
    upload_uuid
    presigned_url
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.30 InitiateBulkInventoryUploadOutput

Description

Output from the `initiate_bulk_inventory_upload` mutation.

Fields

`upload_uuid` - `String!`
Unique identifier assigned to the upload record.
`upload_url` - `String!`
Presigned URL for uploading the file. Use an HTTP PUT request.

Request Response

```
fragment InitiateBulkInventoryUploadOutputFragment on InitiateBulkInventoryUploadOutput {
  upload_uuid
  presigned_url
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.31 InventoryAssetsCsvInput

Description

Input for generating an inventory assets CSV export.

Supports three export modes:

- **Filter-based ("Download All"):** Pass `page_input` with active filters to export all matching rows.
- **Selection-based ("Download Selected"):** Pass `uuids` to export specific assets.
- **Scope-level:** Pass `inventory_op_scope_uuid` to export all assets for a scope.

Fields

`export_type` - `InventoryExportType!`
Type of assets to export: DOMAINS, IPS, or ALL.
`inventory_op_scope_uuid` - `ID`
Optional scope UUID to restrict export to a single scope's assets.
`page_input` - `PageInput`
Filter-based export: export all rows matching these filters.
`uuids` - `[ID!]`
Selection-based export: export specific assets by UUID.

Request

```
query InventoryAssetsCsvInputExample(
  $input: InventoryAssetsCsvInput = {
    export_type: DOMAINS
    inventory_op_scope_uuid: "aaaa1111-bbbb-2222-cccc-dddd3333eeee"
  }
) {
  inventory_assets_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.32 InventoryCidr

Description

An InventoryCidr represents a CIDR block in an external inventory.

Fields

`uuid` - `ID!`
Unique identifier of the CIDR record.
`cidr` - `String`
CIDR notation (e.g. 10.0.0.0/24).
`asn` - `String`
Autonomous System Number associated with this CIDR.
`isp` - `String`
Internet Service Provider associated with this CIDR. Not in designs. Always returns null.
`max_ips_in_range` - `Int`
Maximum number of IPs in this CIDR range (e.g. /24 = 256, /32 = 1).
`discovered_count` - `Int`
Number of assets discovered within this CIDR.
`tested_count` - `Int`
Number of assets tested within this CIDR. Not in designs. Always returns null.
`organizations` - `[String!]!`
Organizations associated with this CIDR. Not in designs. Always returns null.
`last_seen` - `String`
Timestamp when this CIDR was last seen.
`first_seen` - `String`
Timestamp when this CIDR was first seen.
`scopes` - `[InventoryOpScope]`
Op scopes this CIDR belongs to.

Request Response

```
fragment InventoryCidrFragment on InventoryCidr {
  uuid
  cidr
  asn
  isp
  ownership_status
  asset_type
  discovered_count
  tested_count
  first_seen
  last_seen
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.33 InventoryCidrDeleteOutput

Description

Output returned when deleting inventory CIDRs.

Fields

`deleted_count` - `Int!`
Number of CIDRs deleted.
`deleted_inventory_cidrs` - `[InventoryCidr!]!`
The CIDRs that were deleted.

Request Response

```
fragment InventoryCidrDeleteOutputFragment on InventoryCidrDeleteOutput {
  deleted_count
  deleted_inventory_cidrs
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.34 InventoryCidrsPage

Description

A paginated list of InventoryCidr records.

Fields

`inventory_cidrs` - `[InventoryCidr!]!`
The list of inventory CIDRs on the current page.
`page_info` - `PageInfo!`
Pagination metadata.

Request Response

```
fragment InventoryCidrsPageFragment on InventoryCidrsPage {
  inventory_cidrs {
    uuid
    cidr
    asn
  }
  page_info {
    total_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.35 InventoryDomain

Description

An InventoryDomain represents a domain record in an external inventory scope.

Fields

`uuid` - `ID!`
Unique identifier of the domain record.
`domain_name` - `String`
The domain name.
`is_authorizable` - `Boolean!`
Whether this domain can be authorized (i.e. passes the pentest eligibility rules). Derived from pentestable_rules.is_allowed; treated as false when rules haven't been computed. This matches the gate used by authorize_all_owned_inventory_domains.

Request Response

```
fragment InventoryDomainFragment on InventoryDomain {
  uuid
  domain
  domain_type
  authorization_status
  ownership_status
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.36 InventoryDomainDeleteOutput

Description

Output returned when deleting inventory domains.

Fields

`deleted_count` - `Int!`
Number of domains deleted.
`deleted_inventory_domains` - `[InventoryDomainDetail!]!`
The domains that were deleted.

Request Response

```
fragment InventoryDomainDeleteOutputFragment on InventoryDomainDeleteOutput {
  deleted_count
  deleted_inventory_domains
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.37 InventoryDomainDetail

Description

An InventoryDomainDetail represents a domain with its associated IPs and metadata in an external inventory. This is the full representation used in domain list views and mutations.

Fields

`uuid` - `ID!`
Unique identifier of the domain record.
`domain_name` - `String`
The domain name.
`ips` - `[InventoryIP]`
Inventory IP records linked to this domain.
`asset_types` - `[String]`
Asset type fingerprints detected on this domain.
`pentestable_rules` - `PentestableRules`
Pentestable rules governing authorization for this domain.
`is_authorizable` - `Boolean!`
Whether this domain can be authorized (i.e. passes the pentest eligibility rules). Derived from pentestable_rules.is_allowed; treated as false when rules haven't been computed. This matches the gate used by authorize_all_owned_inventory_domains.
`last_seen` - `String`
Timestamp when this domain was last seen.
`first_seen` - `String`
Timestamp when this domain was first seen.
`scopes` - `[InventoryOpScope]`
Op scopes this domain belongs to.
`endpoint_isps` - `[String]`
Internet Service Providers associated with this domain's IPs, from the most recent op.
`endpoint_organizations` - `[String]`
Organizations associated with this domain's IPs, from the most recent op.
`endpoint_asns` - `[String]`
Autonomous System Numbers associated with this domain's IPs, from the most recent op.
`web_page_proofs_count` - `Int`
Number of web page proof screenshots for this domain, from the most recent op.
`is_reachable` - `Boolean`
Whether this domain was reachable in its most recent ExternalAssetDiscovery record.
`last_external_domain_uuid` - `String`
UUID of the most recent ExternalDomain record for this domain.
`last_external_domain` - `ExternalDomain`
The most recent ExternalDomain record for this domain, providing access to per-op data such as web page proofs.

Request Response

```
fragment InventoryDomainDetailFragment on InventoryDomainDetail {
  uuid
  authorization_status
  ownership_status
  domain_name
  ip_assignment_type
  ips
  asset_types
  last_seen
  first_seen
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.38 InventoryDomainsPage

Description

A paginated list of InventoryDomain records.

Fields

`inventory_domains` - `[InventoryDomain!]!`
The list of inventory domains on the current page.
`page_info` - `PageInfo!`
Pagination metadata.

Request Response

```
fragment InventoryDomainsPageFragment on InventoryDomainsPage {
  inventory_domains {
    uuid
    domain
    domain_type
  }
  page_info {
    total_count
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.39 InventoryExportType

Description

Export type for inventory asset CSV exports.

Values

`DOMAINS`
Export domains only.
`IPS`
Export IPs only.
`ALL`
Export all assets (domains and IPs).

Example

```
"DOMAINS"
```

Platform Features: Inventory

8.44.40 InventoryIP

Description

An InventoryIP represents an IP address in an external inventory.

Fields

`uuid` - `ID!`
Unique identifier of the IP record.
`ip` - `String`
The IP address string.
`is_dynamic_ip` - `Boolean`
Whether this is a dynamic IP address.
`is_reachable` - `Boolean`
Whether this IP is currently reachable.
`cidrs` - `[InventoryCidr!]!`
CIDR blocks that contain this IP.
`domains` - `[InventoryDomain!]!`
Domains that resolve to this IP.
`pentestable_rules` - `PentestableRules`
Pentestable rules governing authorization for this IP.
`is_authorizable` - `Boolean!`
Whether this IP can be authorized (i.e. passes the pentest eligibility rules). Derived from pentestable_rules.is_allowed; treated as false when rules haven't been computed. This matches the gate used by authorize_all_owned_inventory_ips.
`last_seen` - `String`
Timestamp when this IP was last seen.
`first_seen` - `String`
Timestamp when this IP was first seen.
`scopes` - `[InventoryOpScope]`
Op scopes this IP belongs to (derived via CIDR membership).
`isp` - `String`
Internet Service Provider for this IP, from the most recent op.
`organization` - `String`
Organization associated with this IP, from the most recent op.
`asn` - `String`
Autonomous System Number for this IP, from the most recent op.
`asset_types` - `[String]`
Asset type fingerprints detected on this IP, from the most recent op.
`web_page_proofs_count` - `Int`
Number of web page proof screenshots for this IP, from the most recent op.
`last_host_tab_uuid` - `String`
UUID of the most recent HostTab record for this IP.

Request Response

```
fragment InventoryIPFragment on InventoryIP {
  uuid
  ip
  authorization_status
  ownership_status
  is_dynamic_ip
  is_reachable
  assignment_type
  last_seen
  first_seen
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.41 InventoryIpDeleteOutput

Description

Output returned when deleting inventory IPs.

Fields

`deleted_count` - `Int!`
Number of IPs deleted.

Request Response

```
fragment InventoryIpDeleteOutputFragment on InventoryIpDeleteOutput {
  deleted_count
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.42 InventoryIpRangesCsvInput

Description

Input for generating an inventory IP ranges CSV export.

Supports three export modes:

- **Filter-based ("Download All"):** Pass `page_input` with active filters to export all matching rows.
- **Selection-based ("Download Selected"):** Pass `uuids` to export specific ranges.
- **Scope-level:** Pass `inventory_op_scope_uuid` to export all IP ranges linked to a scope.

Fields

`inventory_op_scope_uuid` - `ID`
Optional scope UUID to restrict export to a single scope's IP ranges.
`page_input` - `PageInput`
Filter-based export: export all rows matching these filters.
`uuids` - `[ID!]`
Selection-based export: export specific IP ranges by UUID.

Request

```
# Example 1: Filter-based ("Download All") export
query DownloadAllIpRangesCsv(
  $input: InventoryIpRangesCsvInput = {
    page_input: {
      page_size: 100
      filter_by_inputs: [
        { field_name: "ownership_status", values: ["Owned"] }
      ]
    }
  }
) {
  inventory_ip_ranges_csv_presigned_url(input: $input)
}

# Example 2: Selection-based ("Download Selected") export
query DownloadSelectedIpRangesCsv(
  $input: InventoryIpRangesCsvInput = {
    uuids: [
      "aaaa1111-bbbb-2222-cccc-dddd3333eeee",
      "bbbb2222-cccc-3333-dddd-eeee4444ffff"
    ]
  }
) {
  inventory_ip_ranges_csv_presigned_url(input: $input)
}

# Example 3: Scope-level export
query DownloadScopeIpRangesCsv(
  $input: InventoryIpRangesCsvInput = {
    inventory_op_scope_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
  }
) {
  inventory_ip_ranges_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.43 InventoryOpScope

Description

An InventoryOpScope defines a named scope for organizing inventory assets (domains and CIDRs) for external attack surface monitoring.

Fields

`uuid` - `ID!`
Unique identifier of the op scope.
`name` - `String!`
Display name of the scope.
`authorized_assets` - `Int`
Number of authorized assets in this scope. Use `authorized_domain_count` and `authorized_ip_count` instead.
`unauthorized_assets` - `Int`
Number of unauthorized assets in this scope. No longer tracked. Use total counts minus authorized counts.
`reachable_assets` - `Int`
Number of reachable assets in this scope. No longer tracked.
`domain_count` - `Int`
Number of domains in this scope.
`cidr_count` - `Int`
Number of CIDR blocks in this scope.
`ip_count` - `Int`
Number of discovered IPs associated to the scope via the CIDR chain.
`authorized_domain_count` - `Int`
Number of authorized domains in this scope.
`authorized_ip_count` - `Int`
Number of authorized IPs associated to the scope via the CIDR chain.
`authorized_asset_count` - `Int`
Total number of authorized assets (domains + IPs) in this scope.
`last_updated` - `String`
Timestamp when this scope was last updated. Updated on scope modification, asset assign, and asset unassign.
`created_at` - `String`
Timestamp when this scope was created.
`user_account_uuid` - `String!`
UUID of the user who created this scope.
`created_by_user_account` - `UserAccount!`
The user who created this scope.
`bulk_upload_status` - `BulkInventoryUploadStatus`
Status of the most recent bulk upload targeting this scope. Null if no bulk uploads have been performed.
`op_template_uuid` - `String`
UUID of the linked OpTemplate, if any.
`op_template` - `OpTemplate`
The linked OpTemplate containing the scope's EAD configuration.
`ead_schedule_uuid` - `String`
UUID of the EAD ScheduledAction linked to this scope, if any. Resolved at read time via the scope → op_template → scheduled_action join.
`ead_schedule` - `ScheduledAction`
The EAD schedule (run-pentest ScheduledAction) linked to this scope, if any.
`last_pentest_scheduled_at` - `Datetime`
scheduled_at of the most recent ExternalAttack op that targeted this scope. Null if no pentests have run.
`last_pentest_op_id` - `String`
op_id of the most recent ExternalAttack op that targeted this scope.
`last_pentest_op` - `OpTab`

The most recent ExternalAttack op that targeted this scope.

`last_ead_scheduled_at` - `Datetime`

scheduled_at of the most recent ExternalAssetDiscovery op that targeted this scope. Null if no EAD has run.

`last_ead_op_id` - `String`

op_id of the most recent ExternalAssetDiscovery op that targeted this scope.

`last_ead_op` - `OpTab`

The most recent ExternalAssetDiscovery op that targeted this scope.

`pentest_templates` - `[OpTemplate!]!`

Pentest OpTemplates that target this scope via schedule_op_form.inventory_op_scope_uuids. Non-empty means deleting this scope will also delete these templates and their scheduled_actions.

`pentest_template_count` - `Int!`

Count of pentest OpTemplates targeting this scope. Supports filtering and sorting on the Scopes tab.

`scope_domains_page` - `InventoryDomainsPage!`

Paginated domains linked to this scope.

Arguments

`page_input` - `PageInput`

`scope_domains_count` - `Int!`

Count of domains linked to this scope, respecting any filters in page_input.

Arguments

`page_input` - `PageInput`

`scope_cidrs_page` - `InventoryCidrsPage!`

Paginated CIDRs linked to this scope.

Arguments

`page_input` - `PageInput`

`scope_cidrs_count` - `Int!`

Count of CIDRs linked to this scope, respecting any filters in page_input.

Arguments

`page_input` - `PageInput`

`scope_discoveries_count` - `Int!`

Count of completed/canceled EAD discoveries run against this scope.

Arguments

`page_input` - `PageInput`

`scope_discoveries_page` - `ScopeDiscoveriesPage!`

Paginated list of completed/canceled EAD discoveries run against this scope.

Arguments

`page_input` - `PageInput`

Request Response

```
fragment InventoryOpScopeFragment on InventoryOpScope {
  uuid
  name
  asset_type
  domain_count
  cidr_count
  ip_count
  authorized_domain_count
  authorized_ip_count
  last_updated
  created_at
  user_account_uuid
  created_by_user_account {
    uuid
    email
    name
  }
  bulk_upload_status
  op_template_uuid
  scope_discoveries_count
```

```
  scope_discoveries_page {
    discoveries {
      op_id
      op_name
      op_state
      nodezero_ip
      scheduled_at
      completed_timestamp
    }
    page_info {
      page_num
      page_size
    }
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.44 InventoryOpScopeDeleteOutput

Description

Output returned when deleting inventory op scopes.

Fields

`deleted_count` - `Int!`
Number of scopes deleted.
`deleted_inventory_op_scopes` - `[InventoryOpScope!]!`
The scopes that were deleted.

Request Response

```
fragment InventoryOpScopeDeleteOutputFragment on InventoryOpScopeDeleteOutput {
  deleted_count
  deleted_inventory_op_scopes
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.45 InventoryScopesCsvInput

Description

Input for generating an inventory scopes CSV export.

Supports two export modes:

- **Filter-based ("Download All"):** Pass `page_input` with active filters to export all matching rows.
- **Selection-based ("Download Selected"):** Pass `uuids` to export specific scopes.

Fields

`page_input` - `PageInput`
Filter-based export: export all rows matching these filters.
`uuids` - `[ID!]`
Selection-based export: export specific scopes by UUID.

Request

```
# Example 1: Filter-based ("Download All") export
query DownloadAllScopesCsv(
  $input: InventoryScopesCsvInput = {
    page_input: {
      page_size: 100
      filter_by_inputs: [
        { field_name: "ownership_status", values: ["Owned"] }
      ]
    }
  }
) {
  inventory_scopes_csv_presigned_url(input: $input)
}

# Example 2: Selection-based ("Download Selected") export
query DownloadSelectedScopesCsv(
  $input: InventoryScopesCsvInput = {
    uuids: [
      "aaaa1111-bbbb-2222-cccc-dddd3333eeee",
      "bbbb2222-cccc-3333-dddd-eeee4444ffff"
    ]
  }
) {
  inventory_scopes_csv_presigned_url(input: $input)
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.46 ProcessBulkInventoryUploadInput

Description

Input for the `process_bulk_inventory_upload` mutation.

Fields

`upload_uuid` - `String!`
UUID of the bulk inventory upload to process.
`scope_uuid` - `String`
Optional scope UUID to associate uploaded assets with.

Request

```
mutation ProcessBulkInventoryUploadExample(
  $input: ProcessBulkInventoryUploadInput = {
    upload_uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    inventory_op_scope_uuid: "5678efgh-5678-efgh-5678-efgh5678efgh"
  }
) {
  process_bulk_inventory_upload(input: $input) {
    upload_uuid
    status
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.47 ProcessBulkInventoryUploadOutput

Description

Output from the `process_bulk_inventory_upload` mutation.

Fields

`success` - `Boolean!`
True if the processing request was successfully submitted.

Request Response

```
fragment ProcessBulkInventoryUploadOutputFragment on ProcessBulkInventoryUploadOutput {
  upload_uuid
  status
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.48 ScopeDiscoveriesPage

Description

Paginated list of EAD discoveries (completed/canceled ops) run against a scope.

Fields

`discoveries` - `[OpTab!]!`
The list of discovery ops on the current page.
`page_info` - `PageInfo`
Pagination metadata.

Request Response

```
fragment ScopeDiscoveriesPageFragment on ScopeDiscoveriesPage {
  discoveries {
    op_id
    op_name
    op_state
    nodezero_ip
    scheduled_at
    completed_timestamp
  }
  page_info {
    page_num
    page_size
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.49 UpdateInventoryCidrInput

Description

Fields that can be updated on an `InventoryCidr`.

Request

```
mutation UpdateInventoryCidrExample(
  $uuid: ID!
  $input: UpdateInventoryCidrInput = {
    ownership_status: "owned"
  }
) {
  update_inventory_cidr(uuid: $uuid, input: $input) {
    uuid
    cidr
    ownership_status
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.50 UpdateInventoryDomainBatchInput

Description

Input for batch-updating an inventory domain. Includes the UUID of the domain to update along with the fields to change.

Fields

`uuid` - `ID!`
UUID of the domain to update.

Request

```
mutation UpdateInventoryDomainsExample(
  $input: [UpdateInventoryDomainBatchInput!]! = [{
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    ownership_status: "Owned"
    ip_assignment_type: "Static"
    authorization_status: "Authorized"
  }]
) {
  update_inventory_domains(input: $input) {
    uuid
    domain_name
    authorization_status
    ownership_status
    ip_assignment_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.51 UpdateInventoryIpInput

Description

Fields that can be updated on an `InventoryIP`.

Fields

`uuid` - `ID!`
UUID of the IP to update.

Request

```
mutation UpdateInventoryIpsExample(
  $input: [UpdateInventoryIpInput!]! = [{
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    assignment_type: "static"
    ownership_status: "owned"
    authorization_status: "authorized"
  }]
) {
  update_inventory_ips(input: $input) {
    uuid
    ip
    authorization_status
    ownership_status
    assignment_type
  }
}
```

Examples may not include all available fields for a type.

Platform Features: Inventory

8.44.52 UpdateInventoryOpScopeInput

Description

Input for updating an existing `InventoryOpScope`.

Fields

`uuid` - `ID!`
UUID of the scope to update.
`name` - `String`
New display name for the scope.
`schedule_op_form` - `ScheduleOpFormInput`
Updated schedule op form config. Updates the scope's External Asset Discovery OpTemplate. Does NOT re-sync domains/CIDRs from the form. Use assign/unassign operations to manage scope assets.
`ead_cron_expression` - `String`
Cron expression for the scope's EAD schedule. When provided, creates the schedule if missing or updates the cron expression on the existing schedule.
`ead_max_triggers` - `Int`
Maximum number of times the EAD schedule should trigger. Pass `-1` to clear (run indefinitely). Omit to leave unchanged.
`ead_schedule_enabled` - `Boolean`
Enable or disable the scope's EAD schedule. When false, the schedule remains but does not fire.

Request

```
mutation UpdateInventoryOpScopesExample(
  $input: [UpdateInventoryOpScopeInput!]! = [{
    uuid: "1234abcd-1234-abcd-1234-abcd1234abcd"
    name: "Updated Scope Name"
  }]
) {
  update_inventory_op_scopes(input: $input) {
    uuid
    name
    asset_type
  }
}
```

Examples may not include all available fields for a type.

8.45 Admin

Administrative APIs for your Horizon3.ai account. View your account settings and the client accounts you have access to. For Managed Service Providers (MSPs), view aggregated data across your subclient accounts.

8.46 Admin: Account Management

View account settings and organizational configuration.

8.46.1 `session_user_account`

Description

Returns the UserAccount for the currently authenticated user, including their profile information and role within the current ClientAccount.

Response

Returns a `UserAccount`

Request | Response | Python
cURL | h3-cli

```
query SessionUserAccount {
  session_user_account {
    ...UserAccountFragment
  }
}

fragment UserAccountFragment on UserAccount {
  uuid
  email
  name
  user_role_id
  sign_in_type
  last_signed_in_at
  has_tripwires_access
  has_rapid_response_access
  has_unread_notifications_by_type
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.2 `client_accounts_count`

Description

Returns the number of ClientAccount records accessible by the currently authenticated user. Supports filtering via `page_input`.

Response

Returns an `Int!`

Arguments

`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query ClientAccountsCount($page_input: PageInput) {
  client_accounts_count(page_input: $page_input)
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.3 `client_accounts_page`

Description

Returns a paginated list of ClientAccount records that the currently authenticated user has access to. Use `exclude_user_roles` to omit accounts where the user holds a specific role (e.g., exclude accounts where the user is `READONLY`).

Response

Returns a `ClientAccountsPage!`

Arguments

`page_input` - `PageInput`

`exclude_user_roles` - `[AuthzRole]`

Request | Response | Python

cURL | h3-cli

```
query ClientAccountsPage($page_input: PageInput) {
  client_accounts_page(page_input: $page_input) {
    client_accounts {
      ...ClientAccountFragment
    }
  }
}

fragment ClientAccountFragment on ClientAccount {
  uuid
  company_name
  company_short_name
  session_user_role_id
  company_logo_url
  tripwires_enabled
  assets_count
  internal_assets_count
  external_assets_count
  row_created_at
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.4 ClientAccount

Description

Represents an organization (tenant) on the platform. Accounts may be organized in parent-child hierarchies, where a parent account manages multiple subclient accounts.

Fields

`uuid` - `String!`
Unique identifier of the client account.
`parent_uuid` - `String`
Unique identifier of the parent client account, if this is a subclient account. Returns `null` for top-level accounts.
`child_client_accounts` - `[ClientAccount!]!`
List of subclient accounts managed by this parent account. Requires multi-tenancy to be enabled; returns an empty list otherwise. Only subclient accounts the current user has access to are included.
`child_client_accounts_page` - `ClientAccountsPage!`
Paginated list of subclient accounts managed by this parent account. Requires multi-tenancy to be enabled. Returns a ClientAccountsPage.

Arguments

`page_input` - `PageInput`
`child_client_accounts_count` - `Int!`
Number of subclient accounts managed by this parent account. Requires multi-tenancy to be enabled; returns `0` otherwise.

Arguments

`page_input` - `PageInput`
`company_name` - `String!`
Full legal name of the company associated with this client account.
`company_short_name` - `String!`
Abbreviated name of the company, typically used in reports and dashboard labels.
`company_logo_url` - `String`
Pre-signed URL for the company's primary logo image. The URL expires after one hour.
`secondary_company_logo_url` - `String`
Pre-signed URL for the company's secondary logo image. The URL expires after one hour.
`company_colors` - `[BrandColor!]`
The white-label BrandColor scheme for this account, containing primary and secondary brand colors used in co-branded reports.
`white_label_reports_enabled` - `Boolean`
Whether co-branded (white-label) reports are enabled for this client account.
`white_label_reports_cascade` - `Boolean`
Whether the co-branded report settings from this account are cascaded to its subclient accounts.
`row_created_at` - `Datetime`
Timestamp when the client account was created, in ISO 8601 format (UTC).
`session_user_role_id` - `AuthzRole`
The AuthzRole of the currently authenticated user in this client account. Returns `null` if the user does not have access to this specific account.
`tripwires_enabled` - `Boolean!`
Whether NodeZero Tripwires is enabled for this client account. When enabled, the account can deploy and monitor tripwires to detect attacker activity.
`assets_count` - `Int`
Total number of tracked assets for this client account. Does not include assets from subclient accounts.
`internal_assets_count` - `Int`

Number of internal (on-premises / LAN) assets tracked for this client account. Does not include assets from subclient accounts.

`external_assets_count` - `Int`

Number of external (internet-facing) assets tracked for this client account. Does not include assets from subclient accounts.

Request Response

```
fragment ClientAccountFragment on ClientAccount {
  uuid
  parent_uuid
  company_name
  company_short_name
  company_logo_url
  secondary_company_logo_url
  company_colors {
    type
    color
  }
  white_label_reports_enabled
  white_label_reports_cascade
  session_user_role_id
  tripwires_enabled
  assets_count
  internal_assets_count
  external_assets_count
  row_created_at
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.5 UserAccount

Description

A user's identity and profile on the platform. Users are identified by their email address and may have access to multiple ClientAccounts with different AuthzRoles in each. A user may also have multiple SignInTypes configured (e.g., BASIC, GOOGLE, SSO).

Fields

`uuid` - `String`
Unique identifier of the user account.
`email` - `EmailAddress`
The user's email address, which serves as their unique login identifier across the platform.
`name` - `String`
Full name of the user.
`user_role_id` - `AuthzRole`
The AuthzRole assigned to this user within the calling user's ClientAccount.
`sign_in_type` - `SignInType`
The SignInType (authentication method) used by this specific user account, such as `BASIC`, `GOOGLE`, `MICROSOFT`, or `SSO`.
`last_signed_in_at` - `Datetime`
Timestamp of the most recent sign-in by this user within the calling user's ClientAccount, in ISO 8601 format (UTC).
`has_unread_notifications_by_type` - `[NotificationType]!`
Returns the list of notification types that have unread notifications for this user in their current client account. An empty list indicates all notifications have been read.
`has_tripwires_access` - `Boolean!`
Whether this user has access to the Tripwires product in their current ClientAccount. Users with Tripwires access can query tripwire data and receive notifications when tripwires detect activity.
`has_rapid_response_access` - `Boolean!`
Whether this user is authorized to receive Rapid Response notifications about evidence of breaking vulnerabilities in their current ClientAccount.

Request Response

```
fragment UserAccountFragment on UserAccount {
  uuid
  email
  name
  user_role_id
  sign_in_type
  last_signed_in_at
  has_tripwires_access
  has_rapid_response_access
  has_unread_notifications_by_type
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.6 APIKey

Description

An API key used for programmatic access to the GraphQL API. API keys are associated with a specific ClientAccount and UserAccount, and carry the permissions of the assigned role.

Fields

`uuid` - `String!`
Unique identifier for the API key.
`api_key` - `String`
The full API key value (base64-encoded). This field is only populated in the response when the API key is first created. It cannot be retrieved again after creation.
`h3_cli_install_command` - `String`

A shell command that installs the h3-cli and configures it to use this API key. Only populated when the API key is first created.

`starts_with` - `String!`
The leading characters of the API key, useful for identifying a specific key without exposing the full value.
`created_at` - `Datetime!`
Timestamp when the API key was created, in ISO 8601 format (UTC).
`last_accessed_at` - `Datetime`
Timestamp of the most recent API request made with this key, in ISO 8601 format (UTC). Returns `null` if the key has never been used.

Request Response

```
fragment APIKeyFragment on APIKey {
  uuid
  api_key
  h3_cli_install_command
  starts_with
  created_at
  last_accessed_at
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.7 AuthzRole

Description

The authorization role assigned to a user within a specific ClientAccount. A user's role determines their permissions and may differ across client accounts they have access to.

Values

`USER`
Standard user role with permissions to run pentests, view results, and manage configurations within the client account. Cannot manage other users or account settings.
`READONLY`
Read-only user role. Can view pentest results and account data but cannot initiate pentests, modify configurations, or manage users.
`ORG_ADMIN`
Organizational administrator role with full permissions within the client account, including user management, account configuration, and subclient management for MSP accounts.
`NODEZERO_RUNNER`
Role used by the h3-cli to configure and manage an automated Runner for NodeZero. Grants permissions to register runners and execute pentest deployments.
`NODEZERO_OPERATOR`
Role used by the NodeZero Kubernetes Operator to access the API when provisioning Runners or pentests within a Kubernetes cluster.
`PHISHER`
Role used for pentests that include phishing scenarios, enabling the submission of phished credentials to the platform.

Example

```
"USER"
```

Admin: Account Management

8.46.8 BrandColor

Description

A brand color in a ClientAccount's white-label color scheme. Used for co-branded reports and portal customization.

Fields

`type` - `BrandColorType!`
Whether this is the primary or secondary brand color.
`color` - `HexColor!`
The color value in hexadecimal notation.

Request Response

```
fragment BrandColorFragment on BrandColor {
  type
  color
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.9 BrandColorType

Description

Identifies a brand color as either the primary or secondary color in a ClientAccount's white-label color scheme.

Values

`primary`
The primary brand color, used as the main accent color in co-branded reports and portal elements.
`secondary`
The secondary brand color, used as a complementary color in co-branded reports and portal elements.

Example

```
"primary"
```

Admin: Account Management

8.46.10 ClientAccountsPage

Description

A paginated list of ClientAccount records. Returned by client_accounts_page.

Fields

`page_info` - `PageInfo`
Pagination metadata including total count and cursor information.
`client_accounts` - `[ClientAccount!]!`
The list of ClientAccount records for the current page.

Request Response

```
fragment ClientAccountsPageFragment on ClientAccountsPage {
  page_info {
    page_num
    page_size
  }
  client_accounts {
    uuid
    company_name
    company_short_name
    session_user_role_id
    row_created_at
  }
}
```

Examples may not include all available fields for a type.

Admin: Account Management

8.46.11 HexColor

Description

A string representing a color in hexadecimal notation (e.g., `#def` or `#bb032d`). Used for white-label BrandColor customization.

Example

```
"#bb032d"
```

Admin: Account Management

8.46.12 SignInType

Description

The authentication method used by a UserAccount to sign in to the platform. A single user (identified by email) may have multiple sign-in types available.

Values

`BASIC`
Standard authentication using a username (email) and password managed by the platform.
`GOOGLE`
Federated authentication via Google identity provider.
`LINKED_IN`
Federated authentication via LinkedIn identity provider.
`MICROSOFT`
Federated authentication via Microsoft identity provider.
`SSO`
Single Sign-On authentication through a customer-configured identity provider (e.g., Okta, Azure AD).

Example

```
"BASIC"
```

8.47 Admin: MSP

For Managed Service Providers (MSPs), view aggregated data across your subclient accounts.

8.47.1 `subclient_account_summaries`

Description

Returns a list of SubclientAccountSummary records containing all-time usage metrics (ops count, sign-in counts, user counts, etc.) for each subclient under the current user's parent account. Set `include_parent` to `true` to include the parent account's own metrics. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns `[SubclientAccountSummary!]!`

Arguments

`include_parent` - `Boolean`
`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query SubclientAccountSummaries($include_parent: Boolean, $page_input: PageInput) {
  subclient_account_summaries(include_parent: $include_parent, page_input: $page_input) {
    ...SubclientAccountSummaryFragment
  }
}

fragment SubclientAccountSummaryFragment on SubclientAccountSummary {
  client_account_uuid
  company_name
  enrolled_date
  ops_count
  successfully_launched_ops_count
  first_op_date
  last_op_date
  uniq_ips_count
  users_count
  user_sign_ins_count
  first_sign_in_date
```

```
  last_sign_in_date
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.2 `user_summaries_for_subclient_accounts`

Description

Returns a list of ClientAccountUserSummary records showing per-user sign-in activity for each subclient account. Set `include_parent` to `true` to include users from the parent account. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns `[ClientAccountUserSummary!]!`

Arguments

`include_parent` - `Boolean`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query UserSummariesForSubclientAccounts($include_parent: Boolean, $page_input: PageInput) {
  user_summaries_for_subclient_accounts(include_parent: $include_parent, page_input: $page_input) {
    ...ClientAccountUserSummaryFragment
  }
}

fragment ClientAccountUserSummaryFragment on ClientAccountUserSummary {
  company_name
  user_email
  company_linkedin_url
  sign_in_count
  first_sign_in_date
  last_sign_in_date
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.3 `impact_summaries_for_subclient_accounts`

Description

Returns a list of ClientAccountImpactSummary records aggregating attack path impacts by type for each subclient account. Each record shows how many pentests observed the impact and the total count of attack paths. Set `include_parent` to `true` to include the parent account. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns `[ClientAccountImpactSummary!]!`

Arguments

`include_parent` - `Boolean`
`page_input` - `PageInput`

Request | Response | Python
cURL | h3-cli

```
query ImpactSummariesForSubclientAccounts($include_parent: Boolean, $page_input: PageInput) {
  impact_summaries_for_subclient_accounts(include_parent: $include_parent, page_input: $page_input) {
    ...ClientAccountImpactSummaryFragment
  }
}

fragment ClientAccountImpactSummaryFragment on ClientAccountImpactSummary {
  client_account_uuid
  impact_type
  first_seen_date
  last_seen_date
  ops_count
  impact_paths_count
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.4 `weakness_summaries_for_subclient_accounts`

Description

Returns a list of ClientAccountWeaknessSummary records aggregating weakness occurrences by vulnerability ID for each subclient account. Set `include_parent` to `true` to include the parent account. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns `[ClientAccountWeaknessSummary!]!`

Arguments

`include_parent` - `Boolean`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query WeaknessSummariesForSubclientAccounts($include_parent: Boolean, $page_input: PageInput) {
  weakness_summaries_for_subclient_accounts(include_parent: $include_parent, page_input: $page_input) {
    ...ClientAccountWeaknessSummaryFragment
  }
}

fragment ClientAccountWeaknessSummaryFragment on ClientAccountWeaknessSummary {
  company_name
  vuln_id
  first_seen_date
  last_seen_date
  first_op_id
  last_op_id
  ops_count
  weaknesses_count
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.5 `cpe_strings_for_subclient_accounts`

Description

Returns a list of ClientAccountCPEString records containing Common Platform Enumeration (CPE) data for each endpoint discovered across subclient accounts. CPE strings provide standardized identification of hardware, operating systems, and software on network endpoints. Set `include_parent` to `true` to include the parent account. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns `[ClientAccountCPEString!]!`

Arguments

`include_parent` - `Boolean`

`page_input` - `PageInput`

Request | Response | Python

cURL | h3-cli

```
query CPEStringsForSubclientAccounts($include_parent: Boolean, $page_input: PageInput) {
  cpe_strings_for_subclient_accounts(include_parent: $include_parent, page_input: $page_input) {
    ...ClientAccountCPEStringFragment
  }
}

fragment ClientAccountCPEStringFragment on ClientAccountCPEString {
  client_account_company_name
  op_id
  endpoint_created_at_date
  ip
  cpe
  os_vendors
  os_products
  os_versions
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.6 `overall_summary_for_subclient_accounts`

Description

Returns an OverallSummarySubclientAccounts record with high-level aggregated metrics (total ops, weaknesses, attack paths, etc.) across all subclient accounts. Set `include_parent` to `true` to include the parent account's own metrics. Requires multi-tenancy to be enabled; returns an empty list otherwise.

Response

Returns an `OverallSummarySubclientAccounts`

Arguments

`include_parent` - `Boolean`

Request | Response | Python

cURL | h3-cli

```
query OverallSummaryForSubclientAccounts($include_parent: Boolean) {
  overall_summary_for_subclient_accounts(include_parent: $include_parent) {
    ...OverallSummarySubclientAccountsFragment
  }
}

fragment OverallSummarySubclientAccountsFragment on OverallSummarySubclientAccounts {
  enrolled_clients_count
  non_empty_ops_count
  ops_count
  hosts_or_unique_ips_scanned_count
  impact_paths_count
  weaknesses_count
  weaknesses_mitigated_count
  total_duration_of_ops_seconds
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.7 ClientAccountCPEString

Description

Common Platform Enumeration (CPE) data for an endpoint discovered in a subclient account's pentest. CPE is a standardized naming scheme for identifying hardware and software. Returned by cpe_strings_for_subclient_accounts.

Fields

`client_account_company_name` - `String`
Company name of the subclient account where this endpoint was discovered.
`op_id` - `String`
Identifier of the pentest (op) in which this endpoint was observed.
`endpoint_created_at_date` - `Date`
Date when this endpoint record was created.
`ip` - `String`
IP address of the endpoint.
`cpe` - `String`
Full CPE string identifying the endpoint's software or hardware stack.
`hw_vendors` - `[String]`
Hardware vendor names extracted from the CPE.
`hw_products` - `[String]`
Hardware product names extracted from the CPE.
`hw_devices` - `[String]`
Hardware device types extracted from the CPE.
`hw_versions` - `[String]`
Hardware version strings extracted from the CPE.
`hw_models` - `[String]`
Hardware model identifiers extracted from the CPE.
`hw_families` - `[String]`
Hardware family classifications extracted from the CPE.
`hw_cpes` - `[String]`
Full hardware CPE strings for this endpoint.
`os_vendors` - `[String]`
Operating system vendor names extracted from the CPE.
`os_architectures` - `[String]`
Operating system CPU architectures extracted from the CPE (e.g., x86_64, ARM).
`os_products` - `[String]`
Operating system product names extracted from the CPE.
`os_versions` - `[String]`
Operating system version strings extracted from the CPE.
`os_editions` - `[String]`
Operating system edition identifiers extracted from the CPE.
`os_builds` - `[String]`
Operating system build numbers extracted from the CPE.
`os_families` - `[String]`
Operating system family classifications extracted from the CPE (e.g., Linux, Windows).
`os_devices` - `[String]`
Operating system device types extracted from the CPE.
`os_languages` - `[String]`
Operating system language identifiers extracted from the CPE.
`os_cpes` - `[String]`
Full operating system CPE strings for this endpoint.

Request Response

```
fragment ClientAccountCPEStringFragment on ClientAccountCPEString {
  client_account_company_name
  op_id
  endpoint_created_at_date
  ip
  cpe
  os_vendors
  os_products
  os_versions
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.8 ClientAccountImpactSummary

Description

Aggregated summary of attack path impacts for a subclient account, grouped by impact type. Each record represents a unique combination of subclient and impact type. Returned by impact_summaries_for_subclient_accounts.

Fields

`client_account_uuid` - `String`
Unique identifier of the subclient ClientAccount.
`impact_type` - `String`
The category of impact observed (e.g., data exfiltration, lateral movement).
`first_seen_date` - `Date`
Date when this impact type was first observed in a pentest for the subclient.
`last_seen_date` - `Date`
Date when this impact type was most recently observed in a pentest for the subclient.
`ops_count` - `Int`
Number of pentests in which this impact type was observed.
`impact_paths_count` - `Int`
Total number of attack paths associated with this impact type across all pentests.

Request Response

```
fragment ClientAccountImpactSummaryFragment on ClientAccountImpactSummary {
  client_account_uuid
  impact_type
  first_seen_date
  last_seen_date
  ops_count
  impact_paths_count
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.9 ClientAccountUserSummary

Description

Per-user activity summary for a subclient account, showing sign-in frequency and recency. Each record represents a unique user within a specific subclient. Returned by user_summaries_for_subclient_accounts.

Fields

`company_name` - `String!`
Company name of the subclient account this user belongs to.
`user_email` - `String`
Email address of the user.
`company_linkedin_url` - `String`
LinkedIn URL of the subclient's company.
`sign_in_count` - `Int`
Total number of times this user has signed in to the subclient account, all-time.
`first_sign_in_date` - `Datetime`
Timestamp of this user's first sign-in to the subclient account, in ISO 8601 format (UTC).
`last_sign_in_date` - `Datetime`
Timestamp of this user's most recent sign-in to the subclient account, in ISO 8601 format (UTC).

Request Response

```
fragment ClientAccountUserSummaryFragment on ClientAccountUserSummary {
  company_name
  user_email
  company_linkedin_url
  sign_in_count
  first_sign_in_date
  last_sign_in_date
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.10 ClientAccountWeaknessSummary

Description

Aggregated summary of weakness occurrences for a subclient account, grouped by vulnerability identifier. Each record represents a unique combination of subclient and vulnerability. Returned by weakness_summaries_for_subclient_accounts.

Fields

`company_name` - `String`
Company name of the subclient account where this weakness was observed.
`vuln_id` - `String`
Vulnerability identifier (e.g., a CVE ID such as `CVE-2021-44228`) for this weakness.
`first_seen_date` - `Date`
Date when this weakness was first discovered in a pentest for the subclient.
`last_seen_date` - `Date`
Date when this weakness was most recently observed in a pentest for the subclient.
`first_op_id` - `String`
Identifier of the pentest where this weakness was first observed.
`last_op_id` - `String`
Identifier of the pentest where this weakness was most recently observed.
`ops_count` - `Int`
Number of distinct pentests in which this weakness was detected.
`weaknesses_count` - `Int`
Total number of times this weakness was observed across all pentests (may exceed `ops_count` if the weakness appears multiple times within a single pentest).

Request Response

```
fragment ClientAccountWeaknessSummaryFragment on ClientAccountWeaknessSummary {
  company_name
  vuln_id
  first_seen_date
  last_seen_date
  first_op_id
  last_op_id
  ops_count
  weaknesses_count
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.11 OverallSummarySubclientAccounts

Description

High-level aggregated metrics across all subclient accounts under a parent MSP account. Provides a single-row summary of total activity and findings. Returned by overall_summary_for_subclient_accounts.

Fields

`enrolled_clients_count` - `Int`
Total number of subclient accounts enrolled under the parent account.
`non_empty_ops_count` - `Int`
Number of pentests that discovered at least one endpoint (non-empty results).
`ops_count` - `Int`
Total number of pentests and ops executed across all subclient accounts.
`hosts_or_unique_ips_scanned_count` - `Int`
Total number of unique hosts or IP addresses scanned across all subclient pentests.
`impact_paths_count` - `Int`
Total number of attack paths identified across all subclient pentests.
`weaknesses_count` - `Int`
Total number of weaknesses discovered across all subclient pentests.
`weaknesses_mitigated_count` - `Int`
Total number of weaknesses that have been remediated or mitigated across all subclients.
`total_duration_of_ops_seconds` - `Long`
Cumulative duration of all pentests across all subclients, measured in seconds.

Request Response

```
fragment OverallSummarySubclientAccountsFragment on OverallSummarySubclientAccounts {
  enrolled_clients_count
  non_empty_ops_count
  ops_count
  hosts_or_unique_ips_scanned_count
  impact_paths_count
  weaknesses_count
  weaknesses_mitigated_count
  total_duration_of_ops_seconds
}
```

Examples may not include all available fields for a type.

Admin: MSP

8.47.12 SubclientAccountSummary

Description

All-time summary metrics for a subclient account, including pentest counts, user activity, and enrollment status. Returned by subclient_account_summaries. Requires multi-tenancy to be enabled on the parent account; returns an empty list otherwise.

Fields

`client_account_uuid` - `String!`
Unique identifier of the subclient ClientAccount.
`company_name` - `String!`
Company name of the subclient account.
`enrolled_date` - `Date`
Date when the subclient account was enrolled (created) on the platform.
`ops_count` - `Int`
Total number of pentests and other ops run by the subclient, all-time.
`successfully_launched_ops_count` - `Int`
Number of ops that were successfully launched and executed.
`failed_or_autocanceled_ops_count` - `Int`
Number of ops that were initiated but either failed to launch or were automatically canceled.
`auto_canceled_ops_count` - `Int`
Number of ops that were initiated but automatically canceled before launch.
`failed_ops_count` - `Int`
Number of ops that failed to provision or launch due to platform errors.
`empty_ops_count` - `Int`
Number of ops that completed but discovered no endpoints in scope.
`first_op_date` - `Date`
Date when the subclient ran their first op.
`last_op_date` - `Date`
Date of the most recent op run by the subclient.
`uniq_ips_count` - `Int`
Number of unique IP addresses observed across all pentests for this subclient.
`users_count` - `Int`
Number of active users with access to the subclient account.
`user_sign_ins_count` - `Int`
Total number of user sign-ins across all users, all-time.
`first_sign_in_date` - `Date`
Date of the first user sign-in for this subclient account.
`last_sign_in_date` - `Date`
Date of the most recent user sign-in for this subclient account.

Request Response

```
fragment SubclientAccountSummaryFragment on SubclientAccountSummary {
  client_account_uuid
  company_name
  enrolled_date
  ops_count
  successfully_launched_ops_count
  first_op_date
  last_op_date
  uniq_ips_count
  users_count
  user_sign_ins_count
  first_sign_in_date
  last_sign_in_date
}
```

Examples may not include all available fields for a type.

8.47.13 Boolean

Description

The `Boolean` scalar type represents `true` or `false`.

Example

```
true
```

Types

8.47.14 Date

Description

`String` scalar type with date ISO format serialization, eg. 2021-07-22.

Example

```
"2021-07-22"
```

Types

8.47.15 Datetime

Description

`String` scalar type with datetime ISO format serialization, eg. 2021-07-22T05:02:40.294996 .

Example

```
"2021-07-22T05:02:40.294996"
```

Types

8.47.16 EmailAddress

Description

`String` scalar type with email address format required.

Example

```
"user@example.com"
```

Types

8.47.17 Float

Description

The `Float` scalar type represents signed double-precision fractional values as specified by IEEE 754.

Example

```
987.65
```

Types

8.47.18 ID

Description

The `ID` scalar type represents a unique identifier, often used to refetch an object or as key for a cache. The ID type appears in a JSON response as a String; however, it is not intended to be human-readable. When expected as an input type, any string (such as `"4"`) or integer (such as `4`) input value will be accepted as an ID.

Example

```
"4"
```

Types

8.47.19 Int

Description

The `Int` scalar type represents non-fractional signed whole numeric values. Int can represent values between -(2^31) and 2^31 - 1.

Example

```
987
```

Types

8.47.20 JSONObject

Description

`String` scalar type with JSON serialization.

Example

```
"{\"hello\": \"world\"}"
```

Types

8.47.21 Long

Description

`Int` scalar type alias.

Example

```
5234
```

Types

8.47.22 String

Description

The `String` scalar type represents textual data, represented as UTF-8 character sequences. The String type is most often used by GraphQL to represent free-form human-readable text.

Example

```
"abc123"
```

Types

8.47.23 StringNoWhitespace

Description

`String` scalar type that cannot be an empty string or contain whitespace.

Example

```
"non-empty-string"
```

Types

8.47.24 StringNotEmpty

Description

`String` scalar type that cannot be an empty string.

Example

```
"non empty string"
```

9. Release Notes

9.1 Release Notes

Our Release Notes provide a monthly summary of enhancements, updates, and continuous improvements, giving you a clear and comprehensive overview of the latest changes.

For real-time updates and the latest release info, check notifications directly within the NodeZero Portal.

9.1.1 2026.06

June Updates ☀️ Summer is here, and June brings major upgrades to how NodeZero helps you identify, investigate, fix, and verify real risk across your environment.

- **Rapid Response enhancements** distill vulnerability noise into personalized asset-level exploitability status indications, remediation guidance, and verification feedback.
- **Real-Time View (RTV) redesign** helps you distinguish and deconflict NodeZero testing from real attacks. A visual timeline, event-log summaries, and granular event details all provide flexible filtering options.
- **External Pentesting rebuild** replaces Asset Groups with reusable Scopes, improves bulk asset management and uploads, and enhances support for large external attack surfaces.
- **Critical Rapid Response coverage** for vulnerabilities including Starlette/LiteLLM BadHost, Apache Solr, Palo Alto Networks PAN-OS, Nginx UI, Ivanti Sentry, Check Point Remote Access VPN, Splunk Enterprise, Cisco Unified Communications Manager, Squid, and more.
- **MSP and license-management improvements** make it easier for parent accounts to manage clients' product access, asset allocations, and custom branding.
- **Platform stability improvements** improve Host Discovery, External Asset Discovery, subnet enumeration, credential validation, Runner selection, and pentest comparison workflows.

Check out the 2026.06 Details Page for all the details.

9.1.2 2026.05

May Updates 🌞 As summer approaches, we're bringing the heat with coverage for critical vulnerabilities and weaknesses, an enhanced Vulnerability Management Hub (VMH), and significant improvements to MSP asset management.

- **Rapid Response Coverage** for critical vulnerabilities, including Palo Alto PAN-OS (CVE-2026-0300), Next.js SSRF (CVE-2026-44578), and Drupal SQL Injection (CVE-2026-9082).
- **VMH & Pentest Workflow upgrades** feature new clickable weakness alias badges for quick filtering, searchable runner drop-downs, and clearer naming for report ZIP files.
- **MSP & Asset Management enhancements** now allow parent accounts to view and set contracted asset counts directly via the Asset Allocation modal.
- **Expanded Attack Capabilities** include new Linux privilege escalation implants and NetBIOS support for AD operations.
- **Platform & Identity Fixes** deliver major stability improvements to asset matching and stale-asset detection, alongside robust fixes to SSO and the Portal UI.

Check out the 2026.05 Details Page for all the details.

9.1.3 2026.04

April Updates 🌱 Spring brings fresh updates! This month, we're introducing clearer risk-to-impact diagrams in pentest summaries, faster client asset recovery, and workflow improvements around tests management and inventory exports.

- **Enhanced pentest summaries** display redesigned sankey diagrams with CISA KEV badges to quickly highlight actively exploited vulnerabilities in the wild.
- **Rapid Response coverage** for critical vulnerabilities affecting Fortinet, Cisco, Apache ActiveMQ, ManageEngine, and cPanel vulnerabilities.
- **Pentest resiliency upgrades** including the ability to resume interrupted operations and improved queue handling.
- **Co-Branding** now enables downloading co-branded sample reports in groups, and enforcing parent branding across child accounts.
- **Client deletion** now enables parents to recover the client's scanned assets allocation within 30 days, rather than the prior 60-day hold.
- **Expanded automation** with improved inventory export capabilities.
- **Operational workflow improvements** include the ability to resume interrupted operations, and scheduled actions that automatically supersede active ops.
- **Performance improvements** across dashboards and data loading for faster, more responsive workflows.

Check out the 2026.04 Details Page for all the details.

9.1.4 2026.03

March Updates 🌷 As we head into spring, we’re delivering powerful new attack coverage, expanded Tripwire capabilities, and key performance and usability improvements to help you stay ahead of emerging threats.

- **Rapid Response coverage** includes new tests for Cisco SD-WAN, Citrix NetScaler, GeoServer RCE, and LiteLLM supply-chain vulnerabilities.
- **Expanded Threat Actor Intelligence**, an Elite-tier feature, is available to Pro and Core accounts for a limited time to offer broader visibility into adversary behavior.
- **Tripwires enhancements** including CSV export, credential rotation, and improved Active Directory validation.
- **Multi-tenant alerting improvements**, where email notifications now include account names – making it easier for MSPs and Enterprises to triage alerts across subclients.
- **Platform hardening** includes significant bug fixes for credential sanitization, SSO enforcement, and asset pipeline reliability.

Check out the 2026.03 Details Page for all the details.

9.1.5 2026.02

February Updates 🌷 As spring approaches, we’re blooming with critical updates to our core attack engine, platform-wide usability enhancements, and powerful filtering enhancements to help you focus on what matters most.

- **Rapid Response coverage** for Ivanti EPMM and BeyondTrust vulnerabilities, plus new Kubernetes RCE detection.
- **Cross-domain credential injection** for more-realistic modeling and validation of multi-domain attacks.
- **Filtering and sorting enhancements** across Pentests, VMH (Vulnerability Management Hub), and Results views.
- **Runner visibility upgrades** with Command History, Credentials, and Past Pentests tabs.
- **MSP/MSSP flexibility improvements**: Subclients can now draw from parent asset pools, and white-labeling fixes promote a consistent brand experience.
- **FedRAMP Updates**: FedRAMP High customers can now access a Horizon3-hosted MCP Server.

Check out the 2026.02 Details Page for all the details.

9.1.6 2026.01

January Updates ❄️ We're kicking off the new year with new Rapid Response coverage for major vulnerabilities, expanded attack capabilities, and enhancements to platform usability and API support.

- **New Rapid Response coverage** for widespread vulnerabilities including WatchGuard Fireware remote code execution, MongoDB memory disclosure (MongoBleed), Fortinet FortiSIEM, HPE OneView, and Samsung MagicINFO.
- **Expanded attack content** adding SmarterMail, osTicket, FortiSIEM, HPE OneView, and ManageEngine ADSelfService Plus tests.
- **NodeZero Federal external attack support** enabled across all federal environments.
- **Vulnerability Management Hub (VMH)**: New `No Longer Found` status, plus enhanced sorting and filtering by Host, Weakness ID, and Downstream Impact.
- **Integration Improvements**: Added automated weakness retrieval for Palo Alto Cortex XSOAR, and expanded API support for subclient management.
- **Runners and ARM environment stability improvements** for more reliable operations.
- **API & Assets page enhancements** for improved data visibility and automation workflows.

Check out the 2026.01 Details Page for all the details.

9.1.7 2025.12

December Updates ❄️ As the year closes out, this release delivers meaningful performance improvements, richer vulnerability workflows, expanded Rapid Response coverage for critical CVEs, and significant capability expansion for NodeZero Federal.

- **Vulnerability Management Hub enhancements** with faster grouping, richer filters, and improved status workflows.
- **Expanded Rapid Response coverage** for Gladinet, XWiki, React2Shell, and multiple FreePBX vulnerabilities.
- **High-Value Targeting & ADP refinements**, including clearer labeling and badges across hosts and credentials.
- **Pentest reliability improvements** to reduce processing delays in large environments.
- **NodeZero Federal expansion**, bringing cloud cracking, HVT/ADP, VMH, and Threat Informed Perspectives to Fed customers.

Check out the 2025.12 Details Page for all the details.

9.1.8 2025.11

November Updates 🍂 As the year winds down, we're releasing major enhancements to help you clearly understand and prioritize your highest risks, plus rapid response coverage for critical new vulnerabilities.

- **High Value Targeting updates** with richer explanations and AI-enriched insights.
- **Vulnerability Risk Intelligence (Early Access)** adds improved scanner support, new exports, and easier navigation.
- **Zero-day/rapid response coverage** for FortiWeb, N-able N-central, Gladinet Triofox, WSUS, and more.
- **Cloud cracking fully enabled** across both US and EU for faster credential analysis.
- **External Asset Discovery optimization** delivers up to an **85% faster** experience in large environments.
- Numerous **UI improvements** across Compare, Tripwires, Templates, Weaknesses, VMH, and onboarding flows.

Check out the 2025.11 Details Page for all the details.

9.1.9 2025.10

October Updates 🎃 This spooky season brings no tricks—just a bag full of powerful new features and attack capabilities to strengthen your defenses! Check out the latest enhancements designed to keep your security strategy robust and ready.

- **Managed Control Plane (MCP) GA**: The Hosted MCP Server is now Generally Available See more on MCP.
- **High-Value Targeting (HVT) and Advanced Data Pilfering (ADP)**: Launched High-Value Targeting (HVT) and Advanced Data Pilfering (ADP) to simulate sophisticated targeting and exfiltration. and .
- **New SCCM Compromise**: Added full support for exploiting the **TAKEOVER-1** misconfiguration in Microsoft SCCM, a high-impact internal attack path.
- **Vulnerability Hub Performance**: Significant performance improvements to the Vulnerability Management Hub, including **50% faster** "Weaknesses Over Time" queries and quicker chart loading.
- **Tripwire Enhancements**: Introduced new Active Directory (AD) Tripwire capabilities and a dedicated navigation section.
- **Rapid Response Coverage**: New and updated tests for critical vulnerabilities in Redis, Cisco ASA, Oracle EBS, and WatchGuard Fireware OS.

Check out the 2025.10 Details Page for all the details.

9.1.10 2025.09

September Updates 🍂 As summer transitions into fall, we're bringing you refinements for Active Directory Tripwires, expanded Rapid Response coverage, and improved platform accessibility across account types—ensuring your defenses remain sharp as the season changes.

- **Rapid Response Coverage**: New tests deployed for critical Cisco ASA/FTD vulnerabilities (CVE-2025-20362, CVE-2025-20333), and FreePBX (CVE-2025-57819), helping you stay ahead of emerging threats.
- **MCP Hosted by Horizon3.ai**: An MCP Server hosted by Horizon3.ai is available, providing OAuth 2.1 authentication, support for multiple users, and more.
- **AD Tripwires Refinements**: Enhanced host resolution handling, improved configuration warnings, and consistent visual indicators across the product for Early Adopter customers.
- **EDR Feature Expansion**: EDR capabilities provisioned for all account types including Flex and C+, with improved filtering UX for better usability.
- **Enhanced Asset Management**: Asset filter controls now available to all users, making contract-based asset tracking more accessible.
- **Bug Fixes & Stability**: Resolved issues affecting AD Tripwire management, password audit configuration, and subclient operations.

See 2025.09 Details Page for all the changes.

9.1.11 2025.08

August Updates ☀️ As summer winds down, NodeZero delivers fresh Rapid Response coverage, Active Directory attack expansions, and streamlined client management for MSPs.

- **Rapid Response Tests**: New coverage for Xerox FreeFlow Core, Fortinet FortiSIEM, Citrix NetScaler ADC/Gateway, and CrushFTP — with improved UI showing CVE aliases and IDs.
- **Attack Content Updates**: Added Cisco HyperFlex HX exploit (CVE-2021-1498) plus stronger Active Directory attack paths including domain trust enumeration and Golden Ticket techniques.
- **CISA KEV Status Updates**: Citrix Session Recording (CVE-2024-8069) and Cisco ISE/ISE-PIC (CVE-2025-20281) now flagged as actively exploited.
- **MSP Enhancements**: Clearer account hierarchy with “child of” indicators and subclient assets displayed directly in the Portal.
- **UI Refinements**: Client account columns in templates and improved tooltips make navigation and management smoother.

Check out the 2025.08 Details Page for all the details.

9.1.12 2025.07

July Updates 🚀 This month delivers powerful new insights into how real-world attackers could target your environment, along with high-impact Rapid Response coverage, major MSP upgrades, and the GA release of the MCP Server.

- **Threat Actor Mapping (Elite GA)**: See which adversary groups would exploit your top CVEs, with new visualizations, detailed profiles, and filtering.
- **Rapid Response Coverage**: New tests for CitrixBleed 2, PAN-OS GlobalProtect XSS, Wing FTP, FortiWeb SQLi, CrushFTP, Infoblox NetMRI, and Flowise.
- **Expanded Attack Content**: Critical exploits for Microsoft SharePoint Toolshell, Cisco ISE, FortiWeb, and SonicWall SMA XSS.
- **MSP Enhancements**: Rolled-up asset counts, SSO-only subclient access, and new self-service subclient management.
- **MCP Server GA**: Public image release with Bring Your Own LLM support and enriched Tripwire API integration.
- **Accessibility**: New "No Animation" setting plus visual and content refinements.

Check out the 2025.07 Details Page for all the details.

9.1.13 2025.06

June Updates ☀️ The start of summer brings new ways to streamline your security journey! This month’s release introduces the new Vulnerability Management Hub, a smoother Flex onboarding flow, and expanded MSP tooling—plus fresh Rapid Response and general attack content.

Vulnerability Management Hub: Now generally available to help you prioritize, verify, and remediate what matters most.

Flex New User Wizard: A guided workflow for Flex customers to quickly set up clients, runners, and pentests.

MSP Functionality Improvements: Better visibility and easier management for multi-tenant accounts.

New Attack Content: Rapid Response tests for ZendTo and Infoblox vulnerabilities, plus new general tests and fixes.

Check out the 2025.06 Details Page for all the details.

9.1.14 2025.05

May Updates 🌻 As the season shifts, so does NodeZero—introducing new access options, attack resilience, and comparative insight. This month's updates expand platform flexibility, deepen credential access capabilities, and enhance Tripwire deployment reliability across environments.

- **Flex Access Level**: A new pay-as-you-go option for teams needing one-time or ad hoc pentesting, without a continuous testing commitment.
- **Improved Credential Access**: NodeZero now uses stealthier techniques to dump credentials from SAM and LSA, boosting success rates against modern EDR protections.
- **RAT for AARCH64**: The NodeZero Remote Access Tool now supports aarch64 Linux targets, expanding post-exploitation coverage across platforms.
- **SMB-Based Tripwire Deployment**: Tripwires can now be deployed using the SMB protocol, making deployment more reliable in cases of implant failure.

Check out the 2025.05 Details Page for all the details.

9.1.15 2025.04

April Updates 🌹 Spring growth continues with impactful new features across NodeZero—from hands-on Tripwire orchestration to scalable MSP tools and fresh attack coverage. This release boosts operational flexibility, enhances accessibility, and expands detection precision for cloud and enterprise environments.

- **Tripwire Job Orchestration**: Launch custom Tripwire jobs directly from the Dashboard or Tripwires page, enabling targeted sensor deployment beyond standard compromise triggers. Track deployment progress with enhanced visibility in Job Details and Manage pages.
- **Multi-Tenant Client Management**: MSPs can now move pentests between subclients, rename accounts, and reset passwords—simplifying operations at scale.
- **Trial API Keys + CLI Enhancements**: Trial users can now generate API keys, and the H3 CLI supports environment variables to streamline scripting workflows.
- **UX & Accessibility Improvements**: Enhanced screen reader support, license and role visibility in the org switcher, and template safeguards against IPv6 misconfigurations.
- **New Attack Content**:
- **Commvault RCE – CVE-2025-34028**: Path traversal enables unauthenticated remote code execution.
- **SAP NetWeaver – CVE-2025-31324**: Unauthenticated file uploads lead to system compromise.
- **Craft CMS – CVE-2025-32432**: Deserialization of malicious objects allows full RCE.

Check out the 2025.04 Details Page for all the details.

9.1.16 2025.03

March Updates 🌷 As spring approaches, we're blooming with new capabilities across NodeZero—from improved user experience and performance boosts to expanded insights and post-exploitation techniques. This release strengthens operational efficiency and security visibility across the platform.

- **MSP Subclient Management**: MSPs using asset-based SKUs can now create, manage, and deallocate subclients with improved visibility into asset recovery timelines.
- **Kubernetes Operator Upgrade**: Enhanced pod introspection improves host/IP matching accuracy during Kubernetes pentests.
- **RAT Enhancements**: NodeZero can now steal Chrome cookies, decrypt them offline, and hijack Okta sessions—plus deploy implants via Metasploit.
- **Attack Graph Improvements**: Better visualization and traceability of exploit chains make pentest results easier to interpret.
- **Performance & Stability**: Pentest post-processing runs up to 80% faster, with auto-healing pipelines and improved ActionLog query speed.

Check out the 2025.03 Details Page for all the details.

9.1.17 2025.02

February Updates ❄️ While the winter chill lingers, we're heating things up with powerful new enhancements and attack content to keep your security posture strong. Check out this month's updates designed to refine pentest workflows, improve insights, and expand attack capabilities.

- **Improved Pentest Scheduling**: Enhancements to the template creation experience, sticky action buttons, and new scheduling options make it easier than ever to set up and run pentests.
- **Scope Discovery Tuning**: Users can now adjust speeds to reduce network disruptions during scope discovery.
- **Insights Enhancements**: New **Mean-Time-To-Remediation (MTTR) trends** and **CISA KEV filtering** provide deeper insights into vulnerability mitigation.
- **New Attack Content**: Fresh attack capabilities targeting Fortinet, HuangDou UTCMS, Active Directory privilege escalation, and enhanced WordPress and GitHub token verification.
- **Bug Fixes & Stability Improvements**: Multiple UI and backend fixes, including refinements to scheduled pentests, improved AWS credential handling, and better report accuracy.

Check out the 2025.02 Details Page for all the details.

9.1.18 2025.01

January Updates ❄️ A fresh start to the year brings powerful new features and attack content to keep your security posture strong! This month's release focuses on **enhanced workflows, expanded attack coverage, and rapid response to critical vulnerabilities**—ensuring you stay ahead of emerging threats.

- **New Insights & Tripwires Enhancements**: A redesigned **Insights Dashboard** and a **new landing page for Tripwires** improve usability and visibility.
- **Pentest Scheduling Workflow**: New users now get **guided onboarding** for continuous penetration testing with streamlined setup.
- **Expanded External Attack Content**: Improved scanning for **Microsoft IIS, GitHub security, Azure credentials, and more**.
- **Rapid Response: Ivanti EPM Vulnerabilities**: Immediate testing for four critical Ivanti CVEs before public disclosure.
- **Bug Fixes & UX Improvements**: Enhanced template management, Azure credential handling, and other key refinements.

Check out the 2025.01 Details Page for all the details.

9.1.19 2024.12

December Updates ❄️ As we wrap up the year, we're excited to bring you a host of new features and enhancements designed to bolster your security posture during the holiday season. This month's update is packed with powerful attack content and crucial improvements to keep you ahead of emerging threats.

- **NodeZero Insights**: Introducing our new Insights product, providing a comprehensive data analytics and management dashboard that empowers CIOs and CISOs to launch and monitor organization-wide security initiatives and compliance checklists.
- **Improved Workload Type Identification for K8s**: Enhancements to Kubernetes workload identification make penetration testing assessments more accurate, simplifying the evaluation of Kubernetes environments.
- **Expanded Attack Content**: This month's update includes critical vulnerabilities affecting PAN-OS, CyberPanel, ProjectSend, and Cleo Harmony, enhancing your ability to detect and respond to threats across diverse platforms.

Check out the 2024.12 Details Page for all the details.

9.1.20 2024.11

November Updates 🍂 As we dive into the heart of fall, we're bringing you new features and enhancements to keep your security game strong through the colder months. This update is packed with exciting additions, including enhanced pentesting capabilities, new attack content, and important bug fixes to help you stay ahead of evolving threats.

- **New Infrastructure Attack Surface Test**: NodeZero now supports Kubernetes pentests, allowing users to test pods, nodes, clusters, and namespaces for security vulnerabilities, making it easier to secure modern hybrid-cloud environments.
- **New Operational Scenario Test**: Introducing the Insider Threat Pentest—now you can confirm the blast radius of potential insider threats by injecting Windows AD credentials during test setup.
- **Expanded Attack Content**: This month's attack content includes critical vulnerabilities targeting PTZOptics cameras, QNAP Photo Station, TP-Link routers, D-Link devices, and more!

Check out the 2024.11 Details Page for all the details.

9.1.21 2024.10

October Updates 🎃 This spooky season brings no tricks—just a bag full of powerful new features and attack capabilities to strengthen your defenses! Check out the latest enhancements designed to keep your security strategy robust and ready.

- **MFA-Protected Azure Credentials**: NodeZero now supports injecting MFA-protected Azure credentials using Microsoft's Device Code Flow.
- **Tripwires for MSP Accounts**: Tripwires is now available for MSP Client and Sub-Client accounts, with an improved alerting UX.
- **Expanded Azure and AWS Attack Capabilities**: New tools for Azure KeyVault pilfering, AWS Cognito misconfiguration exploitation, and enhanced privilege escalation detection in AWS environments.
- **New Attack Content**: Critical vulnerabilities added this month target Palo Alto, CloudStack, CUPS, Ruckus Wireless, AVTECH CCTV, Zimbra, CyberPanel, SharePoint, and more!

Check out the 2024.10 Details Page for all the details.

9.1.22 2024.09

September Updates 🍂 As the crisp days of fall begin, we're rolling out new features and powerful attack content to keep your cybersecurity strategies sharp. Check out the latest updates that will help fortify your defenses this month.

NodeZero Tripwires has been made Generally Available! See the Tripwires section for more information on this exciting new capability!

- **1CV Badge & Summary Table**: Easily identify and track 1-Click Verify (1CV) pentests with new badges and summary tables.
- **Enhanced Rapid Response Filtering**: Improved tagging and filtering options make it easier to find relevant Rapid Response tests.
- **Expanded Weaknesses CSV**: Export more comprehensive data with new columns for Impact, Mitigations, and References.
- **New Attack Content**: This month's attack content includes critical vulnerabilities affecting Ivanti, Sitecore, Apache Struts2, Mitel, Microsoft SQL Server Reporting Services, and more.

Check out the 2024.09 Details Page for all the details.

9.1.23 2024.08

August Updates 🍎 As summer winds down and the back-to-school season kicks into gear, we're bringing you the latest updates to sharpen your cybersecurity defenses. Whether you're squeezing in last-minute summer adventures or prepping for a productive fall, these new features and attack content will help keep your organization safe and secure.

- **Rapid Response Access for Team Members**: Org Admins can now extend Rapid Response alert access to other team members, allowing for broader collaboration and faster reaction times.
- **Co-Branding Sample Reports**: New functionality for MSP/MSSP/C+ licenses allows for co-branding of sample operation reports, making it easier to align with your organization's branding.
- **AWS Pentest Enhancements**: New Cloud Connection types, expiration settings, and improved capabilities for inspecting cloud resources.
- **New Attack Content**: This month's attack content includes critical updates targeting vulnerabilities in SolarWinds Web Help Desk.

Check out the 2024.08 Details Page for all the details.

9.1.24 2024.07

July Updates 🎆 As we celebrate the height of summer, we're excited to bring you an array of new features and enhancements to strengthen your cybersecurity defenses. Check out this month's updates, designed to empower your security strategies.

- **NodeZero AWS Pentest**: Introducing a comprehensive gray-box pentest utilizing a role with read-only permissions to assess the security of your AWS accounts.
- **NodeZero Azure Entra ID Pentest**: This new gray-box pentest uses an injected Entra ID credential and a privileged Active Directory credential to analyze attack paths in your hybrid environment. It includes an advanced configuration option for Entra ID App/Directory Role Privilege Escalation.
- **New Pentest Setup Workflow**: A revamped full-page UX for running pentests, allowing selection across four categories: Infrastructure Attack Surface, Identity Attack Surface, Operational Scenario Testing, and Rapid Response.
- **Attack Content**: Attack content affecting GeoServer, WhatsUp Gold, Palo Alto, OpenSSH, Cisco Smart Software Manager, and more!
- **Added capabilities**: Enhanced AWS analysis capabilities, including Lambda function code extraction and EC2 instance user data analysis.

Check out the 2024.07 Details Page for all the details.

9.1.25 2024.06

June Updates ☀️ As summer brings longer days filled with sunlight, we are excited to introduce a wave of enhancements and new attack vectors to the NodeZero platform. Dive into our latest updates, designed to brighten your security outlook.

- **AWS Canary Tokens Detection**: Improved detection capabilities for AWS Canary Tokens to enhance accuracy and avoid false alerts.
- **Office 365 Azure Token Pilfering**: Expanding our RAT capabilities to capture Azure access tokens from Office 365 applications, potentially escalating to serious compromises.
- **SolarWinds and Ubiquiti Exploits**: New exploits for SolarWinds Serv-U and Ubiquiti UniFi Video vulnerabilities, targeting critical infrastructures.
- **Enhanced Credential Leak Checks**: Added sophisticated checks for exposed NTLM endpoints and a new classification for a Jupyter Windows Credential Leak.
- **Rapid Response and CVE Exploits**: Rapidly deploying tests for emerging threats such as PHP-CGI Command Injection and new exploits for Adobe Commerce, Telerik Report Server, and Veeam Backup.
- **Additional Attack Content**: This month's updates also feature new attack content targeting specific technologies including Ivanti Endpoint Manager, Apache HugeGraph, PHP-CGI, Jupyter, and Veeam Backup Enterprise Manager, enhancing our extensive coverage across a variety of critical systems.

Check out the 2024.06 Details Page for all the details.

9.1.26 2024.05

May Updates 🌻 As May blossoms unfold, so do new features and updates in our platform, ensuring your cybersecurity is as vibrant and resilient as spring itself. This month, we've introduced several significant enhancements that will enrich your security measures and streamline your testing capabilities.

- **Rapid Response Alert Center**: Dive into proactive security with our newly launched Rapid Response Alert Center. It equips your organization with early alerts and actionable intelligence, enabling you to stay ahead of emerging threats.
- **Phishing Pentest Enhancements**: We've enhanced the Phishing Pentest Summary Page with new visual metrics and detailed mappings of phished credentials to their impacts, improving your insight into the phishing landscape.
- **Template Management Enhancements**: Experience greater flexibility in scheduling with our improved template management UI, now featuring customizable scheduling options.
- **New Attack Content**: Our attack repertoire has expanded with:
- Enhancements to our Remote Access Tool (RAT) capabilities.
- New exploits for vulnerabilities in Exchange, ColdFusion, FortiSIEM, and Traccar, bolstering our coverage and response to the latest threat vectors.

Check out the 2024.05 Details Page for all the details.

9.1.27 2024.04

April Updates ☔ April showers bring not just flowers, but also a deluge of enhancements to fortify your digital defenses! This month, we're rolling out new features designed to proactively empower your organization against the evolving threat landscape.

- **Rapid Response Alert Center**: Stay ahead of cyber threats with our new Rapid Response Alert Center. It provides early alerts and actionable intelligence, allowing you to proactively address vulnerabilities before they're exploited widely. This strategic addition is designed for real-time defense adjustments, ensuring you're always prepared.
- **New Attack Content**: Enhance your defense with our latest attack modules targeting:
- Progress Kemp LoadMaster for remote command execution.
- Entra Seamless SSO to forge Entra ID credentials.
- Azure Instance Metadata Service for querying sensitive data.
- AWS RDS databases to check and exploit default database credentials.
- GlobalProtect for critical infrastructure protection.
- MLFlow for targeting specific machine learning workflows.
- And more!
- **Auto-injected Azure Credentials**: Boost your Azure operations with auto-injected credentials now available for NodeZero scheduled pentests, enhancing both efficiency and security posture.

Check out the 2024.04 Details Page for all the details.

9.1.28 2024.03

March Updates ☘️ March marches in with the promise of new growth and our commitment to continuous innovation. This month, we've cultivated a crop of robust features and enhancements aimed at strengthening your security landscape. Spring into action with our latest update highlights:

- **Rapid Response Tests**: Spring into action with our new Rapid Response tests, tailored to help you surgically test and verify the most critical and emerging vulnerabilities within your environment.
- **New Attack Content**: New content like the Fortinet FortiClient EMS SQL injection vulnerability that leads to remote code execution, and other high-profile CVEs.
- **Template Management Page**: Organize and streamline your attack templates with our newly designed template management UI.
- **Active Directory Password Audit**: Enhanced performance, now dumping NTDS secrets 10 times faster.

Check out the 2024.03 Details Page for all the details.

9.1.29 2024.02

February Updates ❤️ Love is in the air, and so is the promise of enhancing cybersecurity with our February updates. This month, we're delivering a bouquet of new features and improvements, all designed to sweeten your security strategy. From user interface enhancements to testing your defenses from the attacker's perspective, let our latest offerings be your Valentine's gift from NodeZero.

- **Sticky Table Headers**: Navigate large data tables with ease, thanks to sticky headers that stay in view as you scroll.
- **Dashboard Views**: Customize your dashboard experience with new vertical and horizontal layout options, ensuring the most critical information is always where you need it.
- **Cookie Consent for GDPR**: Enhance user privacy with our updated cookie consent feature, now in compliance with GDPR.
- **New Attack Content**: Stay ahead of attackers with new attack modules, including CVEs targeting Ivanti Connect Secure, GitLab, ConnectWise SecureConnect, and more.
- **Azure and AWS Enhancements**: Gain deeper insights and control in cloud environments with our latest Azure user creation and AWS metadata service credential harvesting capabilities.

Check out the 2024.02 Details Page for all the details.

9.1.30 2024.01

January Updates ❄️ As the new year begins amidst the quiet chill of winter, our team has ignited a beacon of innovation to heat up your security strategy. This January, we're rolling out powerful updates aimed at enhancing your cybersecurity posture. Embrace new beginnings and make a resolution to harden your infrastructure with our latest developments from the attacker's perspective!

- **Phishing Impact Test**: A significant leap forward with the launch of the Phishing Impact Test in NodeZero, enabling organizations to measure the potential impact of phishing with precision. See our phishing page
- **Attack Path Enhancements**: Major improvements to Attack Paths, introducing a Vertical Display and Concise/ Detailed views, for a clearer narrative on your security landscape.
- **External Asset Discovery Updates**: A series of updates to enhance the identification and management of external assets.
- **New Attack Content**: Expanding our arsenal with critical vulnerabilities targeting Ivanti Connect Secure VPN, Fortra GoAnywhere MFT, Apache OFBiz, Jenkins, and more, alongside inclusion of 21 vulnerabilities from the CISA KEV list.

Check out the 2024.01 Details Page for all the details.

9.1.31 2023.11

November Updates 🦃 As November ushers in the crispness of late autumn, our team has been busy harvesting a rich array of updates and enhancements. This month, we present a bountiful selection of new features and refinements, each designed to fortify and streamline your security landscape. Step into November's technological cornucopia and explore what we've cultivated for you!

- **New Attack Content**: Attack content targeting Cisco IOS XE, Citrix NetScaler, Apache ActiveMQ, and Confluence.
- **Cyanide Activity Identification**: Enhanced NodeZero activity identification in logs with updated Cyanide, now including a static suffix 'H3N0' for simplified tracking.
- **Azure Attack Flow Improvements**: Strengthened Azure integration with the ability to use Azure Refresh and Access Tokens, streamlining the authentication process.
- **Advanced Pentest Management**: Introducing new functionalities like moving pentests between accounts and downloading key pentest data such as External IPs and AD Password Audit results in CSV format.

Check out the 2023.11 Details Page for all the details.

9.1.32 2023.10

October Updates 🎃 As the nights grow longer and Halloween shadows creep in, we've conjured up a spellbinding set of updates for you this month. Like a cauldron brimming with potions, our platform brews with enhancements to bewitch and bolster your security endeavors. Dive in, if you dare!

- **Credential Injection with Node Zero Runners**: Node Zero Runners now support automatic credential injection for scheduled operations, requiring zero manual input post-setup. Especially useful for monthly Active Directory Password Audits, ensuring process adherence and catching overlooked policy errors.
- **Revamped Fix Actions Report**: The newly refreshed fix action report offers an intuitive table of contents and detailed insights, pinpointing affected hosts for each identified weakness. It's a consolidated resource for action-based insights.
- **Enhanced Exposure Score Visibility**: The pentest summary now displays an Overall Exposure Score, derived from a meticulous assessment of critical impacts, weaknesses, and data exposure. Improve your security by addressing these highlighted vulnerabilities.

Check out the 2023.10 Details Page for all the details.

9.1.33 2023.09

September Updates 🍂 As the leaves turn golden and begin their descent, we're thrilled to unveil a flurry of fresh features this autumn. Just as trees are shedding layers, we added layers of innovation in September!

- **Expanded Attack Content**: New content for Citrix devices, Azure VM access, Adobe Coldfusion, advanced password spray, and more!
- **Phishing Integration**: Dive into NodeZero brand-new test type and seamlessly integrate it with your Phishing campaigns.
- **NodeZero Runner Resilience**: Use `h3-cli` for effortless registration of your NodeZero Runner as a system service.
- **Enhanced Data Discovery**: See "Protected Data" results during pentests for more insightful findings.
- **Revamped Executive Summary**: Discover our refreshed, intuitive design

Check out the 2023.09 Details Page for all the details.

9.1.34 2023.08

August Updates 🌞 As the summer sun continues to shine bright, so do our platform enhancements! We've brought in a fresh wave of updates this month, aiming to make your experience more seamless and engaging.

- **Enhanced Proxy Support:** Easier and more streamlined proxy configurations.
- **Expanded Coercion Methods:** New methods added to exploit PetitPotam vulnerabilities.
- **Improved Single Sign-On (SSO) Experience:** Open beta for paid accounts.
- **Portal UI Updates:** Introducing new color themes ("Modern" and "Light") and redesigned navigation bar for enhanced user interaction.
- **Phishing Impact Test (Beta):** Introducing new Phishing Impact Test to measure the impact of phishing attacks.
- **Feature Additions:** Added attack content for Juniper, cPanel, H2 Database, Adobe ColdFusion, and Metabase.

Check out the 2023.08 Details Page for all the details.

9.1.35 2023.07

The only thing hotter than July is all the new features. Here are some highlights:

- **New/Updated Vulnerability Detections**: Added several new detections and exploits for weaknesses.
- **Password Spraying**: Improved dynamic generation of weak passwords.
- **External Host Discovery**: Expanded NodeZero coverage and accuracy for identifying hosts during external enumeration.
- **Domain Controller Identification**: Added better domain controller identification in adverse networks.

Check out the 2023.07 Details Page for all the details.

9.1.36 2023.06

Summer is here, along with a release packed with great new features! Here are a few highlights:

- **Single Sign-On (SSO) Integration**: Added support for Single Sign-On using OpenID Connect (OIDC).
- **Password Audit Operations**: Easily audit the strength and similarity of user passwords in your Active Directory environment.
- **Remote Access Tool (RAT)**: NodeZero can now leverage detected weaknesses and vulnerabilities to deploy Remote Access Tools (RATs).
- **Bulk Authorize External Assets**: The External Assets page has improved ability to sort, filter, and bulk-authorize assets.
- **NodeZero Runner Management**: New Runner Management page improves visibility and control over your Runners.
- **BloodHound Data Collection**: NodeZero now collects BloodHound data during Pentest operations, which can be downloaded post-op.

Check out the 2023.06 Details Page for all the details.

9.1.37 2023.05

After the abundance of amazing attacks in April, we're thrilled to share even more May blooms with you! Here are some of the key highlights:

- **Pentest Scheduling in the Portal**: Say goodbye to manual configurations! You can now easily schedule future pentests and series of pentests directly in the Portal, streamlining your workflows. See the scheduling page for more information.
- **VirtualHost Support for Kubernetes**: NodeZero now supports VirtualHosts in Kubernetes modules, providing enhanced testing capabilities for containerized environments.
- **Real-Time View Enhancements**: Gain deeper insights with Real-Time View updates for External Pentests, including status updates for injected credentials. Stay on top of the progress with real-time information.
- **Portal Login Enhancements**: Experience enhanced authentication capabilities with a new Social Sign-In button for Microsoft/Azure.
- **Attack Content Updates**: As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.05 Details Page for all the details.

9.1.38 2023.04

This release packs in some great new features, including NodeZero Runners and the Network Enumeration operation type. A few highlights for this release include:

- Introduction of `NodeZero Runners`, which enable automated deployment of NodeZero without needed to copy-paste the curl script.
- The `Network Enumeration` operation, the first of several Targeted Tests, which enables you to discover the attack surface of your internal network without identifying or exploiting vulnerabilities.
- Added ability for users to inject credentials immediately after scheduling a pentest and while a pentest is paused
- Added new visualizations and filtering to the Hosts Page
- Enhanced password spray and password cracking routines to utilize usernames from breach data
- As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.04 Details Page for all the details.

9.1.39 2023.03

Spring is in the air, and with it comes the latest updates to NodeZero! March brings a fresh breeze of features and improvements to help your cybersecurity program bloom. Check out the highlights below, or view in detail on the March 2023 page.

- New user experience enabled by default!
- The new user experience is now enabled by default. Customers that had access to the old experience can still switch back for a limited time.
- As always, we're continually keeping NodeZero up-to-date with important exploits and attack techniques.

Check out the 2023.03 Details Page for all the details.

9.1.40 2023.02

This release is filled with amazing new capabilities and we are excited for you to use them

View the 2023.02 Details Page for detailed explanations, but here are some highlights:

- New User Experience: This is a whole new look and feel to the Portal. We've revamped the executive summary and made it much easier to navigate through the results of your pentest
- Externally pentest IP Addresses: Available in the new user experience, you can now add IPs to the scope for an External pentest
- Pause and Resume pentest operations: Available in the new user experience, you can now pause and resume ops from the Portal
- 1-Click-Verify multiple weaknesses at a time: Available in the new user experience
- H3 CLI: You can now schedule a pentest to run automatically on a recurring basis using the H3 CLI tool

And much more. Check out the 2023.02 Details Page for all the details.

9.1.41 2023.01

Happy New Year!

This month's release improves functionality in user interface and additional attack content including:

- VMware vRealize Log Insight VMSA-2023-0001
- Active Directory Certificate Services (ADCS) ESC8
- Additional Cloud Attack Content
- Multiple CISA KEVs

View the 2023.01 Details Page for detailed explanations, enhancements, and bugfixes!

9.2 2026.06

9.2.1 Features & Enhancements

Personalized Rapid Response Guidance

Our redesigned Rapid Response capability cuts through the noise of emerging security advisories by identifying which active CVEs (Common Vulnerabilities and Exposures) and Horizon3.ai-identified weaknesses pose an exploitability risk to your specific environment and assets. This release introduces a comprehensive remediation and verification workflow:

- Personalized insights defining your organization's risk exposure to each escalated threat.
- A dedicated advisory page for each threat, with full visibility into assets classified as Exploitable, Potentially Relevant, and Not Exploitable.
- Proactive offensive tests on external assets (for opted-in customers).
- Actionable, personalized remediation guidance, with embedded workflows to assign and verify fixes.

Real-Time View (RTV) Redesign

The Real-Time View (RTV) of ongoing pentests has been redesigned to help SOCs (Security Operations Centers) distinguish and deconflict benign NodeZero scanning operations from active, hostile attacks:

- **Action Log Timeline** – A graphical event display featuring an adjustable time-span window.
- **Event Metadata Table** – Advanced filtering controls, enabling security teams to search by Target Entity, Module, and event ID to correlate with alerts in their security tools.
- **Granular Event Logs** – Raw event logs exported in OCSF (Open Cybersecurity Schema Framework) format for easy copying and local investigation.

External Pentesting

External Pentesting has been rebuilt around discovering, scoping, authorizing, and testing external attack surfaces at larger scale.

- Replaced the legacy Asset Groups model with reusable Scopes, while keeping legacy Asset Groups available.
- Added External Assets v2, which combines domain, Internet Protocol (IP) address, and test status information in a unified view.
- Added bulk asset management for 10,000 to 100,000+ assets.
- Added support for bulk-uploading CSV (Comma-Separated Values) or plain-text files containing asset information.
- Added support for tagging third-party infrastructure so it cannot be tested.
- Added support for bulk-authorizing owned assets, and for launching pentests against all authorized assets by default.
- Added direct loading support for large Classless Inter-Domain Routing (CIDR) ranges from `/16` through `/32`.
- Adjusted domain-name validation to prevent authorization behavior that could incorrectly block legitimate domains.

MSP Client & License Management

- **MSP Client Product Management** – Parent Managed Service Provider (MSP) accounts can now retrieve and configure asset-toggle, enforcement values, and contracted asset counts across Clients within backend systems and frontend license-management views.
- **Product Selection Drawer** – Moved product add-on toggles (including Tripwires, Rapid Response, and Risk-Based Vulnerability Management (RBVM)) to a dedicated **Products** submenu in the **Update Client** drawer.
- **Sankey Diagram Visual Enhancements** – Improved the visualization quality and structure of Sankey diagrams to provide clearer mapping of credential flows.
- **Runner Template Query Improvements** – Enhanced back-end template queries to increase performance and reliability when rendering custom scan templates.
- **Entitlement Access Controls** – Implemented read-only entitlement states for Campaigns, Perspectives, and Vulnerability Management pages to support restricted access roles.
- **Maximum Runtime Configuration** – Updated the `Max Runtime` setting to represent absolute wall clock time.
- **Quick Filters** – Implemented quick filters and badge replacements for more streamlined data sorting and navigation.

Portal & Workflow Improvements

- Updated **Max Runtime** to represent wall clock time.
- Added a new Executive Summary Segmentation (PDF) Report.
- Improved template queries for better reliability.
- Added read-only entitlement handling for Campaigns, Perspectives, and Vulnerability Management pages.
- Implemented quick filters and badge replacements for more efficient navigation.

Tripwires

- Improved the robustness of mounting AD (Active Directory) agent service account credentials.
- Improved credential validation reliability when both Server Message Block (SMB) and Secure Shell (SSH) services are present.

New Attack Content

- **Starlette and LiteLLM Unauthenticated Remote Code Execution (CVE-2026-48710 and CVE-2026-42271)** – Released targeted testing for a critical exploit chain (known as BadHost) where an authentication bypass in the Starlette Asynchronous Server Gateway Interface (ASGI) framework is combined with a command injection flaw in LiteLLM Model Context Protocol (MCP) server preview endpoints, allowing unauthenticated remote attackers to execute arbitrary operating system commands and access sensitive Large Language Model (LLM) API keys.
- **Apache Solr Basic Authentication Hardcoded Credentials (CVE-2026-44825)** – Released targeted testing for a vulnerability in Apache Solr where hardcoded default credentials are silently installed in the basic authentication setup tool, allowing remote attackers to gain full administrative access to the Solr cluster.
- **Palo Alto Networks PAN-OS GlobalProtect Authentication Bypass (CVE-2026-0257)** – Added targeted testing for an authentication bypass vulnerability in Palo Alto Networks PAN-OS GlobalProtect portal and gateway configurations, which allows remote, unauthenticated attackers to bypass security restrictions and establish unauthorized Virtual Private Network (VPN) sessions.
- **Nginx UI Unauthenticated Backup Disclosure (CVE-2026-27944)** – Added coverage for a vulnerability in Nginx UI where an unauthenticated backup endpoint exposes sensitive system backup archives along with the encryption keys required to decrypt them, risking exposure of user credentials, session tokens, and Secure Sockets Layer (SSL) private keys.

- **Progress Kemp LoadMaster Command Injection (CVE-2024-7591)** – Added a targeted test for an improper input validation vulnerability in Progress Kemp LoadMaster, which allows unauthenticated remote attackers to execute arbitrary operating system commands via crafted HTTP requests.
- **Ivanti Sentry Operating System Command Injection (CVE-2026-10520)** – Released targeted testing for a critical command injection vulnerability in Ivanti Sentry that allows unauthenticated remote attackers to execute arbitrary operating system commands with root-level privileges.
- **Oracle PeopleSoft Enterprise PeopleTools Remote Code Execution (CVE-2026-35273)** – Added coverage for a critical, unauthenticated remote code execution vulnerability in the Updates Environment Management component of Oracle PeopleSoft Enterprise PeopleTools, which can allow an unauthenticated attacker to take complete control of affected systems over HTTP.
- **Check Point Remote Access VPN Authentication Bypass (CVE-2026-50751)** – Released a targeted test for an authentication bypass vulnerability in Check Point Remote Access VPN and Mobile Access services using deprecated IKEv1 key exchange, which allows unauthenticated remote attackers to establish VPN sessions without providing a valid password.
- **SimpleHelp OpenID Connect Authentication Bypass (CVE-2026-48558)** – Added coverage for a critical authentication bypass vulnerability in SimpleHelp remote support and monitoring software where improper OpenID Connect (OIDC) token validation allows unauthenticated attackers to forge identity tokens and gain unauthorized technician sessions with full administrative access.
- **Splunk Enterprise PostgreSQL Sidecar Service Arbitrary File Write (CVE-2026-20253)** – Released targeted testing for a critical missing authentication vulnerability in the PostgreSQL sidecar service endpoint of Splunk Enterprise and Splunk Cloud Platform, which allows unauthenticated remote attackers to perform arbitrary file creation or truncation operations.
- **Cisco Unified Communications Manager Server-Side Request Forgery (CVE-2026-20230)** – Added targeted testing for a critical server-side request forgery (SSRF) vulnerability in the Cisco Unified Communications Manager WebDialer service, which allows remote, unauthenticated attackers to execute arbitrary requests and write files to the underlying operating system.
- **Squid Proxy Heap Buffer Overread (CVE-2026-47729)** – Released targeted testing for a heap buffer overread vulnerability (known as Squidbleed) in the File Transfer Protocol (FTP) directory-listing parser of Squid Web Proxy, which allows attackers to coerce the proxy into reading adjacent heap memory and disclosing sensitive data, including cleartext HTTP credentials and session tokens from other users.
- **Active Directory Ticket Granting Ticket Generation** – Added a Ticket Granting Ticket (TGT) creation submodule to simulate Authentication Service Request (AS-REQ) exploitation and capture credentials.

9.2.2 Platform Performance & Stability

- Optimized External Pentesting for larger asset sets.
- Added yield-based rescan pruning to improve scope discovery efficiency.
- Improved Known Exploited Vulnerabilities (KEV) detection on fragile network appliances, including Palo Alto Networks PAN-OS and Fortinet appliances.
- Improved Active Directory credential database ingestion performance for Kerberos keys.
- Improved external asset database operations at large scale, including workflows with up to 300,000 assets.
- Improved subnet enumeration behavior and reserved subnet handling.
- Improved network inventory query performance to reduce page-loading cost growth.
- Improved Host Discovery behavior to avoid unnecessary repeated discovery runs.

9.2.3 Bug Fixes

- Fixed a NetBIOS domain name population issue that could cause credential verification failures.
- Fixed Active Directory credential database ingestion performance after Kerberos key ingestion.

- Fixed a password spraying issue where more than five attempts in 15 minutes could unintentionally lock out users.
- Fixed incorrect admin-panel tagging for H3-2025-0002.
- Softened Host Discovery enforcement to log TTL (Time-to-Live) variance across ports rather than dropping hosts entirely.
- Fixed Host Discovery parsing for port-unreachable responses.
- Fixed Host Discovery behavior that could incorrectly mark scopes as ineligible for rescans.
- Fixed scope filtering behavior during repeated Host Discovery runs.
- Resolved a bug where Host Discovery execution loops would trigger recursively.
- Fixed DNS zone lookup behavior when multiple Pointer (PTR) records are present.
- Fixed a duplicate-key race condition in scope domain linking.
- Fixed subnet enumeration behavior in which the same skip token could be returned repeatedly.
- Restored pentest execution capabilities for FLEX-license customers.
- Fixed template configuration UI's unintended reset every few minutes.
- Fixed Runner selection behavior when a previously used Runner was not included in the initial query results.
- Fixed operation form behavior so Runner and Kubernetes fields can be cleared correctly.
- Fixed pentest comparison pagination by removing the first-100-pentests limit.
- Fixed missing single sign-on (SSO) test button and Initiator URL behavior in the EU Portal.
- Fixed an SSO setup issue where the Initiator URL did not populate.
- Fixed a login issue affecting users restricted to SSO-only authentication.
- Fixed a credential extraction timeout issue.
- Fixed the Vulnerability Management Hub (VMH) Unprocessed Pentest modal to show the end date instead of the start date.
- Fixed premature form resets by adding template checks before resetting the run operation form.

---

deral

Users of NodeZero Federal might experience a 1–2 week delay in the availability of some features, Attack Content, or bug fixes.

9.3 2026.05

9.3.1 Features & Enhancements

Vulnerability Management Hub (VMH) & Pentest UX

- Added **weakness alias badges** within the Vulnerability Management Hub to improve identification and filtering of related weaknesses.
- Users can now **search directly within the Runner selection drop-down** when launching operations, simplifying Runner selection in large environments.
- Downloaded external pentest report ZIP files now use clearer naming conventions that reflect the associated test and the included report files.

Rapid Response & Threat Coverage

- Added Rapid Response coverage for newly disclosed vulnerabilities affecting enterprise software platforms and web applications.
- Added Linux privilege escalation implant support for additional copy/fail privilege escalation techniques associated with CISA Known Exploited Vulnerabilities (KEVs).

Credential & Attack Path Enhancements

- Added support for **NetBIOS domain names** during Active Directory auditing workflows and secretsdump operations, improving credential validation coverage in complex Windows environments.

MSP & Asset Allocation Improvements

- MSP parent organizations can now **view and manage contracted asset counts** for subclients directly from the Asset Allocation interface.
- Added an **asset lock status badge** to improve visibility into temporary asset allocation restrictions.

9.3.2 New Attack Content

- **Palo Alto PAN-OS User-ID Authentication Portal Buffer Overflow** – CVE-2026-0300
  A critical buffer overflow in the PAN-OS User-ID Authentication Portal (Captive Portal) that allows unauthenticated remote code execution as root on affected PA-Series and VM-Series firewalls.
- **Next.js Server-Side Request Forgery** – CVE-2026-44578
  A server-side request forgery (SSRF) flaw in the Next.js WebSocket upgrade handler that allows unauthenticated attackers to proxy requests to internal services or cloud metadata endpoints.
- **Drupal SQL Injection** – CVE-2026-9082
  A critical SQL injection vulnerability in Drupal's database abstraction API that allows unauthenticated attackers to execute arbitrary SQL commands on systems using PostgreSQL databases.
- **XWiki Platform Path Traversal** – CVE-2026-23734
  A path traversal vulnerability in the XWiki Platform that could allow an attacker to bypass access controls and read unauthorized files or directories on the underlying system.

- **Apache ActiveMQ Authentication Bypass** – CVE-2026-40466
  An authentication bypass vulnerability in Apache ActiveMQ that could allow unauthorized actors to circumvent security controls and access the broker.
- **Linux Privilege Escalation** Added implant support for Linux copy/fail privilege escalation, which is now tracked as a CISA Known Exploited Vulnerability.
- **NetBIOS Domain Name Support** Added support for NetBIOS domain names and their use with `secretsdump` in Active Directory Audit operations.

9.3.3 Platform Performance & Stability

- Improved Vulnerability Management Hub reliability for environments with a large number of users and associated operations.
- Enhanced backend reliability in asset processing and host matching, to improve data consistency across large environments.
- Improved stale-asset detection logic, to strengthen asset health validation workflows over long periods.

9.3.4 Bug Fixes

- Fixed schedule picker validation issues affecting valid cron expressions containing comma-separated lists.
- Fixed a Ruckus SSH detection false positive that could incorrectly report successful credential validation.
- Fixed client-management console warnings in the portal UI.
- Fixed SSO Initiator URL population issues during SSO (single sign-on) configuration.
- Fixed missing SSO Test button and Initiator URL issues in the EU portal.
- Fixed the handling of failed basic-auth logins for SSO-only users.
- Fixed large user environments' improper updating of Vulnerability Management Hub data.
- Fixed host-matching edge cases that could incorrectly associate findings across operations.
- Fixed the Portal's color rendering of MFA (multi-factor authentication) screen text.

Federal

Users of NodeZero Federal might experience a 1–2 week delay in the availability of some features, Attack Content, or bug fixes.

9.4 2026.04

9.4.1 Announcements

- Our ServiceNow integration will soon be deprecated, although it remains available for use. Horizon3 expects to release an enhanced ServiceNow integration on June 3.

9.4.2 Features & Enhancements

Attack Path Visualization

- Within pentest summaries, redesigned **sankey diagrams** show a clearer flow from threat actors through to weaknesses, impacts, and business risks. We've added KEV badges on weaknesses that map to CISA Known Exploited Vulnerabilities, helping you quickly prioritize weaknesses actively exploited in the wild.
- We've also made these diagrams more responsive across a wide range of screen sizes.

Co-Branding

- You can now download co-branded sample reports in groups.
- You can now (optionally) enforce a parent account's branding across child accounts.

Client Assets Recovery

- On the Portal's Client Management page, admins can now **schedule deletion** of a client to return the client's allocation of scanned assets to the parent's pool in as soon as 30 days. This replaces the prior 60-day hold, improving asset reuse and operational flexibility.

Pentest & Runner Operations

- Added the ability to **resume interrupted pentests** directly from the pentest detail page.
- Scheduled actions now **automatically take precedence** over active operations on runners, reducing manual intervention.
- Users are now **notified when operations fail to dequeue** due to inactive runners.
- Added protections against **runner queue misordering** caused by rapid user interaction.

Inventory & Data Export

- Improved **CSV export capabilities**, including support for selected items, IP ranges, and scopes.
- Enhanced inventory data with new fields such as **last external domain** and **last host tab**.
- Added improved filtering and sorting support across inventory tables.

Vulnerability Management & Credential Insights

- Improved handling of **unverified credentials** with clearer confidence indicators and filtering options.
- Enhanced logic for identifying **“no longer found” vulnerabilities** with improved metadata tracking.

UX Improvements

- Improved **pentest table load performance** by parallelizing data and pagination queries.
- Added **quick-select duration options** for access grants to streamline common workflows.
- Runner names remain **accessible as links** even after decommissioning for better historical analysis.
- Clarified **pentest removal messaging** to reduce confusion around data retention.
- Improved **modal feedback and loading states** across pentest actions.
- Added **confirmation dialogs** for sensitive actions like archiving pentests.
- Adjusted **spinner sizing** in icon buttons for consistency.
- Fixed minor grammar and typo errors in Runner queue **notification emails**.

9.4.3 New Attack Content

- **cPanel and WHM Login Flow Authentication Bypass** – CVE-2026-41940
  An authentication bypass vulnerability in the cPanel and WHM login flow that could allow unauthenticated attackers to gain unauthorized access to the system.
- **ManageEngine Log360 Authentication Bypass** – CVE-2026-3324
  An authentication bypass vulnerability in ManageEngine Log360 that could allow unauthenticated attackers to bypass authentication mechanisms and access sensitive data.
- **Fortinet FortiClient EMS Improper Access Control** – H3-2026-0012
  An improper access control vulnerability in Fortinet FortiClient EMS that could allow an attacker to bypass intended access restrictions and perform unauthorized actions.
- **Fortinet FortiClient EMS SQL Injection** – CVE-2026-21643
  A SQL injection vulnerability in Fortinet FortiClient EMS that could allow an unauthenticated attacker to execute arbitrary SQL queries against the underlying database.
- **Cisco Smart Software Manager On-Prem Arbitrary Command Execution** – CVE-2026-20160
  A vulnerability in Cisco Smart Software Manager On-Prem that could allow an unauthenticated attacker to execute arbitrary commands on the affected system.
- **Apache ActiveMQ Jolokia Remote Code Execution** – H3-2026-0008 / CVE-2026-34197
  A remote code execution vulnerability in Apache ActiveMQ Jolokia that could allow an attacker to execute arbitrary code on the target server.
- **Fortinet FortiClient EMS Improper Access Control** – CVE-2026-35616
  An improper access control vulnerability in Fortinet FortiClient EMS that could allow an attacker to bypass intended access restrictions.
- **DNS Hostname Enumeration** Added support for domain-joined Linux machines to have their DNS hostnames properly enumerated during internal operations.
- **OCI User Permission Enumeration** Added a new module for enumerating Oracle Cloud Infrastructure (OCI) user permissions.

9.4.4 Platform Performance & Stability

- Improved **portal and dashboard performance**, including faster loading for RBVM-related views.
- Reduced perceived latency by optimizing **data loading and query execution** across key tables and dashboards.

9.4.5 Bug Fixes

- Fixed a bug in **ADCS (Active Directory Certificate Services) modules** that caused them to fail for certain Certificate Authorities.
- Resolved a bug that caused findings to be missed if web endpoints returned a **non-success HTTP response** (non-`2xx` or non-`3xx` error code).
- Fixed external scanning logic to properly detect **non-standard ports**.
- Fixed **Content Security Policy** headers to properly support co-branded deployments.
- Corrected **date preset calculations** for access grant durations, to ensure that "end of week" covers through Sunday, and "end of month" covers through the final day of the month.

Federal

Users of NodeZero Federal might experience a 1–2 week delay in the availability of some features, Attack Content, or bug fixes.

9.5 2026.03

9.5.1 Features & Enhancements

Rapid Response & Alerting Improvements

- Rapid Response alert emails now include **account context** (account name, Rapid Response name, and associated weakness IDs) for organizations with parent/subclient structures, improving clarity in multi-tenant environments.
- Simplified the **Rapid Response selection UI** when configuring pentests.

Tripwires Enhancements

- Added the ability to **export all Tripwire data to CSV**, including key metadata such as name, type, file path, monitored processes, status, and deployment date.
- Tripwire jobs now support **Active Directory credential validation** against domain-joined systems.
- Introduced new utility to assist in **rotating service account passwords**.

Threat Actor Intelligence Expansion

- **Threat Actor Intelligence** features are temporarily extended to Pro and Core tier accounts, to assist a broader range of customers in defending against increased geopolitical cyber threats.

Security & Data Protection

- **FedRAMP & Security Controls** – Users of NodeZero Federal are now warned when sensitive data leaves the Portal. Implemented a Content Security Policy (CSP) header for enhanced Web security.
- We now enforce stricter SSO-only access where SSO is configured.
- We now prevent including deleted accounts during sign-in resolution

9.5.2 New Attack Content

- **Cisco Catalyst SD-WAN** – CVE-2026-20127
  Coverage for vulnerabilities in Cisco Catalyst SD-WAN Manager and Controller that could allow an unauthenticated, remote attacker to gain unauthorized access to the affected systems.
- **Cisco Secure Firewall Management Center (FMC)** – CVE-2026-20079 & CVE-2026-20131
  Detection and targeted testing for authentication bypass and unauthenticated remote code execution (RCE) vulnerabilities in the Cisco FMC web-based management interface.
- **NetScaler ADC & Gateway** – CVE-2026-3055
  Detection for a memory overread vulnerability in Citrix NetScaler when configured as a SAML Identity Provider, which could lead to the disclosure of sensitive information.
- **GeoServer XPath Injection** – CVE-2024-36404
  Added an exploit module for a critical RCE vulnerability in GeoServer where unauthenticated users can execute arbitrary code via specially crafted XPath expressions.
- **Argo Workflows** – CVE-2026-28229
  Introduced version detection and metrics for a vulnerability in Argo Workflows that could impact cluster security.

- **SSH ControlMaster Hijack** – H3-2026-0007
  New attack module targeting the abuse of SSH ControlMaster sockets, which can allow an attacker to hijack existing SSH sessions without requiring credentials.
- **ADCS ESC11 Support** – Added support for the ESC11 attack path within Active Directory Certificate Services (ADCS), expanding NodeZero's ability to identify credential theft and privilege escalation via web enrollment.
- **LiteLLM Supply Chain Attack** – Rapid Response coverage and customer notifications regarding the BerriAI LiteLLM credential stealer supply chain vulnerability.

9.5.3 Platform Performance & Stability

- Improved **GraphQL service connection handling** to reduce latency and prevent degradation under high load.
- Optimized query performance and pagination for large datasets across multiple services.
- Fixed issues with background job processing and scheduling reliability.

Reliability Enhancements

- Improved **host discovery reliability** and stability during large-scale operations.
- Enhanced asset processing pipelines to prevent data inconsistencies and stuck operations.
- Fixed issues affecting scheduled and template-based pentest execution.

Usability Enhancements

- **Runner Detail Upgrades** – Added a new **Schedules** tab to the Runner detail page, allowing users to view all pentest schedules associated with a specific runner.
- **Advanced Command Filtering** – Improved the Agent Commands view with pagination controls and faceted filtering by exit status, enabling more efficient analysis of runner activity.

9.5.4 Bug Fixes

- Fixed CSV export failures caused by backend processing errors.
- Resolved a bug where **co-branding settings were overwritten** during account synchronization.
- Corrected issues with **external asset discovery search** and unreachable IP filtering.
- Fixed duplicate and inconsistent results in credential and asset queries.
- Resolved incorrect **asset usage percentage calculations**.
- Fixed errors affecting **scheduled and templated pentests**.
- Addressed issues with **NodeZero launch scripts**, including timezone handling and EDR detection.
- Improved **password and hash sanitization** across multiple attack modules.
- Fixed incorrect **operating system detection** in certain network scan scenarios.
- Resolved issues with **SCCM enumeration and LDAP binding behavior**.
- Fixed false-positive findings in password reuse detection.
- Corrected UI issues including a flashing 404 page during login, incorrect footer colors in light mode, and broken spinner loaders on certain buttons.
- Fixed authentication and access control edge cases, including SSO enforcement and account routing.
- Resolved issues affecting **asset pipeline processing**, including missing data and duplicate records.
- Fixed issues with **bulk email handling** to improve privacy and delivery behavior.

9.5.5 New Trial Accounts

To initiate product trials, Horizon3 AI now asks that you contact our Sales team. You can do so via the horizon3.ai site's **See a Demo** or **Speak with an Expert** links.

Federal

Users of NodeZero Federal might experience a 1–2 week delay in the availability of some features, Attack Content, or bug fixes.

9.6 2026.02

9.6.1 Features & Enhancements

NodeZero Federal – MCP Server Access

- FedRAMP High customers can now access a Horizon3-hosted **MCP Server** at: https://mcp.gov-horizon3ai.com/mcp. This simplifies secure model connectivity for federal environments.

Risk-Based Vulnerability Management (RBVM)

VULNERABILITY MANAGEMENT HUB (VMH)

- Added **port**, **protocol**, and **IP byte count** fields to VMH data for richer technical context.
- Clarified **first seen** and **last seen** timestamps in the API.
- Expanded filtering and faceted search across:
- Operators
- Runners
- Templates
- Tickets (including “assigned by”)
- Impacts
- Weaknesses

These improvements make it easier to analyze and prioritize vulnerabilities at scale.

High-Value Targeting (HVT)

- Added new **HVT categories** and improved hostname and username pattern matching to enhance business risk identification.
- Refined HVT prompts for improved consistency and accuracy in AI-assisted classification.

Cross-Domain Credential Injection

- Added the ability to inject **cross-domain credentials** into Password Audit and internal pentests, enabling more realistic multi-domain attack path validation.

Runner UI Enhancements

- Expanded the Runner detail page with new tabs covering:
- **Command History**
- **Credentials** injection
- **Past Pentests**

Reporting Improvements

- Pentest report downloads now include an optional **LLM action log** for greater transparency into AI-assisted operations.

MSP Subclient Flexibility

- MSP subclients without explicit asset allocations can now run operations using the parent organization's licensed asset pool, improving operational flexibility.

Filtering & Sorting Enhancements

Significant filtering and sorting improvements across the platform:

PENTESTS

- Runners: Kubernetes namespace, User, User role
- Kubernetes Operators: Version, Chart version, User
- Templates: Runner name; sortable by min/max run times

VULNERABILITY MANAGEMENT HUB (VMH)

- Tickets: Asset category, Status, Severity, User
- Sort by Asset count and User

PENTEST RESULTS

- Impacts: Weakness name, Credential name, Host name, Impact type
- Weaknesses: Weakness name/ID, Category, Impact type
- Credentials: Username, IP address, Impact type
- Data: Service type, Data store type, IP address, Protocol & Port, Permissions, Name
- Hosts: Access roles, Service types, Impact types; sortable by operating system

---

9.6.2 New Attack Content

- **Ivanti Endpoint Manager Mobile (EPMM)** – CVE-2026-1281
  An authentication bypass vulnerability that allows unauthenticated attackers to access protected functionality, potentially leading to unauthorized system access.
- **BeyondTrust** – CVE-2026-1731
  A vulnerability that could allow an attacker to execute arbitrary code or gain elevated privileges on affected systems, depending on configuration.
- **LDAPS Support**
  Added LDAPS support to internal attack modules to expand secure Active Directory enumeration and exploitation paths.
- **Kubernetes Service Account RCE Identification**
  Introduced detection of a Kubernetes-based remote code execution condition where service accounts with `nodes/proxy` GET permissions can be abused to execute commands against cluster nodes.

9.6.3 Platform Performance & Stability

Host Discovery Improvements

- Improved host discovery logic for operations with scopes smaller than a `/15` CIDR range, improving reliability and consistency of scan coverage in large environments.

Database & Backend Optimizations

- Improved GraphQL request logging to reduce database write load and lower the risk of Portal latency spikes.
- Tuned database idle connection timeouts to reduce connection pool exhaustion under load.
- Fixed intermittent background job timeouts when connecting to the operations database, improving data pipeline reliability.
- Optimized internal weakness-series processing to reduce execution time and resource consumption.
- Corrected tenant backfill logic so jobs resume from the proper checkpoint instead of restarting.

9.6.4 Bug Fixes

- Fixed Windows 11 hosts being misidentified as Windows 10.
- Updated ADCS ESC8 cryptography handling to use supported libraries.
- Resolved silent SMB verification failures when passwords began with a dash (`-`).
- Updated build process to use a patched responder version to reduce unintended network strain during credential poisoning scenarios.
- Fixed an NTDS password cracking timeout issue that caused certain operations to wait indefinitely.
- Updated AD Tripwire installer to use culture-invariant timestamps.
- Fixed co-branding display issues in the Portal.
- Corrected CSV reporting so parent reports properly include subclient assets.
- Fixed blank client names appearing in the “Move Pentest” modal.
- Corrected Rapid Response Alerts page loading and empty states.
- Fixed launch script behavior to properly remove old containers before starting new ones.
- Ensured read-only users see disabled (not hidden) action buttons for consistent UX.
- Corrected asset allocation enforcement edge cases for MSP environments.
- Improved Entra ID refresh token verification logic to maintain compatibility with Microsoft changes.
- Tripwire alert emails now clearly include the associated account name.

Federal

Users of NodeZero Federal might experience a 1–2 week delay in the availability of some new features, Attack Content, or bug fixes.

9.7 2026.01

9.7.1 Features & Enhancements

Vulnerability Management Hub (VMH)

- **New System Status**: Added a "No Longer Found" status to improve tracking of remediated vulnerabilities over time.
- **Enhanced Table Controls**: Added the ability to sort the vulnerability management table by **Host** and **Weakness ID**.
- **Advanced Filtering**: Users can now filter the table by **Downstream Impacts**, providing better visibility into how specific weaknesses contribute to larger attack paths.
- **Security Posture Assessment**: Improved the filtering and context for security posture assessment text, including extended link coverage and refined filter logic.

Platform & Usability Enhancements

- **NodeZero Federal**: External attack capabilities are now enabled for all NodeZero Federal environments.
- **Asset Intelligence**: The Assets page now supports unique asset attributes, allowing users to customize columns to match the flexibility of the Pentests page.
- **Palo Alto Cortex XSOAR Integration**: Added the `fetch-incidents` command to the integration, enabling the automated retrieval of NodeZero-discovered weaknesses into XSOAR.
- **Runner Management**:
- Improved Docker image cleanup logic to run prior to disk-space checks, preventing unnecessary update failures.
- Enhanced environment checks to detect and notify users of incompatible ARM architectures.
- **API & Integrations**:
- Extended authorization roles to allow the `USER` role to access subclient account endpoints.
- Updated GraphQL documentation to include the `agents_page` property, detailing how to expose multiple Runners for parent accounts.

9.7.2 New Attack Content

- **WatchGuard Fireware OS** (CVE-2025-14733): An out-of-bounds write vulnerability that could allow an unauthenticated attacker to achieve remote code execution (RCE).
- **MongoDB Information Disclosure** (CVE-2025-14847): Also known as "MongoBleed," this vulnerability allows unauthorized access to sensitive system information.
- **SmarterTools SmarterMail** (CVE-2025-52691): An unauthenticated file upload vulnerability that can be exploited to gain unauthorized access or execute code on the mail server.
- **Enhansoft osTicket** (CVE-2026-22200): A PHP filter chain injection vulnerability that allows for remote code execution.
- **Fortinet FortiSIEM** (CVE-2025-64155): An arbitrary file write vulnerability that allows unauthenticated attackers to achieve remote code execution on affected systems.
- **HPE OneView** (CVE-2025-37164): A remote code execution vulnerability in HPE’s infrastructure management software; currently listed on the CISA KEV catalog.
- **ManageEngine ADSelfService Plus** (CVE-2025-11250): An authentication bypass vulnerability that could allow attackers to gain unauthorized access to the self-service portal.
- **Samsung MagicINFO 9 Server** (H3-2026-0001): Coverage for directory traversal and remote code execution vulnerabilities in the digital signage management platform.

- **SolarWinds Web Help Desk** (CVE-2025-40551): A remote code execution vulnerability affecting help desk instances.

9.7.3 Platform Performance & Stability

- **AD Password Audit Optimization**: Improved performance and accuracy of Active Directory password audits by correcting partial reporting issues with cracked hashes.
- **Data Pipeline Reliability**: Resolved onboarding failures for customers using tenant-specific databases within the insights module.

9.7.4 Bug Fixes

- **AD Password Audit**: Fixed an issue where successfully cracked password hashes were occasionally missing from the results page.
- **PDF Reporting**: Resolved several PDF generation bugs, including incorrect page numbering, missing Table of Contents links, and character rendering issues.
- **Authorization Handling**: Fixed 403 errors that occurred for read-only accounts attempting to access integration settings or when users navigated the Portal before permissions were fully synchronized.
- **Account Management**:
- Fixed a bug where the `Assets Allocated` field would default to zero instead of pre-populating with existing values.
- Ensured child accounts correctly inherit asset-based licensing status from their parent accounts.

> **Federal**
>
> Users of NodeZero Federal might experience a 1–2 week delay in the availability of some new features, Attack Content, or bug fixes.

10. Downloads

10.1 Downloads

To make deploying and managing your NodeZero host and data quicker and easier, Horizon3.ai offers these utilities to help.

10.1.1 Host Setup and Management

To deploy NodeZero, a corresponding host is also needed. These tools and utilities help with setting that up.

- Quickly deploy a compatible host using the OVA with VMWare vSphere or VirtualBox.
- Use the VHD to deploy in Windows Hyper-V environments.
- Set up and manage hosts with the host utility.
- Upgrade existing Ubuntu installations to be compatible NodeZero Host.
- Verify that your host is set up and configured to work with NodeZero, prior to running a pentest.

Using the OVA — Recommended

Get a jump start on host deployment by using the preconfigured OVA.

Make Ubuntu Compatible

Update an existing Ubuntu image to enable support for NodeZero.

Host Utility

Utility to help with setup and management of a NodeZero host.

Check Host Compatibility

Run this to confirm that requirements are met for a NodeZero host.

10.1.2 Integrations

NodeZero generates lots of data, which can be overwhelming and difficult to sort through. Here are sone filtering options.

- **Locally-Hosted NodeZero MCP Server**: Allows the user to filter their results quickly, to get only the essential information they need.
- **Splunk App**: Funnels all the data generated by a NodeZero pentest into Splunk. Put your Splunk-foo skills to work!

Locally-Hosted MCP Server

Set up and configure a locally-hosted NodeZero MCP server to use with AI.

Splunk Integration

Slurp NodeZero pentest results into a Splunk environment.

(Another option is the Horizon3.ai-hosted MCP Server, which doesn't require a download.)

10.1.3 Verification and CLI

- Horizon3.ai **highly** recommends verifying all downloads from Horizon3.ai, to ensure that you are getting the authentic files.
- The `h3-cli` command-line interface is available to run custom jobs and configurations, from wherever it is installed.

Verify Downloads

How to verify all downloads from Horizon3.ai. Man-in-the-Middle (MITM) attacks are real.

CLI

The command-line interface to interact with NodeZero.

10.2 NodeZero Host Virtual Machine (OVA/VHD)

The NodeZero Host virtual appliance is a small virtual machine based on a preconfigured Ubuntu 24.04 installation. It's designed to execute NodeZero pentests and bundles tools that facilitate pentest execution, as well as debug and maintenance.

10.2.1 Downloads

> **Tip**
>
> Always verify that the downloaded files come from Horizon3.

VMWare/Virtualbox Importable OVA

Download | SHA256

Windows Hyper-V Importable VHD

Download | SHA256

10.2.2 Specifications

The NodeZero host virtual machine comes preconfigured to use these resources:

- 2 x CPUs
- 8GB of RAM
- 128GB of disk (Minimum Required: 40GB)
- Bridged network adapter

10.2.3 Outbound Traffic

If your environment restricts access to external sites through an outbound proxy or similar mechanism, please make sure that this mechanism allows connections to the following sites:

Port/Protocol	Endpoints	Purpose

HTTPS - 443/ TCP	`*.ubuntu.com`
	`*.canonical.com`
	`downloads.horizon3ai.com`
		API access, authentication, storage, updates, and container registry

10.2.4 OVA Setup

After downloading the NodeZero OVA, follow these steps to set it up and configure your environment.

Installation

Installing the virtual machine is as simple as **importing the OVA file** into your virtualization environment. Below, we provide step-by-step instructions for deploying **NodeZero** using **VMWare vSphere**, **Oracle VirtualBox** or **Hyper-V**.

Choose your virtualization environment and follow the view step-by-step instructions:

VMWare vSphere VirtualBox Windows Hyper-V

Deploying NodeZero on VMWare vSphere

The vSphere client is one of VMWare's virtual environment management solutions. You can find more information on the client itself in VMWare's documentation.

> **Note**
>
> The following steps are for vSphere client version `7.0.3.00500`.

After downloading and verifying the most recent `NodeZero-####.ova` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Log into your `VMware vSphere` client.
2. Open the `Actions` menu and select `Deploy OVF Template`.
3. Choose `Local File`, then click `Upload Files` and select the downloaded OVA file.
4. (Optional) Rename the VM and select a deployment location. Click `Next`.
5. Select `compute resources` for the VM. Click `Next`.
6. Verify the import settings and ensure the signature is from `Horizon3.ai`. Click `Next`.
7. Choose the `storage destination` for the VM. Click `Next`.
8. Select the `network` for the VM. Click `Next`.
9. Review all selections, then click `Finish` to start deployment.
10. Once deployment completes, select the VM from the list and click `Power On` to launch it.

Deploying NodeZero on VirtualBox

Oracle's VirtualBox is a powerful open-source virtualization solution for running multiple operating systems on a single physical machine. You can find more information in VirtualBox documentation.

After downloading and verifying the most recent `NodeZero-####.ova` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Open VirtualBox.
2. Click on `Tools`, then `Import`.
3. Enter the location of the OVA file. Click `Continue`.
4. Click `Import`, and wait for it to complete.
5. Make sure the network is set to use a `bridged network adapter`.
6. Select the newly imported NodeZero virtual machine from the list on the left.
7. Click `Settings`, then `Network`.
8. Confirm that `Attached to` is set to `Bridged Adapter`.
9. Confirm that `Name` is set to the name of the adapter connected to your internal network.
10. Click `OK`.
11. Select the NodeZero virtual machine from the list on the left.
12. Launch the VM by clicking `Start`.

Deploying NodeZero on Windows Hyper-V

Hyper-V is Windows' native virtualization platform for creating and managing virtual machines on Windows systems. You can find more information in Hyper-V documentation.

After downloading and verifying the most recent `NodeZero-####.vhd` file, follow these steps to import and launch the NodeZero host virtual machine.

1. Open and enable Hyper-V.
2. Ensure both `Management Tools` and `Platform` are enabled.
3. Create a New Virtual Machine in Hyper-V Manager by going to `Action > New > Virtual Machine...`.

Accessing Your OVA Host

Choose your virtualization environment to set up the initial connection:

VMWare vSphere **VirtualBox / Hyper-V**

Initial connection on VMWare vSphere

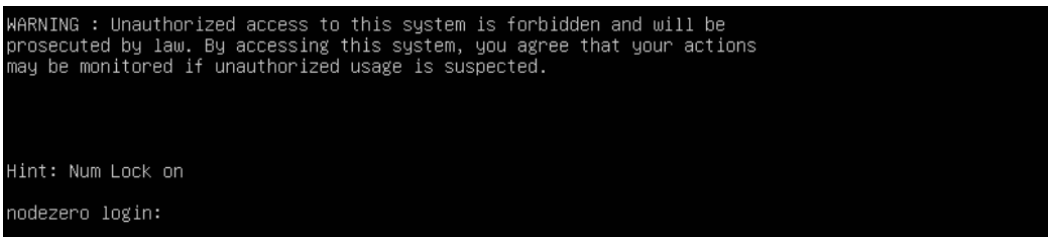
Once the NodeZero host is powered on, you can connect using either the **Web Console** or **Remote Console** from the vSphere client interface.

Initial connection on VirtualBox / Hyper-V

After starting the VM, a display window will appear, showing the operating system loading screen.

1. FIRST-TIME LOGIN & PASSWORD UPDATE

Regardless of the platform, once the OS has fully loaded, you will see a **login screen** similar to this:



Login screen

When launching the **NodeZero Host** for the first time, **SSH password access is disabled** until the initial login is completed and the default password is updated.

Initial Login Credentials

Use the following credentials to log in:

- **Username:** `nodezero`
- **Password:** `nodezero`

2. PASSWORD UPDATE PROCESS

Upon successful login, you will see the following prompt:

```
You are required to change your password immediately (administrator enforced)
Changing password for nodezero.

Current password:
```

- Enter the **current** password (nodezero) and press `Enter`.
- When prompted for the `New password:`, enter a secure password and press `Enter`.
- When prompted to `Retype new password:`, confirm by entering the same password, and press `Enter`.

```
WARNING : Unauthorized access to this system is forbidden and will be
prosecuted by law. By accessing this system, you agree that your actions
may be monitored if unauthorized usage is suspected.



nodezero login: nodezero
Password:
You are required to change your password immediately (administrator enforced).
Changing password for nodezero.
Current password:

Passwords require min 14 characters and must include at least
1 Digit, 1 Special, 1 Upper and 1 Lower Case

New password:
Retype new password:
```

Initial login and password-change prompt

3. SELECT REGION

Once you've successfully changed the password, you must select the region to which the host will connect.

```
    Please select which NodeZero region to connect to:
        1) Standard (US)
        2) GDPR Compliant (EU)

    Enter the number corresponding to your region: 1
[+] Logging out for changes to take effect...
```

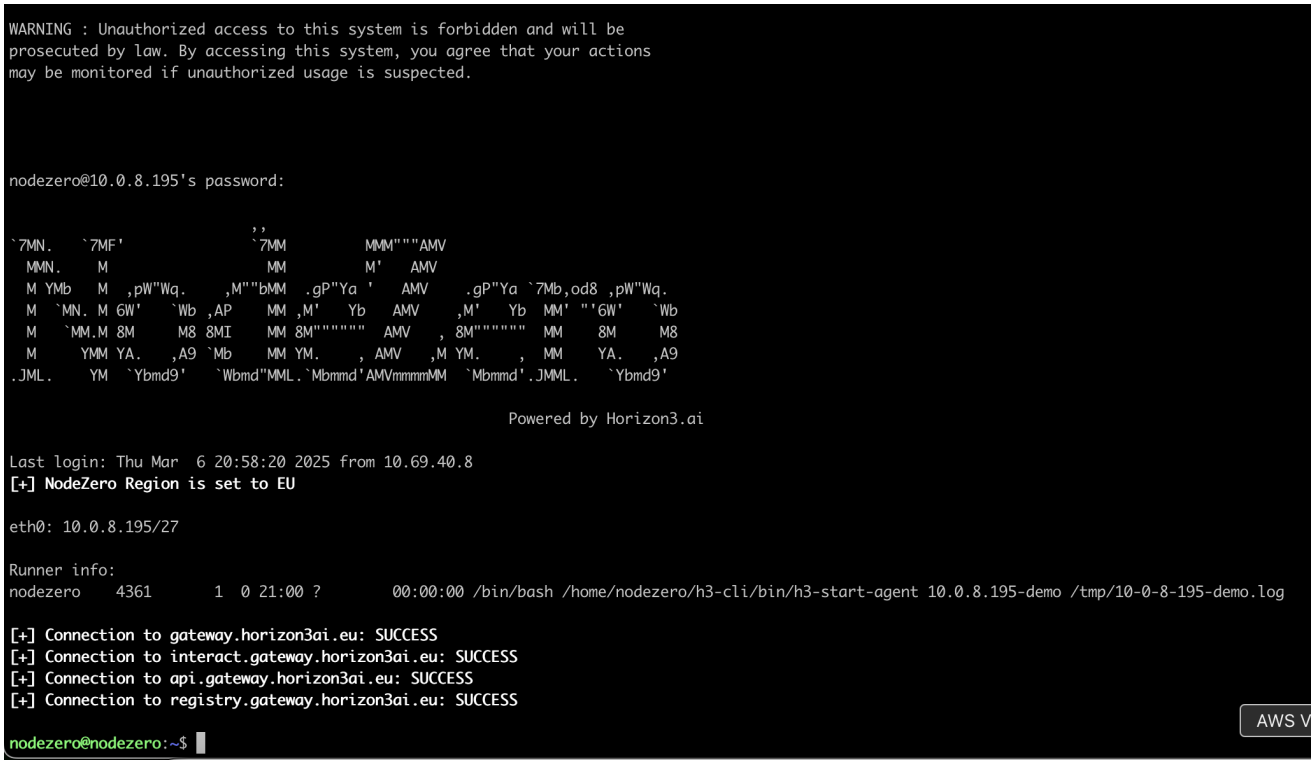
Region selection prompt

After specifying which region the NodeZero host will connect to, log out and back into the system for the change to take effect. Upon logging in, a splash screen will display:

- eth0
- A list of Runner services that are active
- A status of connections to Horizon3.ai endpoints

At this point, **SSH has been enabled** on the host, which is accessible using an SSH client. Below are screenshots of the login screen when nothing is configured on the host, and then when the host is fully set up.

Screenshot of the OVA login with no network or Runners set up



Screenshot of the OVA login when a network is set up and a Runner is added to the host.

4. SSH ACCESS

To connect over SSH with Linux or MacOS, run the command below, replacing `<IP_ADDRESS>` with the one shown in the login screen after `eth0`.

```
ssh nodezero@<IP_ADDRESS>
```

If using Windows, a client like PuTTY will be needed to connect. Fill out the `Host Name (or IP Address)` field with the address shown in the login screen.

Additional setup options

The n0 utility helps with configuring the host with a static ip, using a proxy, downloading the latest CLI, and setting timezones as well as running the checkenv script

EDRs on NodeZero Host

Do not install or configure any EDR (Endpoint Detection and Response) services on the NodeZero host. EDRs are known to disable NodeZero once they detect the deliberately malicious actions that our tests conduct. To get a complete test of your overall environment, NodeZero attacks must launch from the host machine unimpeaded.

10.2.5 Running a NodeZero Pentest

1. Log into the NodeZero Portal.
2. Create a new pentest, providing the relevant setup information.
3. Copy and paste the curl command from the Portal into the shell of a NodeZero host.
4. The pentest starts executing, with indicators like those in the screenshot below.

Initial output from pentest script

10.2.6 Automating NodeZero

NodeZero supports automating the running of itself through the use of the `h3-cli` and setting up a Runner on the host. This can be done by following the steps for setting up a Runner

10.3 Ubuntu Setup Script

10.3.1 Set Up Ubuntu for Running NodeZero

OVA is preferred

We highly recommend using our OVA for NodeZero installation, as it is the preferred method to ensure a seamless setup.

If deploying the NodeZero OVA is not a viable option for your infrastructure, Horizon3.ai provides a script to help set up a base Ubuntu 20.04/24.04 LTS for use with NodeZero.

10.3.2 Minimum Specifications

The NodeZero host requires a minimum configuration of:

- 2 x CPUs (x86_64, x64, or AMD64)
- 8GB of RAM
- 40GB of disk
- Bridged network adapter

10.3.3 Before Setting Up

If your environment restricts access to external sites through an outbound proxy or similar mechanism, please make sure that it allows connections to the following sites:

- `*.ubuntu.com`
- `*.canonical.com`
- `downloads.horizon3ai.com`
- `download.docker.com`

EDRs on NodeZero Host

Do not install or configure any EDR (Endpoint Detection and Response) services on the NodeZero host. EDRs are known to disable NodeZero once they detect the deliberately malicious actions that our tests conduct. To get a complete test of your overall environment, NodeZero attacks must launch from the host machine unimpeaded.

10.3.4 Downloads

Tip

Always verify the files download come from Horizon3.

NodeZero Ubuntu Setup Script

Download SHA256

10.3.5 Running the Script

> **User permissions**
>
> A user with `sudo` permission is required. Do not install as `root`.

Download the script onto the host that is going to be running NodeZero, execute and run the script. After running the script it can be removed.

```
curl -o linux-build.sh https://downloads.horizon3ai.com/utilities/linux-build.sh
bash linux-build.sh
sudo rm -f linux-build.sh
```

> **System updates and reboots**
>
> Executing this script "as is" will update the OS libraries that do not require user interaction. The script will automatically reboot the OS upon completion.

10.3.6 What the Script Does

This script sets up ubuntu with the required libraries for NodeZero to run successfully on the host machine. It sets the MOTD letting users that login know they are logging into a NodeZero host and it installs 2 utilities (`h3-cli` and `n0`)

h3-cli allows for communication and management of NodeZero on the host.

Host Utility enables management of the NodeZero hosts settings.

This script does the following:

- Disables Journald
- Disables the firewall (ufw)
- Updates the OS
- Installs the following libraries:
- cloud-init
- open-vm-tools
- docker.io
- nmap
- tcpdump
- dnsutils
- unzip
- jq
- telnet
- zmap
- Sets up Message of the Day (MOTD)
- downloads h3-cli
- downloads Host Utility
- Add current user to docker group

A successful run of the script looks like

```
nodezero@nodezero:~$ ./ubuntu-build.sh

> Disabling journald...

> Disabling firewall...

> Updating OS packages...
> This may take a bit..

> Installing additional tools...
> This may take a bit..

> Downloading Nodezero Banner...

> Checking connection to github.com...

> Updating h3-cli...

> Adding h3-cli to paths

> downloading n0...

> Enabling docker...

> Building NodeZero-CUSTOM
```

10.4 Host Utility

The NodeZero host virtual machine comes with a script for setup and maintenance of the host. To invoke it, type `n0` into the command prompt and a menu like the one below will be presented.

> **Run `n0` with user `nodezero`**
>
> Horizon3.ai recommends running the `n0` script while logged in to the host as user `nodezero`. If you are not logged in as this user, running `n0` will display a warning message.

```
$ n0

0) Version
1) Check environment
2) System info
3) Configure Static IP
4) Configure network proxy
5) Update
6) Set Timezone
7) Simple Network Scan
8) Set Hostname
9) Set NTP server
10) Show Runners
11) Configure MTU
12) Reset Region
q) Exit

Choose an option: _
```

The following sections provide more information on what these options do.

10.4.1 Version Info

Prints the version of `n0` being used

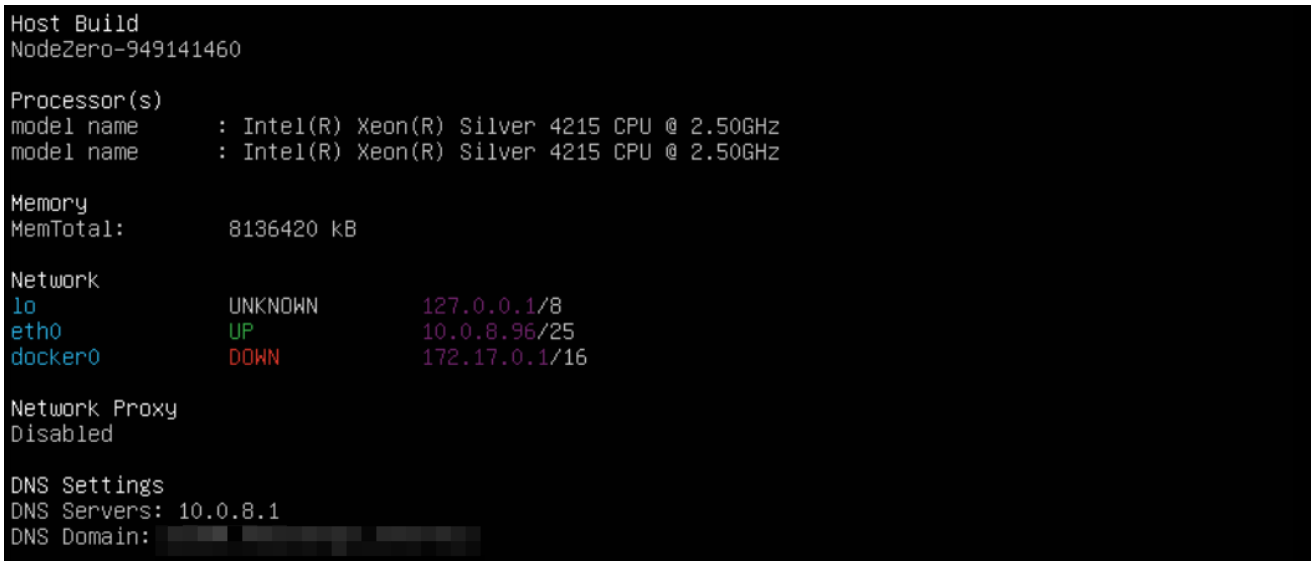
10.4.2 Check Environment

This runs NodeZero's checkenv.sh script to verify the host has the required configuration and settings as well as access externally to the required sites.

10.4.3 System Info

Displays basic system information about the host such as CPUs, Memory and Network settings

Example:



10.4.4 Configure Static IP / Configure DHCP

By default the NodeZero host virtual machine comes with DHCP enabled. The option here will toggle between `Configure Static IP` and `Configure DHCP`.

This defaults to `Configure Static IP` to switch from using DHCP to static ips. Once selected follow the prompts to configure a new IP address, Subnet, Gateway and DNS nameserver

If there is a need to switch back to DHCP, it can be selected with same option number which will show `Configure DHCP`

10.4.5 Configure Network Proxy

To have NodeZero use a proxy for your network provide your proxy setting in the prompt. This command updates values for proxy settings in these files:

/etc/environment

/root/.docker/config.json

/etc/systemd/system/docker.service.d/http-proxy.conf

/etc/docker/daemon.json

> **For Changes To Take Effect**
>
> After setting the proxy, the user must log out and back in before the changes takes effect.

10.4.6 Update

This does three things:

1. Updates the underlying OS, mostly for getting the latest security patches
2. Pulls the latest `h3-cli` while preserving the current configuration
3. Updates the `n0` utility itself.

Ensure Latest Updates

It is suggested to run the update option two times back-to-back to ensure the latest updates are retrieved and implemented

10.4.7 Set Timezone

Allows for setting the timezone on the host.

10.4.8 Simple Network Scan

This runs a simplified network scan using zmap against a the top used ports. This scan is throttled to 150 packets/sec and bandwidth is restricted to 2 Megabits/sec to prevent disrupting the network. This is not to be used to replace running a Segmentation Test which is much more comprehensive in scanning and detecting systems and services.

A complete list of the hosts found is located in `~/.h3/host_list.txt`

10.4.9 Set Hostname

An easy way to change the hostname on the host

10.4.10 Set NTP server

Sets the NTP server in `/etc/systemd/timesyncd.conf`

10.4.11 Show Runners

Prints the list of Runners found on the host using `ps` command

10.4.12 Exit

Exits the `n0` prompt interface

10.5 Validate Host

Once the installation is complete, validate that your NodeZero host is ready to run operations by running the NodeZero environment validation script.

10.5.1 Downloads

Warning

Always verify that the files you download come from Horizon3.

Download validation script

Pick the tab that matches how you are connecting to NodeZero's cloud infrastructure.

US-based + Gateway | EU-based + Gateway | US-based | EU-based | Federal

Environment check script for connectivity and host settings for users connecting with the Gateway and US website.

Download checkenv_ce_us.sh | SHA256 checkenv.sh

Environment check script for connectivity and host settings for users connecting with the Gateway and EU website.

Download checkenv_ce_eu.sh | SHA256 checkenv.sh

Environment check script for connectivity and host settings for users connecting to the US website.

Download checkenv_us.sh | SHA256 checkenv.sh

Environment check script for connectivity and host settings for users connecting to the EU website.

Download checkenv_eu.sh | SHA256 checkenv.sh

Environment check script for connectivity and host settings for users connecting to the Federal infrastructure

Download checkenv_federal.sh | SHA256 checkenv.sh

Plug the file name of your downloaded script version into the run command below.

10.5.2 Use validation script

The validation script ensures that your NodeZero host is properly configured and ready for operations by checking key system and network requirements. The script will complete the following:

- Check for minimum requirements (2 CPUs, 8GB RAM, 40GB+ disk).
- Ensure necessary packages and libraries are installed.
- Confirm compatibility and proper configuration.
- Ensure the host can resolve external domains.
- Check for blocking configurations or restricted access.

Run validation script

To execute the validation script on the OVA host, use a command that matches the script file name you've downloaded. For example, in a basic US environment, with a bash shell, you'd run the following command:

```
curl https://downloads.horizon3ai.com/utilities/checkenv_us.sh | bash
```

In a EU environment with the NodeZero Gateway, you'd run this command:

```
curl https://downloads.horizon3ai.com/utilities/checkenv_ce_eu.sh | bash
```

With any script version, the output should look similar to the following:

```
# curl https://downloads.horizon3ai.com/utilities/checkenv_us.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 18646  100 18646    0     0  61134      0 --:--:-- --:--:-- --:--:-- 61134

[#] Conducting pre-checks to validate the environment is NodeZero ready:

[#] Checking Proxy settings and configurations:
[+] Found 0 of 6 proxy settings in /etc/environment
[+] Found 0 of 6 proxy settings in env
[+] PASSED: Proxy is NOT configured

[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[+] PASSED: Docker is installed and functioning properly.

[#] Checking Docker permissions to volume mount files from /home/nodezero directory:
[+] PASSED: Docker permissions are correct for the /home/nodezero directory location.

[#] Checking Operating System:
[+] PASSED: Linux is a supported Operating System.

[#] Gathering environmental variables to conduct further checks:
[+] PASSED: All environmental variables set and proceeding with next checks.

[#] Checking host time against current UTC time:
[+] PASSED: System time is within 5 minutes of UTC time.

[#] Checking HDD space requirements (30GB Recommended, 20GB Required):
[+] PASSED: There is enough space for the NodeZero container: 79GB

[#] Checking 8GB RAM requirement:
[+] PASSED: This system meets the recommended minimum RAM to support NodeZero.

[#] Checking compute resource requirements:
[+] PASSED: This system has 2 CPUs which meets the minimum logical CPU requirements to run NodeZero.

[#] Pre-check validation completed successfully.
```

After successful validation

After successfully running the validation script, your NodeZero host is ready for operations.

10.5.3 Troubleshooting

NOEXEC flag on partition

Some users report issues running the health check because they are launching from a partition that denies execution. This is why we recommend first changing to your home directory before executing, as in the example above.

Host's system time is out of sync with NodeZero

If you are seeing this error when running the NodeZero launch script or the `checkenv` script

```
The system time is off by more than 5 minutes of UTC time.
```

To address this you'll need to look at your local time syncing service to make sure it is working properly. Possibly sync your service with a known good source from this list:

- https://tf.nist.gov/tf-cgi/servers.cgi
- https://www.ntppool.org/

If your time syncing service is not something that can be easily repaired, you can still run NodeZero. However, certain cryptographic attacks could be affected. To ensure the successful operation of NodeZero, please deactivate any currently active time synchronization service and synchronize your system time with our servers. The below script will stop `timesyncd`, `ntp` and `chrony`, then sync the system time with NodeZero servers via our `/time` API.

```
# stop timesyncd
sudo timedatectl set-ntp false

# stop ntp
sudo service ntp stop

# stop chrony
sudo service chrony stop

# Sets system time
UTC=$(curl -k -s -m 3 https://api.gateway.horizon3ai.com/v1/time | cut -c17-26)
sudo date -s "@$UTC"
```

NodeZero Runners and sudo

It's important to remember that if you're running automated operations with the NodeZero Runner, while the Host Check script has the ability to prompt for `sudo` credentials, you might receive a permissions error:

```
[#] Checking Docker functionality by running the hello-world test container:
[+] PASSED: Docker version installed meets the minimum required version 20.10.
[!] FAILED: Failed to validate Docker. Verify this account has permissions to run Docker and retry.
```

If your NodeZero Host requires `sudo` to run container commands, then you might need to start the Runner using `sudo`.

DOCKER

You can try adding the user that invokes `h3 start-runner` to the `docker` group. For example, with an `ubuntu` user:

```
sudo usermod -aG docker ubuntu
sudo systemctl restart docker
```

> **Logout/login to apply changes**
>
> Be sure to log out and back in after changing groups for the actively logged-on user.

PODMAN

NodeZero always runs Podman with `sudo`. Ensure that the user running the Runner has `sudo` privileges.

10.6 NodeZero App for Splunk

The NodeZero App for Splunk automatically ingests pentest data from the Horizon3.ai NodeZero platform into Splunk. At a configurable polling interval, the app downloads pentest results and indexes them as structured JSON events.

The add-on supports three data types — weaknesses, hosts, and action logs — and maps weakness data to the Splunk CIM Vulnerability data model for use with Splunk Enterprise Security.

You can download the app from Splunkbase.

Version note

This page documents **v3.0.0** of the NodeZero App for Splunk. If you are running v1.x or v2.x, see Version 2.x and Earlier below.

10.6.1 Requirements

- Splunk Enterprise or Splunk Cloud 9.2 or later
- Python 3.9 or later (bundled with Splunk)

10.6.2 Setup

Step 1: Get an API Key

Generate an API key from the NodeZero Portal at portal.horizon3ai.com via **Settings > API Keys**. The API key grants read-only access to pentest results for your organization.

Step 2: Configure an Account

In the Splunk app, navigate to the **Configuration** tab and click **Add** on the **Accounts** sub-tab.

- **Name** (required) — A short identifier for this account. Alphanumeric characters and underscores only.
- **Description** (optional) — Notes about this API key (e.g., which environment it belongs to).
- **API Key** (required) — Your NodeZero API key. Stored encrypted by Splunk's credential manager.
- **API URL** (optional, v3.0.0+) — NodeZero API hostname. Defaults to `api.horizon3ai.com` (US commercial and Federal). Set to `api.horizon3ai.eu` for EU instances. Do not include the `https://` prefix.

Step 3: Create an Input

Navigate to the **Inputs** tab and click **Create New Input**.

- **Name** (required) — A short identifier for this input. Alphanumeric characters and underscores only.
- **Description** (optional) — Notes about the input.
- **API Account** (required) — Select the account configured in Step 2.
- **Polling Interval** (required) — How often to poll for new data, in seconds. Default: `86400` (24 hours). Minimum: `3000` (50 minutes).
- **Pull Pentests After** (optional, v3.0.0+) — Only ingest pentests scheduled after this date (YYYY-MM-DD format). Leave empty to pull pentests from the last 90 days.
- **Index** (required) — The Splunk index to write events to.
- **Pull Hosts** (optional, v3.0.0+) — Ingest host data from pentests. Enabled by default.
- **Pull Weaknesses** (optional, v3.0.0+) — Ingest weakness data from pentests. Enabled by default.
- **Pull Action Logs** (optional, v3.0.0+) — Ingest action log data from pentests. Enabled by default.

The input begins pulling data immediately after you save it.

Index Macro

The sample dashboard relies on a macro called `h3_index`. The default value is `()`, which searches the default index (usually `main`). If you are sending events to a different index, update the macro via **Settings > Advanced Search > Search Macros** to match your index (e.g., `(index=my_nodezero_index)`).

10.6.3 What Gets Indexed

The add-on creates events under three sourcetypes. In v3.0.0, all events are JSON-formatted with automatic field extraction.

Weaknesses (`h3:nodezero:api:weakness_export_csv`) — Vulnerabilities and misconfigurations discovered during pentests. Key fields: `Name`, `Severity`, `ContextScore`, `IP`, `Hostname`, `RootCause`, `PortalUrl`.

Hosts (`h3:nodezero:api:host_export_csv`) — Hosts discovered during pentests, including network attributes and weakness summaries. Key fields: `IP`, `Hostname`, `OS`, `InScope`, `NumWeaknesses`.

Action Logs (`h3:nodezero:api:action_logs_export_csv`) — Step-by-step records of what NodeZero did during each pentest. This is the most granular data type and typically the largest by volume. Key fields: `ModuleName`, `Cmd`, `StartTime`, `IP`.

> ⚠ **Sourcetype change in v3.0.0**
>
> The action logs' sourcetype changed from `h3:nodezero:api:action_logs` (v1.x/v2.x) to `h3:nodezero:api:action_logs_export_csv` (v3.0.0). Host and weakness sourcetypes are unchanged across all versions.

Verify Data Is Being Indexed

```
`h3_index` | stats values(sourcetype) as st
```

You should see three sourcetypes:

```
h3:nodezero:api:action_logs_export_csv
h3:nodezero:api:host_export_csv
h3:nodezero:api:weakness_export_csv
```

CIM Compatibility

Weakness events are mapped to the Splunk **Vulnerability** data model via field aliases and eventtypes, enabling out-of-the-box correlation with other security tools in Splunk Enterprise Security.

10.6.4 Dashboards

The app includes a **NodeZero Operations** dashboard for viewing summary information across all pentests visible in Splunk, with drill-down into specific weaknesses and host-level action details.

In v3.0.0, a monitoring dashboard is also available at `/app/nodezero/dashboard` for tracking ingestion volume, event counts, and errors.

10.6.5 Troubleshooting

v3.0.0

Search the add-on's diagnostic logs:

```
index=_internal sourcetype="nodezero:log"
```

For broader troubleshooting including Splunk's own process logs:

```
index=_internal (sourcetype=splunkd nodezero component=ExecProcessor) OR sourcetype="nodezero:log"
```

Set the log level to DEBUG for more detail: go to the app's **Configuration** tab, select the **Log Level** sub-tab, and change the level. Remember to set it back to INFO after troubleshooting.

v2.x and Earlier

```
index=_internal (sourcetype=splunkd nodezero component=ExecProcessor) OR sourcetype="nodezero-*"
```

There is also a non-navigable dashboard at `/en-US/app/nodezero/kvstore_state` that shows the KVStore state for troubleshooting.

10.6.6 Upgrading from v1.x / v2.x

Version 3.0.0 includes breaking changes:

- **Action logs sourcetype changed** from `h3:nodezero:api:action_logs` to `h3:nodezero:api:action_logs_export_csv`. Update any saved searches, dashboards, or alerts that reference the old name.
- Host and weakness sourcetypes are **unchanged**.
- Host and weakness events are now **JSON instead of CSV**. Field extraction works automatically without custom transforms.
- **API URL is now configurable** — if you use a non-US NodeZero instance, update your account configuration after upgrading.

10.6.7 Version 2.x and Earlier

If you are running v2.x or v1.x of the app, the key differences are:

- **Account setup** has only Name, Description, and API Key fields (no API URL).
- **Input setup** has only Name, Description, API Account, Index, and Polling Interval fields (no date filter or data type toggles).
- **Action logs' sourcetype** is `h3:nodezero:api:action_logs` (without the `_export_csv` suffix).
- **Host and weakness data** is CSV-formatted (not JSON).
- **Troubleshooting logs** use `sourcetype="nodezero-*"`.
- A KVStore debug dashboard is available at `/en-US/app/nodezero/kvstore_state`.

We recommend upgrading to v3.0.0 for significantly faster data ingestion, EU/FedRAMP region support, and improved reliability.

10.6.8 Support

Email splunk@horizon3.ai with any questions, comments, or concerns.

10.7 Verify Download Authenticity

To ensure the integrity and authenticity of your files, always verify that downloads originate from **Horizon3.ai**.

Every download distributed here includes:

- A **download file** (the actual tool or resource).
- A **checksum file** containing an **SHA256 hash** to confirm the file's authenticity and integrity.

Given the nature of cybersecurity, we strongly recommend verifying any file claiming to come from Horizon3.ai against the official checksums posted on this site.

10.7.1 Choose Your Operating System

Select a tab below to view step-by-step instructions.

Linux MacOS Windows Windows (manual)

Linux

Ensure the **downloaded image file** and the **sha256sum checksum file** are in the same directory.

Run the following command in the directory:

```
sha256sum -c *.sha256.checksum
```

Sample output:

```
nodezero@nodezero:~$ sha256sum -c *.sha256.checksum
NodeZero-1717717928.ova: OK
checkenv.sh: OK
nodezero@nodezero:~$
```

Note

Most Linux distributions come with a `sha256sum` command line utility that reads a checksum file, computes the checksum against the file it represents, and returns `OK` if they match.

MacOS

Ensure the **downloaded image file** and the **sha256sum checksum file** are in the same directory.

Run the following command:

```
shasum -c *.sha256.checksum
```

Sample output:

```
horizon3@mac:~$ shasum -c *.sha256.checksum
NodeZero-1717717928.ova: OK
horizon3@mac:~$
```

Note

Most MacOS systems come with a utility called `shasum` that is a command line utility that reads a checksum file, computes the checksum against the file it represents, and returns `OK` if they match.

Windows

Ensure the **downloaded image file** and the **sha256sum checksum file** are in the same directory.

Run the following command:

```
Get-Content -Path .\*.sha256.checksum | % {$Hash, $_, $File = $_.Split(" ", 3); if ($Hash) { [PSCustomObject]@{Path=$File;
Result=if ((Get-FileHash -Path $File -Algorithm SHA256).Hash -eq $Hash) { "OK" } else { "FAILED" }}}} | Format-List
```

Sample output:

```
PS C:\Users\horizon3\Downloads> Get-Content -Path .\*.sha256.checksum | % {$Hash, $_, $File = $_.Split(" ", 3); if ($Hash) {
[PSCustomObject]@{Path=$File; Result=if ((Get-FileHash -Path $File -Algorithm SHA256).Hash -eq $Hash) { "OK" } else {
"FAILED" }}}} | Format-List


Path   : checkenv.sh
Result : OK

Path   : NodeZero-1717717928.ova
Result : OK

PS C:\Users\horizon3\Downloads>
```

Note

Windows Powershell has their own checksum utility.

11. Knowledge Base

11.1 Knowledge Base

Learn the secrets to NodeZero and expanding your security footprint.

Helpful resources pertaining to Horizon3.ai products and services:

NodeZero Modules

Review specific NodeZero functionality to understand how each module works within your security posture.

Glossary

Definition of terms used within NodeZero to help you better understand the terminology and concepts.

Sensitive Data

Learn how sensitive data is handled and displayed within NodeZero, ensuring compliance and security.

Weaknesses

Reference for weaknesses identified by NodeZero and how to address them to strengthen your security posture.

Exposure Score

Description of Exposure Score levels, helping you understand the potential risk and attack surface in your environment.

Attack Configurations

Learn how to configure attack scenarios tailored to your test, ensuring a focused and effective penetration test strategy.

11.2 NodeZero Modules

11.2.1 Cyanide

Cyanide

Cyanide is H3's tool to facilitate and correlate man-in-the-middle (MITM) attacks and credential relays.

Cyanide utilizes opportunistic network protocol poisoning techniques and active coercion techniques to solicit a device to connect to NodeZero with authentication material. If MITM relay has been enabled for the pentest, Cyanide will attempt to relay authentication material to vulnerable target services and applications within the scope of the pentest.

Terminology

MITM Credential Relay attacks typically involve two hosts within the target network. H3 uses the terms "Source" and "Target" to differentiate these two hosts and their associated weaknesses and misconfigurations that enable a successful MITM attack.

- Source - The host/user that initiates a connection and authentication session to NodeZero's relay server. Typically, the source authentication material will provide a username and possibly the path to a requested resource (e.g SMB share, SQL database, etc.)
- Source Weakness - the poisoning or coercion weakness that enabled or caused the source host to connect to NodeZero's relay server.
- Target - The host to which NodeZero will relay the authentication material for exploitation.
- Target Weakness - the Weakness on the target host that enables cyanide to relay credentials successfully and gain unauthorized access.

Purpose

During a pentest, NodeZero has to be able to accurately correlate the source of discovered credentials and track where they are being utilized to access network resources. Additionally, NodeZero needs to accurately determine which weaknesses were utilized in an attack chain. Cyanide's primary purpose is to make these correlations for MITM attacks, and capture authentication material for use or hash-cracking. Cyanide answers the questions:

Authentication Event Details

Category	Description
Who	The user, machine, or service account that the captured authentication material represents.
What	The attack method used to force the source host/user to connect to NodeZero: - **Poisoning:** LLMNR, NBT-NS, MDNS - **Coercion:** PetitPotam, ShadowCoerce, PrinterBug, etc. Additionally, the resource requested by the source user (e.g., SMB share, SQL database, etc.).
Where	The source host of the authentication material (i.e., where the connection originated). Where the credentials were utilized (i.e., the target service/host of the relay attack).
When	The timestamp of each event, including: - When the host was poisoned - When the hash was captured - When a relay attack occurred
Why	With the combined data, Cyanide can determine why the attack happened and why it was successful.

System Breakdown

Currently the Cyanide system consists of 4 distinct parts:

Attack Components

#	Component	Description
1	**Cyanide Message Pump**	Correlates source and target information and provides collected data to NodeZero.
2	**Responder**	Handles broadcast protocol poisoning.
3	**Intimidator**	Executes coercion attacks.
4	**Impacket's ntlmrelayx**	Manages inbound SMB and HTTP connections and relays authentication material to vulnerable targets.

1. Cyanide Message Pump

The main cyanide process, or message pump, processes incoming messages from the other 3 components of the Cyanide system and takes appropriate action correlating source and target information and populating a database that NodeZero can utilize to understand what MITM interactions are occurring and what new authentication material is available for the pentest.

---

2. Responder

Responder is an LLMNR, NBT-NS, and MDNS poisoner. It will answer specific NBT-NS (NetBIOS Name Service) queries based on their name suffix (see: Archived Microsoft KB). By default, the tool will only answer File Server Service requests, which is for SMB.

If Responder poisons a source host via one of these broadcast protocols, it will reach back via whatever protocol the broadcast was for (e.g. SMB, RDP, MSSQL, etc.).

- If the protocol is NOT SMB or HTTP, Responder will simply capture the credential and inform Cyanide of the results. Cyanide will then report this data back to NodeZero's ephemeral cloud architecture and attempt to re-use or crack the captured credential material.
- If the protocol IS SMB or HTTP, ntlmrelayx SMB and HTTP servers will handle it appropriately - either relaying it to a vulnerable service or dumping the credential for cracking.

Poisoners

Name	Description	Protocol	Port(s)
LLMNR	**Link-Local Multicast Name Resolution (LLMNR)** Enables local network computers to resolve hostnames using DNS-like requests. It uses unicast, so only the requesting device sees the reply.	UDP	5355
NBT-NS	**Network Basic Input/Output System Name Service (NBT-NS)** Used for name resolution with NetBIOS names. It can use broadcast, unicast, or multicast.	UDP	137, 138
mDNS	**Multicast Domain Naming System (mDNS)** Allows hostname resolution without a central DNS server. Replies are multicast, updating local caches.	UDP	5353
SERVERS			
MSSQL	Microsoft SQL Server	TCP, UDP	1433 (TCP) 1434 (UDP)
RDP	Remote Desktop Protocol	TCP	3389
Kerberos	Authentication protocol used for secure network login	TCP	88
FTP	File Transfer Protocol	TCP	21
POP	Post Office Protocol (used for retrieving emails)	TCP	110
SMTP	Simple Mail Transfer Protocol (used for sending emails)	TCP	25, 587
IMAP	Internet Message Access Protocol (used for retrieving emails)	TCP	143
HTTPS	Secure HTTP communication	TCP	443
LDAP	Lightweight Directory Access Protocol	TCP, UDP	389
DCERPC	Distributed Computing Environment Remote Procedure Call	TCP	135
WINRM	Windows Remote Management	TCP	5895

Expanded vs. Limited Poisoning

NodeZero's Attack Configuration options have 2 options that control the behavior of Responder:

- Expanded LLMNR and NetBIOS poisoning
- Responder will sniff all available traffic regardless of scope.
- In "Expanded" mode, relay sources from outside the pentest's configured scope will NOT be targeted for any other attacks; they are only used to capture/relay credential material.
- Limited LLMNR and NetBIOS poisoning
- Responder is limited to the scope provided during the configuration of the pentest. If both options are selected, the Limited option will override the Expanded option.

Where does it work?

Since Responder works by capturing broadcast and multicast packets, capturing requests in different networks is not possible and therefore, Cyanide will only work within NodeZero's subnet.

3. Intimidator

Intimidator is H3's framework for integrating NTLM coercion techniques with Cyanide. Intimidator provides a quick plug-and-play capability to facilitate the inclusion of new coercion techniques and open source tools quickly into NodeZero. Cyanide communicates with Intimidator over a duplexed IPC -- allowing the two processes to coordinate coercion and relay attacks effectively.

---

4. Impacket's NTLMRelayx

Cyanide utilize's a modified version of Impacket's NTLMRelayx as the base for our relay server.

When a source host connects and provides authentication material to ntlmrelayx's SMB or HTTP server, it will save the NTLMv2 hash for cracking and relay the authentication session to high-value service vulnerable to NTLM relay within the scope of the pentest. Possible targets include: - SMB servers with SMB-signing disabled: If cyanide is able to successfully log into the server, it will attempt to dump local credentials. - ADCS Server with the ESC8 Misconfiguration - LDAP servers with LDAP Signing disabled.

SCOPING SCENARIOS

The below Scenarios and Examples review cyanide's behavior when the "Limited LLMNR and NetBIOS poisoning option is configured for the pentest.

Scenario 1

No scope is specified OR if the scope of the NodeZero host subnet is specified Scope defaults to the full subnet of the NodeZero host to poison

Example 1

```
NodeZero host subnet: 192.168.0.0/24
Scope: Auto-Expand
```

Result: Cyanide will get a scope of 192.168.0.0/24 because no scope was specified and poisoning can only happen within the network of NodeZero. Relaying will occur against high-value targets that are discovered. Coercion attempts will be made against high-value targets within the pentest scope. Example 2

```
NodeZero host subnet: 192.168.0.0/24
Scope: 172.16.100.0/24, 10.0.0.0/16, 192.168.0.0/24
```

Result: Cyanide will get a scope of 192.168.0.0/24 because the specified scope contains the subnet of the NodeZero host. Relaying and coercion will occur against high-value targets that are within the scope specified.

Scenario 2

The scope of the NodeZero host is within the whitelist, Cyanide will get that as its scope

Example

```
NodeZero host subnet: 192.168.0.0/24
Scope: 172.16.100.0/24, 10.0.0.0/16, **192.168.0.0/30**
```

Result: Cyanide will only poison hosts within the 192.168.0.0**/30** subnet because it falls within the NodeZero hosts subnet. Relaying and coercion will occur against high-value targets that are within the scope specified.

11.3 Glossary

11.3.1 Glossary

NodeZero is heavily involved with the cybersecurity community. As such, we use some industry jargon in our product and documentation. This is where that jargon gets demystified.

Welcome to our product glossary, a quick reference for essential cybersecurity terms woven into our solution. Whether you're new to cybersecurity or a seasoned professional, this resource will help you understand the specific language used in our product. It's organized alphabetically. If you find that something is missing, please provide feedback at the bottom of the page!

Assets

An asset is a networked entity (i.e., system host/machine) that has an IP address and is discovered during a scan.

Examples of assets include:

- Servers
- Laptops, desktops, and mobile devices
- Virtual machines
- Printers
- IoT devices
- Switches, routers, access points, endpoints
- Public static IPs
- Lambdas and serverless architecture
- Kubernetes pods and other containerized applications

If an IP address is dynamically assigned, resulting in changing IPs on the same asset, then an algorithm will detect the ephemeral nature and remove the duplicates on that asset.

Attack Path

An attack path refers to the sequence of steps or actions an attacker may take to compromise a system or network. It involves identifying vulnerabilities and other weaknesses, exploiting them, and navigating through the network to access valuable information or resources.

BloodHound Data

BloodHound data is the reconnaissance information collected and analyzed by the BloodHound tool within an Active Directory and/or Azure environment. NodeZero users have the option to obtain the BloodHound data collected during a pentest.

Impact

Impacts summarize, in business terms, the effects NodeZero was able to achieve as a result of exploiting weaknesses in your environment.

N-Day

An N-day is a software or hardware vulnerability that is already publicly known (N days since disclosure), but there might or might not be a security update available to remediate the vulnerability.

NodeZero Runner

The NodeZero Runner enables automated deployment of a NodeZero container. This enables you to provision and deploy pentests from the NodeZero Portal, without having to manually run a NodeZero launch script.

Notable Event

A feature of Real-Time View (RTV). These events signify that during the pentest, NodeZero performed actions that would likely lead to a critical Impact.

RAT

RAT stands for remote access tool – software that gives a person full control of a tech device, remotely. These tools have legitimate uses, such as technical support, but can also be controlled by attackers with malicious intent. In the context of NodeZero, a RAT is used to provide NodeZero with additional access to further explore attack paths during operations.

Real-Time View (RTV)

Real-Time View in NodeZero provides you with real-time information and updates on the progress of your running pentest, including status updates for injected credentials.

Sensitive Data Exposure

An Impact that indicates that NodeZero was able to **potentially** access sensitive information, given the filetype or service that is compromised. Examples include, but are not limited to:

- Business documents in file shares (.docx, .pdf, .xlsx)
- Outlook PST files
- Confluence RCE
- Exchange RCE

States (Pentest)

The pentest lifecycle encompasses several stages, each with its unique purpose and characteristics. These stages surface in the NodeZero Portal with words like: "Preparing," "Action Needed," "Running," "Processing," and "Done." Below, you will find a detailed breakdown of these stages and their respective descriptions.

State Name	Description
Action Needed	Copy and run the one-time command on your NodeZero Host to launch NodeZero.
Action Needed: Paused	Start your pentest after adding NodeZero's IP to your allowlist.
Done	The pentest is fully complete and results are available in the Portal.
Ended	The pentest was ended early by the user. Results are available in the Portal.
Error	An error was encountered during this pentest.
Paused	The pentest is paused.
Pausing	The pentest is in the process of pausing.
Preparing	The pentest is setting up the resources it needs.
Preparing (Start Paused)	Resources are being provisioned for the pentest. Once provisioning is complete, the pentest will start in a paused state.
Processing	The pentest results are being processed.
Resuming	The pentest is resuming from a paused state.
Running	The pentest is live and running.
Scheduled	The pentest has been scheduled and will begin provisioning resources shortly.
Unknown	The pentest encountered an issue. Contact us for further assistance if this issue persists.

Weakness

A weakness refers to a vulnerability or security flaw that can be exploited by an attacker to compromise a system or network. Weaknesses can include misconfigurations, outdated software, default credentials, or other vulnerabilities that can be leveraged to gain unauthorized access or perform malicious actions.

11.4 Sensitive Data

11.4.1 Sensitive Data Exposure

NodeZero attempts to discover and assess **potentially** sensitive information when a filesystem or service is compromised. Examples include, but are not limited to:

- Business documents in file shares (`.docx`, `.pdf`, `.xlsx`).
- Outlook PST files.
- Confluence RCE.
- Exchange RCE.

When NodeZero gains access to files, it takes different actions depending on the file type. For example, if it finds SSH key or `.aws/credentials` files, NodeZero attempts to extract those credentials for use in the operation. When it discovers business documents, NodeZero first extracts the text from the documents, and then uses a natural-language processing engine and contextual awareness to look for personally identifiable information (PII) and protected health information (PHI).

Sensitive Data Types

NodeZero currently supports the following protected data types:

- Social Security numbers (SSNs).
- Credit card numbers.
- US bank numbers.
- US individual Tax Identification Numbers (TINs).
- US passport numbers.
- ABA (American Bankers Association) routing numbers

Protection Against Exposure

To protect against accidental exposure of PII or PHI, all of the text extraction, natural-language processing, and contextual analysis happen inside the NodeZero container, which resides in the customer's network on the customer's Docker host. Once a file has been analyzed, NodeZero deletes it from the container. The only information retained is metadata about the exposure. That metadata includes the filename, the file share on which it was located, and the type of data discovered (e.g., credit card number).

EPHEMERAL FILE ANALYSIS IN ACTION

As an example, let's say NodeZero discovers a file called `customer_invoice.pdf` on the `C$` share of host `192.168.1.35`. The file's discovery will cause NodeZero to execute its sensitive-data routines. NodeZero copies the file from the share to the NodeZero container, which is running on the Docker host in the customer's network.

NodeZero extracts the text from the `.pdf`, and processes it using pattern matching, natural-language processing and contextual awareness. This leads to discovering a credit card number. NodeZero deletes the file from the container, and reports back that it found sensitive data of type `CREDIT_CARD` in the file `customer_invoice.pdf` on host `192.168.1.35` in the `C$` share. The actual credit card number is not extracted or used, beyond this initial classification.

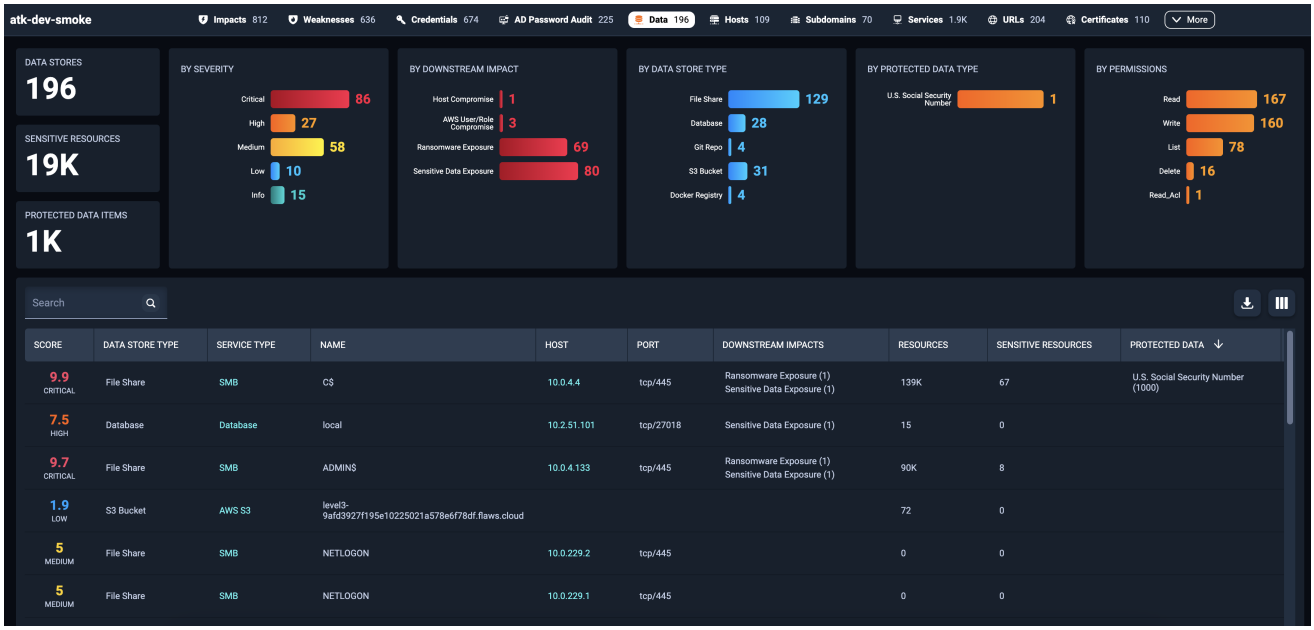
Sampling Limits

For efficiency, NodeZero will not look at every file exposed in a share. It obliges limitations on the number of files, and the size of the files, reviewed. In addition, if all other actions are completed, NodeZero will end the operation

even if there are remaining files to process. This is done to prevent prolonged operations when large quantities of files are exposed.

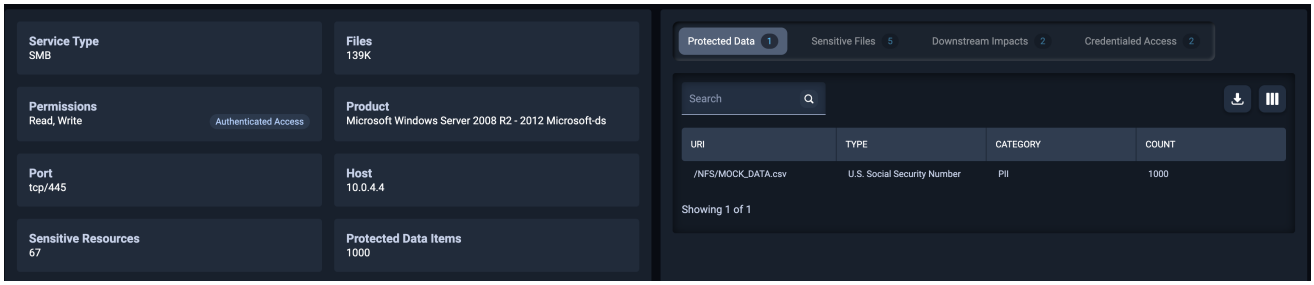
Sensitive Data Exposure Results in the Portal

When NodeZero discovers sensitive data exposures, it documents them in the Portal under the `Data` tab. This tab enables you to browse through exposed data store locations. Records display the data store type, service type, data store name, host location, downstream impacts, number of sensitive resources exposed, and types of protected data exposed (PII/PHI).



EXPOSURE DETAILS

To dig deeper, click on any row. This will display details and examples of the sensitive files, along with the patterns that they matched, downstream impacts, credentials used to gain access, and protected data exposures if any exist. You can use this data to trace back to the origin of the sensitive data exposure. NodeZero cannot tell you what data was exposed because, as outlined above, it does not extract or store the actual contents.



11.5 Weaknesses

11.5.1 Weaknesses

NodeZero identifies and surfaces many weaknesses that it finds during a pentest. Within test results, these weaknesses are designated either with a CVE identifier (e.g., `CVE-2021-44228`) for publicly identified Common Vulnerabilities and Exposures, or with an H3 weakness identifier (e.g., `H3-2022-0001`) for weaknesses discovered through Horizon3.ai original research.

This page provides a reference for H3 (Horizon3.ai-discovered) weaknesses. These weaknesses might appear in your pentest reports up to 90 days before we publicly list them on this site. (For details, please see our Vulnerability Disclosure Policy). Once these weaknesses are publicly disclosed and assigned a CVE ID, we add that CVE in pentest results.

For details on previously disclosed CVEs that NodeZero finds in tests, please search the official CVE website maintained by the MITRE Corporation.

Weakness ID	Name
H3-2026-0051	Swagger API Specification Exposed to Unauthenticated Users
H3-2026-0050	Entra ID Privileged Role Administrator Can Escalate to Global Administrator
H3-2026-0049	Cloud-Native Entra ID User Has Privileged Directory Role
H3-2026-0048	Authenticated Endpoint Accessible Without Authentication
H3-2026-0046	Ubiquiti UniFi OS Authentication Bypass and Path Traversal Vulnerability
H3-2026-0045	Hardcoded Credentials in Azure Automation Runbook
H3-2026-0044	Azure Container Registry Task Code Execution
H3-2026-0043	Bricks Builder Anonymous Authorization Bypass
H3-2026-0042	WordPress XML-RPC Pingback Server-Side Request Forgery Vulnerability
H3-2026-0041	WordPress User Registration Enabled
H3-2026-0040	WordPress XML-RPC API Methods Exposed
H3-2026-0039	Directory Listing Exposed to Unauthenticated Users
H3-2026-0038	Stack Trace Disclosure to Unauthenticated Users
H3-2026-0037	Exposed Google Maps API Key
H3-2026-0036	Internal System Data Exposure to Unauthenticated Users
H3-2026-0035	Personal Information Disclosure to Unauthenticated Users
H3-2026-0034	Exposed API Key in Web Response
H3-2026-0033	Exposed Credentials in Web Response
H3-2026-0032	Azure App Service SCM/Kudu Basic Authentication Enabled
H3-2026-0031	Azure App Service Remote Debugging Enabled
H3-2026-0028	Apache Tomcat JK Status Manager Exposed
H3-2026-0025	SimpleHelp OIDC Authentication Bypass Vulnerability
H3-2026-0023	Azure Container Registry Admin User Enabled
H3-2026-0022	Azure Container Registry Public Network Access Enabled
H3-2026-0021	Azure Container Registry Anonymous Pull Enabled
H3-2026-0016	Net-SNMP EXTEND-MIB Read/Write Access Remote Code Execution
H3-2026-0015	Azure Storage Account Allows Blob Public Access
H3-2026-0014	Nagios XI SQL Injection Vulnerability
H3-2026-0013	Insecure Direct Object Reference (IDOR) / Broken Object Level Authorization (BOLA)
H3-2026-0012	Fortinet FortiClient EMS Improper Access Control Vulnerability
H3-2026-0011	Reversible Password Encryption Enabled on Domain Controller
H3-2026-0009	Apache Solr AuthTool Hardcoded Credentials Vulnerability
H3-2026-0008	Apache ActiveMQ Jolokia Remote Code Execution Vulnerability
H3-2026-0007	SSH ControlMaster Socket Abuse
H3-2026-0005	Web Application UNC Absolute Path Traversal Vulnerability

Weakness ID	Name
H3-2026-0004	Active Directory Certificate Services Misconfiguration: NTLM Relay to AD CS RPC Endpoint
H3-2026-0003	MSSQL EXECUTE AS Impersonation Privilege Escalation Vulnerability
H3-2026-0002	Kubernetes Nodes Proxy GET Permission Remote Code Execution
H3-2025-0080	Sensitive Information Disclosure to Unauthenticated Users
H3-2025-0074	LDAP Channel Binding Not Required
H3-2025-0073	LDAP Signing Not Required
H3-2025-0068	Privilege Escalation - Potato Style Exploit
H3-2025-0067	LAPS Password Exposure
H3-2025-0066	osTicket PHP Filter Chain Injection Vulnerability
H3-2025-0065	osTicket Anonymous Ticket Creation with Rich Text Content Enabled
H3-2025-0064	osTicket Self-Signup Enabled
H3-2025-0062	SCCM Hierarchy Takeover via NTLM Coercion and Relay to SMB
H3-2025-0060	Gladinet Centrestack MachineKey Deserialization Vulnerability
H3-2025-0058	SCCM Hierarchy Takeover via NTLM Coercion and Relay to MSSQL
H3-2025-0057	N-able N-central Authentication Bypass Vulnerability
H3-2025-0056	FreePBX Authentication Bypass SQL Injection
H3-2025-0055	FreePBX Authentication Bypass File Upload RCE
H3-2025-0054	N-able N-central Authenticated XML External Entity (XXE) Vulnerability
H3-2025-0053	Fortinet FortiSIEM Arbitrary File Write Remote Code Execution Vulnerability
H3-2025-0052	Golden Ticket
H3-2025-0051	UpdraftPlus Plugin PEM Key Exposure
H3-2025-0049	Thinkphp Remote Code Execution Vulnerability
H3-2025-0048	PHP Debug Interface Exposure
H3-2025-0047	Puppet Node Manager Authorization Bypass
H3-2025-0046	Samsung MagicINFO 9 Server Remote Code Execution Vulnerability
H3-2025-0044	Oracle EBS Bispgraph File Access Vulnerability
H3-2025-0034	GoCD Encryption Key Exposure in Pipeline Configuration
H3-2025-0033	Docker Compose File Exposure
H3-2025-0032	Generic .env File Exposure
H3-2025-0028	Unsecured InfluxDB Access via Misconfiguration
H3-2025-0027	Kentico Xperience Staging Service Authentication Bypass WT-2025-0011 Vulnerability
H3-2025-0026	Kentico Xperience Staging Service Authentication Bypass WT-2025-0006 Vulnerability
H3-2025-0025	Langflow Code Injection Vulnerability
H3-2025-0024	Active Directory Misconfiguration: Low-Privilege User with GenericAll Privileges

Weakness ID	Name
H3-2025-0023	Wordpress Newsletter Manager < 1.5 - Unauthenticated Open Redirect
H3-2025-0022	Wordpress DB Repair Exposed
H3-2025-0021	Wordpress Directory Listing
H3-2025-0020	Wordpress Accessible WPConfig
H3-2025-0019	Git Repo-Jacking
H3-2025-0018	GitHub Actions Dangerous Bot Conditions
H3-2025-0017	GitHub Actions Secrets Inheritance
H3-2025-0016	GitHub Actions Cache Poisoning
H3-2025-0015	GitHub Actions Dangerous Environment Variable Writes
H3-2025-0014	GitHub Actions Insecure Commands
H3-2025-0013	GitHub Actions Unpinned Uses
H3-2025-0012	GitHub Actions Template Injection
H3-2025-0011	GitHub Actions Self Hosted Runner
H3-2025-0010	GitHub Actions Ref Confusion
H3-2025-0009	GitHub Actions Known Vulnerable Actions
H3-2025-0008	GitHub Actions Imposter Commit
H3-2025-0007	GitHub Actions Hardcoded Container Credentials
H3-2025-0006	GitHub Actions Excessive Permissions
H3-2025-0005	GitHub Actions Dangerous Triggers
H3-2025-0004	GitHub Actions Artifacts Credential Leakage
H3-2025-0003	IIS Shortname Disclosure Vulnerability
H3-2025-0002	Management Console Exposed to the Internet
H3-2025-0001	SimpleHelp Path Traversal Vulnerability
H3-2024-0061	CyberPower PowerPanel Enterprise SQL Injection Vulnerability
H3-2024-0060	Insecure Storage of Connection Strings in Application Properties
H3-2024-0059	Kubernetes Service Account Can Execute Code in Pods
H3-2024-0058	Improper use of K8S Cluster Admin Access
H3-2024-0057	Active Directory gMSA Account Password Exposure
H3-2024-0056	Microsoft SQL Server NTLM Credential Coercion Vulnerability
H3-2024-0054	Kubernetes Identity can Create a Pod and Escape to the Node
H3-2024-0053	Ivanti Endpoint Manager Multiple NTLM Credential Coercion Vulnerabilities
H3-2024-0052	CUPS-browsed Server Side Request Forgery Vulnerability
H3-2024-0051	Intune Mobile Device Management Remote Code Execution (RCE)
H3-2024-0046	Over-Privileged StackSet Execution Role
H3-2024-0045	AWS Privilege Escalation via iam:PassRole and ec2:RunInstances

Weakness ID	Name
H3-2024-0044	AWS Privilege Escalation via iam:PassRole, lambda:CreateFunction, and lambda:InvokeFunction
H3-2024-0043	AWS Privilege Escalation via iam:PassRole and cloudformation:CreateStack
H3-2024-0042	Palo Alto Expedition Authenticated Sensitive Information Leak Vulnerability
H3-2024-0041	Palo Alto Expedition Unauthenticated SQL Injection Vulnerability
H3-2024-0040	Palo Alto Expedition Authenticated Command Injection Vulnerability
H3-2024-0039	Microsoft Graph App Role Privilege Elevation
H3-2024-0038	Microsoft Entra (AzureAD) - Entra Group Takeover
H3-2024-0037	Azure Cloud Kerberos Trust Abuse
H3-2024-0036	Improper use of AWS Administrator Access
H3-2024-0035	AWS Access Key Id Third Party Canary
H3-2024-0034	NTLM Authentication Endpoint Exposed to the Internet
H3-2024-0033	Jupyter Server on Windows Credential Leak Vulnerability
H3-2024-0032	Traccar Self-Signup Enabled
H3-2024-0031	Gradio Arbitrary File Read Vulnerability
H3-2024-0030	Traccar Device Image Upload Remote Code Execution Vulnerability
H3-2024-0029	Active Directory User has Entra Administrator Role
H3-2024-0028	NodeZero Remote Access Tool Deployed and Executed
H3-2024-0019	Credential Dumping - Office365 Application Memory
H3-2024-0018	Unauthenticated Access to Redis
H3-2024-0017	AWS Privilege Escalation - iam:PutGroupPolicy
H3-2024-0016	AWS Privilege Escalation - iam:AttachGroupPolicy
H3-2024-0015	NextChat Open Proxy Server-Side Request Forgery Vulnerability
H3-2024-0012	Microsoft Entra (AzureAD) - Service Principal Takeover
H3-2024-0011	Microsoft Entra (AzureAD) - Over-Privileged Service Principal
H3-2024-0010	Microsoft Entra (AzureAD) Connect Credential Dumping
H3-2024-0009	AWS Privilege Escalation - iam:CreatePolicyVersion
H3-2024-0008	AWS Privilege Escalation - iam:UpdateAssumeRolePolicy
H3-2024-0007	AWS Privilege Escalation - iam:UpdateLoginProfile
H3-2024-0006	AWS Privilege Escalation - iam:CreateLoginProfile
H3-2024-0005	AWS Privilege Escalation - iam:CreateAccessKey
H3-2024-0004	AWS Privilege Escalation - iam:PutRolePolicy
H3-2024-0003	AWS Privilege Escalation - iam:AttachRolePolicy
H3-2024-0002	AWS Privilege Escalation - iam:PutUserPolicy
H3-2024-0001	AWS Privilege Escalation - iam:AttachUserPolicy

Weakness ID	Name
H3-2023-0030	Active Directory - User Password Not Required
H3-2023-0029	Password in Active Directory User Attribute
H3-2023-0028	H2 Embedded Database Misconfiguration
H3-2023-0027	NextGen Mirth Connect Remote Code Execution Vulnerability
H3-2023-0023	Apache Solr Arbitrary File Read Vulnerability
H3-2023-0022	PaperCut Arbitrary File Read and Deletion Vulnerability
H3-2023-0021	Phished Credential
H3-2023-0020	PaperCut File Upload Remote Code Execution Vulnerability
H3-2023-0019	Credential Dumping - Data Protection API (DPAPI) Secrets
H3-2023-0018	Microsoft Windows Machine Account NTLM Coercion via Distributed File System Namespace Management Protocol Manipulation
H3-2023-0017	Microsoft Windows Machine Account NTLM Coercion via File Server Remote VSS Protocol Manipulation
H3-2023-0016	Authenticated Microsoft Windows Machine Account NTLM Coercion via Print Spooler Protocol Manipulation
H3-2023-0015	Authenticated Microsoft Windows Machine Account NTLM Coercion via EventLog Remoting Protocol Manipulation
H3-2023-0014	Authenticated Microsoft Windows Machine Account NTLM Coercion via Distributed File System Namespace Management Protocol Manipulation
H3-2023-0013	Authenticated Microsoft Windows Machine Account NTLM Coercion via File Server Remote VSS Protocol Manipulation
H3-2023-0012	Microsoft Windows Machine Account NTLM Coercion via Print Spooler Protocol Manipulation
H3-2023-0011	Microsoft Windows Machine Account NTLM Coercion via EventLog Remoting Protocol Manipulation
H3-2023-0010	Kerberos Constrained Delegation
H3-2023-0009	Kerberos Unconstrained Delegation
H3-2023-0008	AWS Multi-Factor Authentication Disabled
H3-2023-0007	Detection: 3CXDesktopApp Supply Chain Attack
H3-2023-0006	Detection: Pass-The-Hash
H3-2023-0005	Detection: A Hidden Machine Account Was Created
H3-2023-0004	Detection: Kerberoasting
H3-2023-0003	Pre-Windows 2000 Computer Set
H3-2023-0002	Flask Authentication Bypass Misconfiguration
H3-2023-0001	Apache Superset Authentication Bypass Misconfiguration
H3-2022-0097	Kerberos Pass-the-Ticket Attack
H3-2022-0096	Lexmark Printer Command Injection Vulnerability
H3-2022-0095	Password Reuse Found in Active Directory Services Database (NTDS)

Weakness ID	Name
H3-2022-0094	Kubernetes Read with Service Account Token
H3-2022-0093	Weak or Default Credentials - Cracked Credentials from Active Directory Services Database (NTDS)
H3-2022-0092	Kubernetes Remote Code Execution with Service Token
H3-2022-0091	Credential Dumping - Chrome Browser
H3-2022-0090	Public Access to Amazon RDS Snapshot
H3-2022-0089	Public Access to Amazon EBS Snapshot
H3-2022-0088	Public Access to Amazon EC2 AMI
H3-2022-0087	Password Reuse
H3-2022-0086	Domain User with Local Administrator Privileges
H3-2022-0085	Credential Reuse - Shared Windows Local User and Domain User Accounts
H3-2022-0084	Credential Reuse - Windows Local Administrator Accounts
H3-2022-0083	Anonymous Access to the Kubernetes Dashboard
H3-2022-0082	Exposed Kubernetes Version
H3-2022-0081	Atlassian Jira Unauthenticated User Enumeration via the User Picker Browser
H3-2022-0080	WordPress Unauthenticated User Enumeration
H3-2022-0079	Credential Dumping - AWS Instance Metadata Service v2
H3-2022-0078	Unauthenticated Gitlab User Enumeration
H3-2022-0077	Amazon Relational Database Service (RDS) DB Instance Is Exposed to the Internet
H3-2022-0076	Unauthenticated AWS Cognito Role
H3-2022-0075	Public-Facing Application Exposed with HTTP Basic Authentication
H3-2022-0074	AWS Assume Role Access
H3-2022-0073	Microsoft Windows Machine Account NTLM Coercion via Authenticated LSARPC Spoofing
H3-2022-0072	Apache Airflow Debug Mode Enabled
H3-2022-0071	Jenkins Self-Signup Enabled
H3-2022-0070	Anonymous MongoDB Access
H3-2022-0069	Web Directory Listing
H3-2022-0068	Airflow Configuration Exposure
H3-2022-0067	Weak or Default Credentials - MongoDB
H3-2022-0066	Git Repo Exposed on a Web Server
H3-2022-0065	Unauthenticated Access to Apache Airflow
H3-2022-0064	Rails Secret Token Exposure
H3-2022-0063	Private Keys Exposed on Web Server
H3-2022-0062	Microsoft FrontPage service.pwd File Exposure
H3-2022-0061	Apache Web Server htpasswd File Exposure

Weakness ID	Name
H3-2022-0060	Spring Boot Env Actuator Exposed
H3-2022-0059	Spring Boot Configuration Properties Actuator Exposed
H3-2022-0058	Jolokia Local File Inclusion Misconfiguration
H3-2022-0057	jQuery File Upload Widget Exposed
H3-2022-0056	Anonymous Deployment Privileges in JFrog Artifactory
H3-2022-0055	phpMyAdmin Setup Page Exposed
H3-2022-0054	CGI Test Script Exposed
H3-2022-0053	Laravel .env File Exposure
H3-2022-0052	Ansible Configuration File Exposure
H3-2022-0051	Symfony Configuration File Exposure
H3-2022-0050	PHP-FPM Configuration File Exposure
H3-2022-0049	IIS web.config File Exposure
H3-2022-0048	Apache Web Server Configuration File Exposure
H3-2022-0047	Apache Tomcat Example Scripts Exposed
H3-2022-0046	Rails Database Configuration File Exposure
H3-2022-0045	PHPinfo Page Exposed
H3-2022-0044	Shell History File Exposure
H3-2022-0043	Backup File Exposure
H3-2022-0042	Django Debug Mode Enabled
H3-2022-0041	Symfony Profiler Enabled
H3-2022-0040	Symfony Debug Mode Enabled
H3-2022-0039	Golang pprof Debugging Endpoint Enabled
H3-2022-0038	Ruby on Rails Debug Mode Enabled
H3-2022-0037	Laravel Debug Mode Enabled
H3-2022-0036	Guest Access to Zabbix Dashboards
H3-2022-0035	Unauthenticated Access to JavaMelody Monitoring Console
H3-2022-0034	Anonymous Access to Zoho ManageEngine ADManager Plus Employee Search
H3-2022-0033	Unauthenticated Access to Jenkins People Directory
H3-2022-0032	Unauthenticated Access to Prometheus Alertmanager
H3-2022-0031	Unauthenticated Access to Mongo Express
H3-2022-0030	Unauthenticated Access to Paessler PRTG Network Monitor
H3-2022-0029	Unauthenticated Access to ThoughtWorks GoCD
H3-2022-0028	Unauthenticated Access to Apache Solr
H3-2022-0027	Unauthenticated Access to Jupyter
H3-2022-0026	Unauthenticated Access to Kubeflow

Weakness ID	Name
H3-2022-0025	Unauthenticated Access to Kibana
H3-2022-0024	Active Directory Certificate Services Misconfiguration: NTLM Relay to AD CS HTTP Endpoint
H3-2022-0023	Active Directory Certificate Services: Vulnerable Certificate Authority Access Control
H3-2022-0022	Active Directory Certificate Services - EDITF_ATTRIBUTESUBJECTALTNAME2 flag set
H3-2022-0021	Active Directory Certificate Services Domain Escalation via Vulnerable PKI AD Object Access Controls
H3-2022-0020	Active Directory Certificate Services Misconfigured Template Access Controls
H3-2022-0019	Active Directory Certificate Services - Template May Be Requested by Enrollment Agent Signature
H3-2022-0018	Active Directory Certificate Services Misconfigured Enrollment Agent Template
H3-2022-0017	Active Directory Certificate Services Misconfiguration Privilege Escalation - Any Purpose or No (aka SubCA) EKU Misconfiguration
H3-2022-0016	Active Directory Certificate Services Misconfiguration Privilege Escalation - Subject Alternative Name
H3-2022-0015	Web Application Path Traversal Vulnerability
H3-2022-0013	Unauthenticated Access to Jira Projects
H3-2022-0012	Unauthenticated Access to Jira Dashboards
H3-2022-0011	Zoho ManageEngine ADAudit Plus Remote Code Execution Vulnerability
H3-2022-0010	Risky Port Exposed to the Internet
H3-2022-0009	Simple Network Management Protocol (SNMP) Port Exposed to the Internet
H3-2022-0008	File Transfer Protocol (FTP) Port Exposed to the Internet
H3-2022-0007	Telnet Port Exposed to the Internet
H3-2022-0006	Database Port Exposed to the Internet
H3-2022-0005	Secure Socket Shell (SSH) Port Exposed to the Internet
H3-2022-0004	Server Message Block (SMB) Port Exposed to the Internet
H3-2022-0003	Remote Desktop Protocol (RDP) Port Exposed to the Internet
H3-2022-0002	Azure Multi-Factor Authentication Disabled
H3-2022-0001	Web Application Cross Site Scripting Vulnerability
H3-2021-0048	Open Mail Relay
H3-2021-0047	JBoss Application Server HTTP Invoker Remote Code Execution Vulnerability
H3-2021-0046	Credential Dumping - Active Directory Services Database (NTDS)
H3-2021-0045	Credential Dumping - /etc/shadow File
H3-2021-0044	Credential Dumping - Local Security Authority Subsystem Service (LSASS) Memory
H3-2021-0043	Credential Dumping - Local Security Authority (LSA) Secrets
H3-2021-0042	Credential Dumping - Security Account Manager (SAM) Database
H3-2021-0041	Apache Druid Server-Side Request Forgery Vulnerability

Weakness ID	Name
H3-2021-0040	AWS Instance Metadata Service v1 Exposed
H3-2021-0039	Unrestricted Sudo Privileges
H3-2021-0038	Kerberoasting
H3-2021-0037	Werkzeug Debug Console Enabled
H3-2021-0036	Unauthenticated Access to Elasticsearch
H3-2021-0035	NBT-NS Poisoning Possible
H3-2021-0034	LLMNR Poisoning Possible
H3-2021-0033	mDNS Poisoning Possible
H3-2021-0032	Credential Reuse
H3-2021-0031	Public Access to Git Repository
H3-2021-0030	SMB Signing Not Required
H3-2021-0029	AWS Unrestricted Assume Role Access
H3-2021-0028	Weak Password Strength Requirements
H3-2021-0027	Weak Account Lockout Threshold
H3-2021-0026	Public Self-Signed Certificate
H3-2021-0025	Expired SSL/TLS Certificate
H3-2021-0024	Dangling DNS Record
H3-2021-0023	Public Access to Azure Blob Storage Container
H3-2021-0022	IPV6 DNS Hijacking Possible Using Mitm6
H3-2021-0021	Weak or Default Credentials - Web Applications
H3-2021-0020	Weak or Default Credentials - Cracked Credentials
H3-2021-0019	Weak or Default Credentials - Password Spray
H3-2021-0018	Weak or Default Credentials - Postgres
H3-2021-0017	Weak or Default Credentials - MySQL
H3-2021-0016	Weak or Default Credentials - Microsoft SQL Server
H3-2021-0015	Weak or Default Credentials - SNMP
H3-2021-0014	Weak or Default Credentials - SSH
H3-2021-0013	Weak or Default Credentials - Telnet
H3-2021-0012	Weak or Default Credentials - FTP
H3-2021-0011	Kerberos Pre-Authentication Disabled
H3-2021-0010	Unauthenticated Docker Engine API Access
H3-2021-0009	Unauthenticated Docker Registry API Access
H3-2021-0008	Unauthenticated Etcd Access
H3-2021-0007	Kubernetes Service Account Token Exposure
H3-2021-0006	Unauthenticated Kubernetes API Server Access

Weakness ID	Name
H3-2021-0005	Unauthenticated Kubelet API Remote Code Execution Vulnerability
H3-2021-0004	Kubernetes Privileged Container Exposure
H3-2021-0003	Unauthenticated Access to Sensitive Kubelet API Endpoints
H3-2021-0002	Subdomain Takeover
H3-2021-0001	Public Access to Amazon S3 Bucket
H3-2020-0030	Android Debug Bridge (ADB) over TCP Enabled
H3-2020-0029	Print Spooler Service on Domain Controller Enabled
H3-2020-0028	FTP Directory Traversal Vulnerability
H3-2020-0027	Vulnerable SSL Weak Ciphers
H3-2020-0026	Vulnerable SSL RC4 Algorithm
H3-2020-0025	Vulnerable SSL Version
H3-2020-0024	Vulnerable SSL Insecure Client Renegotiation
H3-2020-0023	Apache Hadoop YARN ResourceManager Unauthenticated Command Execution
H3-2020-0022	Insecure Java JMX Configuration
H3-2020-0021	Unauthenticated Access to the Jenkins Script Console
H3-2020-0020	Credential Reuse Identified
H3-2020-0019	Outdated Vulnerable Software Versions Detected
H3-2020-0018	Fundamentally Insecure Protocols Detected
H3-2020-0017	IPMI Cipher Zero Vulnerability
H3-2020-0016	Insecure IPMI Implementation
H3-2020-0015	Lack of Network Segmentation/Segregation
H3-2020-0014	Weak or Default Credentials
H3-2020-0013	SMB Relay Attack Possible
H3-2020-0012	LLMNR/NBT-NS Poisoning Possible
H3-2020-0011	Weak NTFS Permissions
H3-2020-0010	NFS UID/GID Manipulation Possible
H3-2020-0009	Weak NFS Export Permissions
H3-2020-0008	Guest Account Enabled
H3-2020-0007	SMB Null Session Allowed
H3-2020-0006	LDAP Null Bind Allowed
H3-2020-0005	Anonymous FTP Enabled
H3-2020-0004	Zone Transfer Allowed to Any Server
H3-2020-0003	Anonymous Access to Printer using PJL or PS
H3-2020-0002	Anonymous Access to ZooKeeper API
H3-2020-0001	Remote Desktop Username Disclosure

11.5.2 Weakness Category Definitions

This page explains the four **Category** labels that the NodeZero Portal applies to weaknesses discovered in pentests, with details about the issues that trigger each category.

Security Misconfiguration

These are configuration choices that make a system easier to abuse, even if no specific CVE (Common Vulnerabilities and Exposures) ID is involved. They include lax or disabled security settings on the OS, applications, network gear, or security tools. Examples are:

- EDR (Endpoint Detection and Response) not blocking what it should.
- SMB (Server Message Block) signing disabled.
- Overly permissive firewall rules.
- Misconfigured Kubernetes Cluster.
- Cloud services left overly exposed.

These are settings that technically “work,” but create exploitable behavior or unnecessary attack surface.

Security Controls

These are missing controls – situations where a control that **should** exist is instead absent, disabled, or no longer functioning – allowing attacks that a reasonable control would have stopped. Examples are:

- Hosts or segments with no EDR deployed.
- Lack of MFA (multi-factor authentication) on high-value accounts.
- Lack of segmentation between sensitive zones.
- Missing hardening/baseline controls.
- Security tooling removed since the last test.

NodeZero often surfaces these when it’s able to perform post-exploitation techniques – like credential dumping or lateral movement – that a properly deployed control would have blocked.

(Weak) Credentials

These are credentials that are easily guessed, cracked, reused, exposed, or otherwise unsafe from an attacker’s perspective. Examples are:

- Unmodified default passwords.
- Simple/guessable passwords.
- Cracked hashes.
- Credential reuse across local and domain accounts.
- Credentials harvested via poisoning LLMNR (Link-Local Multicast Name Resolution) or NBT-NS (NetBIOS Name Service).
- Passwords failing AD (Active Directory) policy or AD Password Audit checks.

This category includes both external and internal identity issues where attackers can “”og in instead of hack in.”

Vulnerability

These are software or platform flaws (typically CVEs) where the code itself is vulnerable, and where NodeZero can actually exploit them in your environment. Examples are:

- OS or application CVEs.
- CISA KEVs (Known Exploited Vulnerabilities).
- Zero-day or N-day vulnerabilities that NodeZero is able to weaponize into real attack paths (via remote code execution, privilege escalation, etc.).

In the NodeZero UI, “Vulnerability” is distinct from the other categories. It identifies underlying code defects, which NodeZero cares about only when it can use them in a real attack chain. This distinction is between flaws that are exploitable versus just “present.”

Categories to Impacts

All four buckets roll up under “weaknesses” in NodeZero, meaning actions an attacker can actually use to reach an objective or impact. They are scored and prioritized based on the downstream impacts they enable – such as domain compromise, data exposure, or ransomware – not just their label or a raw CVSS (Common Vulnerability Scoring System) score.

11.5.3 Weakness Scoring System

This page outlines how NodeZero calculates a weakness' severity. It delineates the two overall scores (base versus context score), outlines their components, and explains how all ingredients combine into a final score.

Base Versus Context Score

Every weakness in the Horizon3 AI platform carries two scores:

- **Base Score** – The inherent severity of the vulnerability itself, independent of the environment. Scale: 0–10.
- **Context Score** – The adjusted severity, accounting for the weakness' actual operational impact within the client's specific environment. This reflects what an attacker can actually reach and compromise by exploiting this weakness. Scale: 0–10.

Base Score

The base score is calculated based on a combination of publicly known CVE (Common Vulnerabilities and Exposures) weaknesses and proprietary Horizon3 AI findings.

CVE WEAKNESSES

For CVE-based weaknesses, the base score is the CVSS v3.1 Base Score, as published in the NVD (National Vulnerability Database). CVSS (Common Vulnerability Scoring System) scores are derived from standardized metrics, including:

- **Attack Vector** – Network, Adjacent, Local, or Physical.
- **Attack Complexity** – Low or High.
- **Privileges Required** – None, Low, or High.
- **User Interaction** – None or Required.
- **Impact on Confidentiality, Integrity, and Availability** – None, Low, or High.

Examples:

- CVE-2024-7593: CVSS base score = 9.8 (Critical).
- CVE-2023-32315: CVSS base score = 7.5 (High).

HORIZON3 AI–DISCOVERED WEAKNESSES

For Horizon3 AI-defined weaknesses – misconfigurations, insecure defaults, protocol-level issues, and other findings discovered by NodeZero's testing methodology – Horizon3 AI's security research team assigns base scores in alignment with CVSS scoring standards. Each Horizon3 AI weakness definition includes a manually curated base score reflecting the inherent severity of the issue.

Examples:

- H3-2025-0027 (Authentication Bypass): base score = 9.8.
- H3-2024-0001 (Overprivileged Cloud Policy): base score = 9.0.
- H3-2021-0018 (Default Database Credentials): base score = 8.6.
- H3-2023-0006 (Pass-the-Hash Detection): base score = 7.5.
- H3-2020-0004 (DNS Zone Transfer): base score = 4.8.

These weaknesses can lower the context score

Some Horizon3 AI–discovered weaknesses also carry conditional modifiers that can lower the context score below the base score. For example, if an exploit was attempted but failed, we might set the context score lower than the base score (e.g., to 50% of the base score value). These adjustments reflect the lower real-world risk when exploitation is unsuccessful.

Context Score

The context score is what makes Horizon3 AI's scoring actionable. It answers the question: "Given this specific environment, and taking the perspective of the threat actor’s starting point, what is the real-world impact of this weakness?"

CONTEXT CORE PRINCIPLE

A weakness' context score is set to its **maximum downstream impact** score – that is, the highest-severity impact that can be reached by exploiting that weakness through the discovered attack paths.

For example: If a medium-severity weakness (base score 5.0) leads through a chain of attack steps to Domain Compromise (impact score 10.0), its context score is elevated to 10.0, to reflect that actual risk.

HOW CONTEXT SCORES ARE CALCULATED

The context scoring pipeline works in stages, propagating scores from impacts back up through the attack graph to individual weaknesses.

Stage 1: Score Impacts

NodeZero models 21 distinct impact types, each with a predefined base score that reflects its business severity. The initial set of impacts identifies the nature of the compromise:

- **10.0** – Domain Compromise, AWS Account Compromise, Cloud Compromise, Kubernetes Cluster Compromise, Microsoft Entra Full Tenant Compromise.
- **9.8** – Business Email Compromise.
- **9.5** – Critical Infrastructure Compromise.
- **9.2** – Host Compromise, Perimeter Breach.
- **9.0** – Domain User Compromise, AWS User/Role Compromise, Microsoft Entra User Compromise, Third-Party SaaS User Compromise, Kubernetes Identity Compromise.
- **8.5** – Web Session Compromise.
- **8.0** – Web Application User Compromise.
- **7.8** – Ransomware Exposure.
- **7.5** – Cloud Service Compromise, Brand Compromise.
- **7.2** – Sensitive Data Exposure.

Each impact's context score might be further elevated based on:

- **Business risk associations** – If the affected entity (host, credential, or data store) is tied to identified business risks, the impact score receives a +2.0 boost (capped at an overall score of 10.0).
- **Affected entity scoring** – Vulnerable entities carry their own scores (covered in the next section), which feed into impact scoring.

Stage 2: Score Entities Along the Attack Path

Entities (credentials, data stores, and hosts) that NodeZero discovers and traverses along the attack path receive their own scores, based on privilege level, data sensitivity, and access type.

Credentials

Credentials are scored by privilege level:

- **9.8** – AWS Root User, Domain Admin, Entra Global Admin, AWS Admin User/Role.
- **8.0** – Local Admin, Entra User.
- **5.0** – Domain User.
- **2.0** – Kubernetes Service Account, Other.
- **0.1** – Anonymous.

Data Stores

Data stores (the file systems, databases, shares, and other storage systems that NodeZero discovers) are scored based on a combination of:

- Logarithmic scaling of the volume of data resources discovered: `log10(resource_count)`.
- The highest-scoring sensitive file or data finding within the store.
- Whether the store contains data associated with business risks.

Sensitive Data and Files

As NodeZero discovers sensitive data and files within data stores, it scores them based on multiple factors.

Volume: This is a logarithmic scaling of the discovered resource count – `log10(resource_count)` – with a minimum score of 0.1.

File sensitivity: Files are scored based on file names and extensions known to be sensitive.

High-sensitivity files (score 9.0) include:

- Credential stores and key material: `.pfx`, `.p12`, `.keytab`, `.kdbx` (KeePass), `.keystore`, `.ccache` (Kerberos)
- SSH private keys: `id_rsa`, `id_dsa`
- Credential files: `credentials` (e.g., AWS/Azure credential files), `.htpasswd`, `shadow`, `secrets.tdb`, `passdb.tdb`
- Ransomware indicators: `.lockbit`, `HELP_DECRYPT.*`
- Kerberos ticket caches: files prefixed with `krb5` or `ccache_`

Moderate-sensitivity files (score 5.0–7.0) include:

- Documents and spreadsheets: `.docx`, `.xlsx`, `.pdf`, `.pptx`, `.csv`
- Email archives: `.pst`, `.ost`
- Databases: `.mdb`, `.accdb`
- Archives: `.zip`, `.tar`, `.7z`, `.rar`
- Configuration files: `web.config` (7.0), `.bash_history` (7.0)

Files containing data associated with business risks (e.g., regulated personal data, or financial records) receive a score of 9.0.

Data Store Final Score

A data store's overall score is the maximum score across volume-based scoring, the highest individual file score, and the highest business-risk–related file score.

Credential Final Score

Credential access to a system (representing a credential accessing a specific system, such as a data store) is scored using a probability union of multiple risk signals:

- The credential's base score.
- The data store's base score (if applicable).
- Write access combined with business risks: 9.0.
- Business risks alone: 7.0.
- Write access alone: 5.0.

Stage 3: Propagate Scores Upstream to Weaknesses

Starting from the impacts, NodeZero walks back up the attack graph:

1. Each weakness – a specific vulnerability on a specific asset – receives a context score. This score is equal to the maximum impact score among all downstream impacts reachable through the attack paths that originate from that weakness.
2. Elevation rule – if the maximum downstream impact score exceeds the weakness' base score by more than 0.1, the context score is set to that downstream impact score.
3. Each weakness also tracks:

- **Downstream impact count** – how many distinct impacts are reachable.
- **Downstream impact types** – which categories of impact are reachable (e.g., Domain Compromise, Sensitive Data Exposure).
- **Leverage score** – the ratio of the weakness' downstream impact scores to the operation's total impact score. This leverage score tells you this weakness' share of the overall risk.

Probability Union Formula

When combining multiple independent risk signals, NodeZero uses a probability union formula rather than a simple addition or maximum:

1. Normalize each score from `[0, 10]` to `[0, 1]`.
2. Combine using: `P(A or B) = P(A) + P(B) − P(A) * P(B)`.
3. Scale the result back to `[0, 10]`.

This models the probability of at least one risk factor resulting in compromise. As more risk factors accumulate, it naturally saturates toward 10.0, without exceeding the scale.

As an example, here is how NodeZero would scores of 5.0 and 7.0:

- Normalized: 0.5 and 0.7.
- Union: 0.5 + 0.7 – (0.5 x 0.7) = 0.85.
- Final score: 8.5.

Business Risks

Business risks represent the real-world consequences that matter most to an organization. NodeZero defines these risks categorically, and maps them to specific types of hosts, users, and data, based on their role in the environment. When an entity along an attack path is associated with a business risk, its score is elevated by +2.0 (capped at an overall score of 10.0).

The system defines the following business risk categories:

- **Operational Disruption** – Degraded performance, corrupted configurations, or blocked dependencies that cripple business processes.
- **Critical System Shutdown** – Complete disabling of ERP (Enterprise Resource Planning), email, manufacturing, or other critical systems.
- **Revenue Interruption** – Disabling revenue-critical systems; or halting product delivery, payroll, or customer transactions.
- **Software Delivery Disruption** – Tampering with software pipelines, delaying releases, or inserting malicious code.
- **Supply Chain Breakdown** – Vendor or logistics compromise, cascading into missed shipments and failed deliveries.
- **Leak of Sensitive Communications** – Exposure of board-level slide decks, legal strategies, or HR files that could spark media crises.
- **Executive Fraud & Impersonation** – CEO/CFO impersonation, enabling fraudulent approvals and reputational damage.
- **Regulatory Breach Penalties** – Exposure of protected data, triggering HIPAA, PCI, or GDPR fines, audits, and disclosure obligations.
- **Theft of Intellectual Property or Research and Development** – Stolen code, formulas, or designs, enabling competitors to replicate innovation.
- **Unauthorized Physical Access** – Compromise of surveillance, alarm, or access control systems, enabling physical intrusion.

NodeZero associates these business risks with entities, such as:

- Hosts (e.g., Domain Controllers map to Critical System Shutdown).
- Users (e.g., Corporate Executives map to Executive Fraud & Impersonation.
- Data types (e.g., Payment Card Data maps to Regulatory Breach Penalties).

Summary: What Drives a Weakness' Final Score

- **CVSS/H3-defined base score** – Sets the baseline. This is the inherent severity of the vulnerability.
- **Downstream impact type** – If exploitation leads to Domain Compromise (10.0),the context score rises to 10.0.
- **Attack paths to impacts** – The context score is ultimately determined by the highest-severity impact reachable by an attacker.
- **Credential privilege levels along the path** – Domain admin credentials elevate path scores to 9.8, compared to standard user credentials that are scored at 5.0.
- **Data exposure volume and sensitivity** – Large volumes of sensitive data increase entity scores along the path.
- **Write access to data stores** – Write access contributes additional risk signal, via probability union.
- **Business risk associations** – When affected entities map to identified business risks, this adds a +2.0 boost.
- **Exploit success/failure** – As noted above, some Horizon3 AI–discovered weaknesses lower the context score below the base score when exploits fail.

Key Takeaways

1. **Base scores are directly comparable to CVSS.** CVE weaknesses use CVSS verbatim. Horizon 3 AI–discovered weaknesses are scored on the same 0–10 scale, using equivalent criteria.
2. **Context scores reflect real-world exploitability.** These scores answer "What can an attacker actually achieve with this weakness in this environment?," rather than just "How severe is this vulnerability in isolation?"
3. **Scoring is driven by attack path analysis.** A weakness that leads to Domain Compromise, through a chain of exploited credentials and lateral movement, will score higher than an identical weakness on an isolated system with no downstream impact.

4. **Business-risk mapping ties technical findings to organizational impact.** Weaknesses that affect systems associated with regulatory data, critical infrastructure, or executive access are scored higher, to reflect the business consequences of a corresponding compromise.

11.6 Exposure Score Levels

The results of a completed pentest are assigned an exposure level based on the type of test (Internal versus External), and on the weaknesses and impacts found during the test. The following table details the definitions for the exposure levels given:

11.6.1 Internal Pentests

Exposure Score	Reasons for This Score
Critical	Any of the discovered impacts fall into one of these categories: Domain Compromise, AWS Account Compromise, AWS User Compromise, Azure AD User Compromise, Business Email Compromise; OR 20% of the discovered hosts have a host compromise impact.
High	One or more impacts OR at least one critical weakness discovered.
Medium	No impacts discovered AND one or more high weaknesses discovered.
Low	No impacts discovered AND one or more medium or low weaknesses discovered.
None	No weaknesses or impacts discovered.

11.6.2 External Pentests

Exposure Score	Reasons for This Score
Critical	Any impact EXCEPT Sensitive Data Exposure or Brand Compromise.
High	One or more impacts discovered OR at least one weakness discovered with context score >= 5.0.
Medium	No impacts discovered AND at least one weakness discovered with context score between 3.0 and 5.0.
Low	No impacts discovered AND at least one weakness with context score <= 3.0.
None	No weaknesses or impacts discovered.

11.6.3 Determining Overall Exposure

NodeZero reports an overall exposure level based on a composite of the highest score of each finding, and the corresponding pentest type (internal or external). We score external pentests more severely than internal pentests, due to the extra exposure risk. Because this is a weighted determination, the overall exposure score might be higher or lower than the highest individual severity finding in a given pentest.

11.7 Attack Configurations

An Attack Configuration in NodeZero refers to a set of customizable settings that define how the system behaves during a penetration test. These settings allow you to enable or disable specific attack modules to tailor the pentest according to your target environment and objectives. We make these Attack Configuration Options individually configurable because their inclusion variously affects the performance of the pentest, or has the potential to disrupt the target environment. This flexibility allows NodeZero to adapt to different test scenarios, and helps ensure that tests are focused and effective.

The image below shows a screenshot from an Internal Pentest.

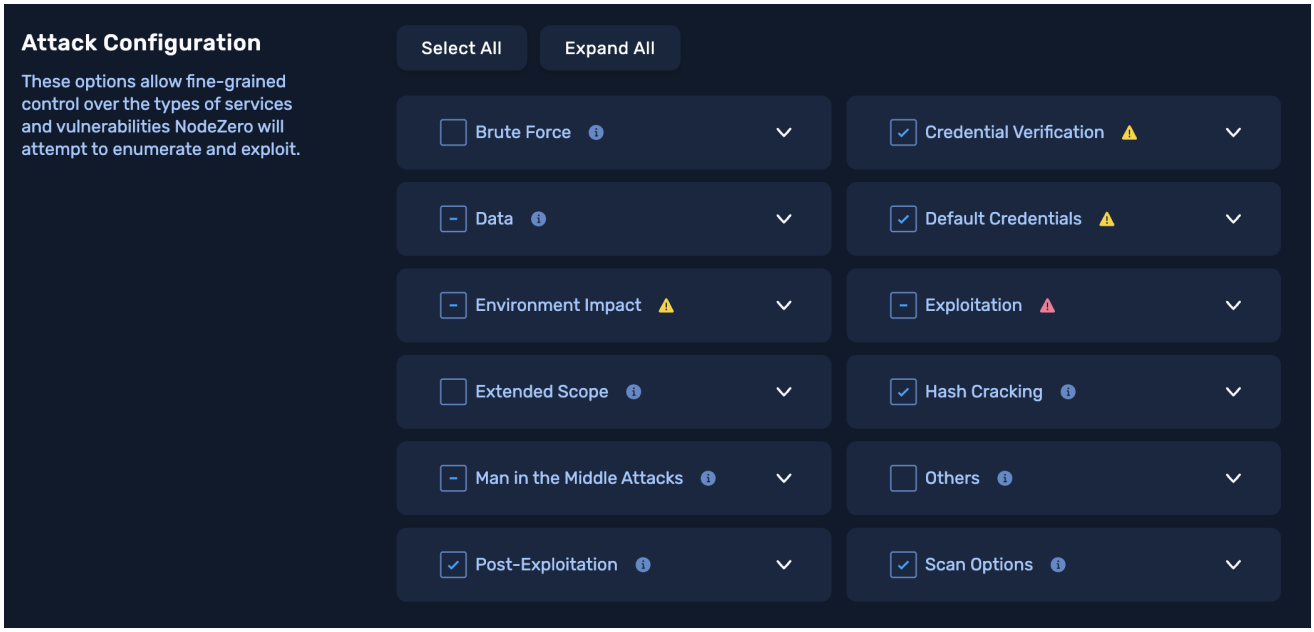


Figure 1 - Screenshot of the attack configuration options for an internal pentest

Attack Configuration options may vary depending on the pentest type.

When all options are disabled, the operation is still a pentest. Click for more info.

The following activities are performed:

- Asset Discovery
- Identifying potential vulnerabilities
- Exploiting most vulnerabilities/misconfigurations (that have been vetted to not have an operational impact on the target)
- Limited credential discovery and credential pivoting

The following activities are not performed:

- Windows Active Directory attacks
- Man-in-the-middle attacks
- Hash cracking
- Password Spray
- Azure AD pivoting
- Default Cred checking
- OS credential dumping
- Any brute force enumeration
- Any exploits specifically disabled in the advanced config (but most exploits are still executed as described above)

11.7.1 Categories and Options

Below is a summary of the different attack configuration categories and their available options:

Brute Force

This category involves using brute-force techniques to discover sensitive resources such as DNS records, subdomains, and cloud storage buckets, often by applying wordlists or domain names.

Option	Description	Risk
DNS	Enables brute-forcing of internal DNS records. Only applies if an operation has been scheduled with the Intelligent Scope option. This may place noticeable load on DNS servers.	low
Domain User	Enumerate domain users using brute force methods against a domain controller. Disabling this flag will hinder the execution of certain attack techniques.	low
S3	Enables brute-force discovery of S3 buckets using wordlists and top-level company domain names. This can add significant time if the pentest runs against many top-level domains.	none
Subdomains	Enables brute-force discovery of company subdomains using a large wordlist of common subdomain names. This can aid in discovering more external assets but significantly extend the time it takes.	none

Data

This section covers options for scanning and analyzing sensitive data on SMB and NFS shares. These configurations help assess the security of network shares and verify permissions, providing a deeper view of data risks during a pentest. Enabling these options may increase the pentest duration.

Option	Description	Risk
Domain Admin Scanning of SMB Shares	Enables scanning of SMB shares using domain administrator credentials that were injected into the pentest or discovered during the course of the pentest. This provides a more complete picture of data risk but can add significant time to the pentest.	none
Extended Domain User Scanning of SMB Shares	Enables scanning of all SMB shares accessible to domain users whose credentials were injected into the pentest or discovered during the course of the pentest.	none
Extract and analyze files from NFS shares	Retrieve, parse, extract, and analyze data from sensitive files on NFS shares. Files are analyzed one at a time and deleted after the analysis is complete.	none
Verify Permissions on SMB Shares	Verify read, write, list, and delete permissions on an SMB share by writing a test file and deleting it afterward. Cleanup of the test file may fail in exceptional circumstances.	none

Environmental Impact

This section configures tests that assess the impact of penetration testing on the environment, including vulnerabilities, misconfigurations, and access control weaknesses in critical systems.

Option	Description	Risk
Active Directory - Creation of Machine Account	Attempts to add a domain machine account during exploitation of common AD vulnerabilities and misconfigurations (e.g. CVE-2022-33679, CVE-2022-26923, RBCD attacks). Deletion of machine accounts may fail if the correct permissions are not obtained.	none
ADCS ESC4 Attack - Misconfigured Templates Access Controls	Exploit vulnerable Active Directory Certificate Templates that allow an unprivileged user to overwrite Certificate Template security features -- enabling Subject Alternative Name (SAN). Restoration of original template configuration may fail in exceptional cases.	none
Anonymous Docker Engine Write Check	Checks for write privileges against a Docker Engine instance that allows anonymous (unauthenticated) access. The check attempts to create a Docker container or pull a Docker image and deletes the container or image afterwards.	none
Anonymous Printer Access	Check for anonymous access to printers over port 9100. This check may cause certain printer models to print out pages.	moderate
Anonymous ZooKeeper Write Check	Checks for write privileges against a ZooKeeper instance that allows anonymous (unauthenticated) access. The check writes to a ZooKeeper node and deletes it afterwards.	none
Application Administrator Account Creation	Some exploits require the creation of an administrative user for an application. In certain instances, NodeZero is unable to remove this user after exploitation. Manual cleanup will be required.	none
Atlassian Crowd and Crowd Data Center Remote Code Execution (CVE-2019-11580)	Checks for exploitability of CVE-2019-11580 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none
AWS IAM Privilege Escalation	Allow NodeZero to attempt IAM privilege escalation by modifying AWS identity and trust relationship policies. There is a small chance setting policies back to their original settings may fail.	none
Elasticsearch Write Check	Checks for write privileges against an Elasticsearch cluster. The check attempts to create an index and deletes it afterwards.	none
Entra ID App/Directory Role Privilege Escalation	Attempt Privilege Escalation in Entra ID by adding App or Directory Roles to a user or service principal. Cleanup may fail -- leaving the affected object with elevated privileges.	none
Entra ID Application Service Principal Abuse	Allow NodeZero to abuse vulnerable Entra Service Principals by modifying application owners, and adding credentials to Service Principals. There is a small chance cleanup of these artifacts fails.	none
FTP Write Check	Checks for write privileges against an FTP server. The check creates a remote directory and deletes it afterwards.	none
Insecure JMX (H3-2020-0022)	Tests exploitability of the insecure JMX weakness (H3-2020-0022). The test checks for remote code execution by installing a payload on the vulnerable JMX service, runs a small set of commands using the payload, and uninstalls the payload at the end. There is a small chance that cleanup of the payload may fail.	none

Option	Description	Risk
ManageEngine ServiceDesk Plus PreAuth RCE (CVE-2021-44077)	Checks for exploitability of CVE-2021-44077 by uploading a malicious payload through that API, and execute the payload through another API. This upload is likely to be caught by AV software on the host. If successful, this exploit will leave behind a file msiexec.exe in the ManageEngine\ServiceDesk\site24x7 folder.	none
Set Expired Credentials	Allow NodeZero to set expired credentials to further attacks.	none
Subdomain Takeover	Proactively takeover and hold onto subdomains that are vulnerable to subdomain takeover (H3-2021-0002) to prevent bad actors from compromising them first.	none
VMWare vCenter Server Access Control Vulnerability (CVE-2020-3952)	Checks for exploitability of CVE-2020-3952 by adding an administrative user and removing it afterwards.	none
VMWare vCenter Server Plugin Remote Code Execution Vulnerability (CVE-2021-21972)	Checks for exploitability of CVE-2021-21972 by installing a webshell, executing a command within the webshell, and removing it afterwards. For vCenter servers running on Linux, it is possible that randomly-named webshells will be left behind on the vulnerable vCenter server if the exploit fails.	none
VMWare vRealize Operations Manager SSRF Vulnerability (CVE-2021-21975)	Checks for exploitability of CVE-2021-21975 and CVE-2021-21983 by installing a randomly named webshell, executing a command within the webshell, and removing it afterwards. Cleanup of the webshell may fail in exceptional cases.	none
Zoho ManageEngine ADSelfService Plus API Auth Bypass (CVE-2021-40539)	Checks for exploitability of CVE-2021-40539 by uploading a malicious JAR file. This upload is likely to be caught by AV software on the host. Cleanup of the JAR file may fail in exceptional cases.	none

Extended Scope

This section includes advanced tests for vulnerabilities in mobile device management systems and other extended targets.

Option	Description	Risk
Intune Mobile Device Management Remote Code Execution (RCE)	Allow NodeZero to attempt RCE on devices running Intune MDM by uploading platform scripts to Microsoft Endpoint Manager.	none

Man in the Middle Attacks

Modules in this section perform man-in-the-middle (MITM) attacks to intercept and manipulate network traffic, capturing cleartext credentials and performing attacks like NTLM hash relaying.

Option	Description	Risk
Expanded LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This will sniff all available traffic regardless of scope.	none
Limited LLMNR and NetBIOS poisoning	Enables sniffing of cleartext passwords and hashes sent over insecure protocols such as LLMNR, NetBIOS, SMB v1, and HTTP. This is limited to the scope provided during the configuration of the pentest. If selected, this option overrides the 'Expanded LLMNR and NetBIOS poisoning' option.	none
Net-NTLM Authentication Coercion	Enables Net-NTLM Authentication coercion techniques. This allows attackers to capture Net-NTLM (NTLMv2) hashes by coercing machines to authenticate to an attacker controlled server.	none
Net-NTLM Hash Relaying	Enables SMB relay attacks. This allows attackers to gain unauthorized access to machines by capturing Net-NTLM (NTLMv2) hashes over the network and relaying them to target SMB servers.	none

Post Exploitation

This section contains actions that are performed after compromising a host, such as deploying remote access tools, dumping credentials, and escalating privileges.

Option	Description	Risk
NodeZero Remote Access Tool	Allows NodeZero to load Remote Access Tools onto compromised hosts to further attack paths. Depending on the type of weakness exploited, files may be left behind on disk within the C:\Windows folder.	none
SSH	Enables post-exploit actions such as system enumeration and privilege escalation on hosts for which SSH access was gained. In exceptional circumstances, files may be left on disk in the /tmp folder.	none
Windows Credential Dumping - DPAPI Secrets	Enables dumping of credentials from secrets encrypted with DPAPI keys after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	none
Windows Credential Dumping - LSA Secrets	Enables dumping of credentials from the Local Security Authority (LSA) after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	none
Windows Credential Dumping - LSASS	Enables dumping of credentials stored in the Local Security Authority Subsystem Service (LSASS) process, after gaining administrative access to a Windows machine. In exceptional circumstances, cleanup may fail, leaving files on disk.	low
Windows Credential Dumping - SAM	Enables dumping of credentials from the Security Account Manager (SAM) database after gaining administrative access. In exceptional circumstances, cleanup may fail, leaving files on disk.	none

Credential Verification

Modules in this section use discovered credentials to gain access to various services, including internal password spraying and credential pivoting.

Option	Description	Risk
Credential Reuse	Checks for access to services and shares using local user (non-domain) authentication.	none
Domain User	Checks for Windows domain user access by authenticating with credentials against the SMB service running on the Windows Domain Controller.	none
Internal Password Spray	Enables password-spraying domain users with common passwords. We recommend enabling the Brute Force: Domain User flag along with this option. By default, a given user will be tried only twice every 60 minutes. If NodeZero detect three or more users as locked out from a given domain, it stops spraying passwords on that domain.	moderate
MS Entra (Azure AD) Credential Pivoting	Enables using domain user credentials discovered in an internal pentest against MS Entra services. Utilizes user-entered Domains from the OSINT step or Entra domains discovered via other methods.	none
Azure AD Password Spray	Enables password-spraying Azure cloud users with common passwords. By default, a given user will be tried only three times every 60 minutes. There is a small chance of locking out accounts.	moderate

Default Credentials

This section involves checking for the use of default credentials on a variety of services like FTP, SQL databases, and web servers.

Option	Description	Risk
FTP	Enables checking default credentials against FTP services found by NodeZero.	low
Microsoft SQL Server	Enables checking default credentials against Microsoft SQL Server databases found by NodeZero. There is a small chance of locking out the sa account.	moderate
MongoDB	Enables checking default credentials against MongoDB databases found by NodeZero.	low
mySQL	Enables checking default credentials against MySQL databases found by NodeZero.	low
PostgreSQL	Enables checking default credentials against PostgreSQL databases found by NodeZero.	low
SNMP	Enables checking default SNMP v1 community strings against SNMP services found by NodeZero.	low
SSH	Enables checking default credentials against SSH services found by NodeZero. Against older ESXi servers vulnerable to CVE-2019-5528, this module may trigger a partial denial of service condition in the hostd process.	moderate
Telnet	Enables checking default credentials against telnet services found by NodeZero.	low
Web	Enables checking default credentials against HTTP or HTTPS web servers found by NodeZero.	low

Exploitation

Modules in this section test for the exploitability of known vulnerabilities, such as EternalBlue and BlueKeep, which could potentially crash or disrupt services.

Option	Description	Risk
Bluekeep (CVE-2019-0708)	Tests exploitability of the Bluekeep vulnerability (CVE-2019-0708). There is a moderate-level risk this exploit may crash the target host, and it is not recommended for use against production systems.	high
Cisco Smart Install Vulnerability (CVE-2018-0171)	Tests exploitability of the Cisco Smart Install vulnerability (CVE-2018-0171). The test attempts to pull router config from the vulnerable router via the TFTP protocol. Against a few older models of Cisco routers, running this exploit may cause the router to reload or go down.	moderate
EternalBlue (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerability EternalBlue. This is a kernel buffer overflow exploit and carries a moderate risk of crashing the target. It is not recommended for use against production systems. This exploit is only attempted if NodeZero is able to reliably determine the target operating system and NodeZero is not able to first exploit EternalChampion/EternalSynergy/EternalRomance.	moderate
EternalChampion/ EternalSynergy/ EternalRomance (MS17-010)	Tests exploitability of the Windows SMB remote code execution vulnerabilities EternalChampion, EternalSynergy, and EternalRomance.	low
Exploding Can (CVE-2017-7269)	Tests exploitability of the IIS 6.0 WebDAV vulnerability CVE-2017-7269, aka Exploding Can.	low
Heartbleed (CVE-2014-0160)	Tests exploitability of the Heartbleed vulnerability (CVE-2014-0160), if discovered by NodeZero. This test dumps memory from the vulnerable server.	low
HP iLO Web API Remote Code Execution (CVE-2017-12542)	Tests exploitability of the HP iLO Web API Remote Code Execution vulnerability (CVE-2017-12542). The test attempts to retrieve users and their credentials by exploiting a heap-based buffer overflow.	low
Server Service Vulnerability (MS08-067)	Tests exploitability of the Windows SMB remote code execution vulnerability CVE-2008-4250, aka MS08-067. There is a high likelihood that this exploit will crash the SMB service on the target after successful exploitation.	high

Hash Cracking

This section automatically attempts to crack hashes discovered during the penetration test to gain further insights into the security of the environment.

Option	Description	Risk
Automatic Hash Cracking	Automatically attempt to crack hashes found in the environment.	none

Others

Exploits and vulnerability checks with other side effects.

Option	Description	Risk
RegreSSHion check	Runs the regreSSHion (CVE-2024-6387) check. This check may result in a high number of false positives.	none

Scan Options

Modules in this section help manage host discovery and asset scanning, including setting packet rate limits and repeating host discovery during a pentest.

Option	Description	Risk
Repeated Host Discovery	Repeatedly attempts to discover hosts throughout a pentest. This may extend the total time of the pentest if a significant number of new hosts are discovered.	none
Scope discovery packets per second	Slow: 150 PPS Normal: 1500 PPS	none

12. Credits

12.1 Credits

This documentation references trade names, trademarks, and registered trademarks of Horizon3 AI Inc. and other organizations. All trademarks referenced here are the property of their respective owners.

12.1.1 Horizon3 AI Inc. Trademarks & Legal Notice

NodeZero® is a registered trademark of Horizon3 AI Inc.

NodeZero Insights™, NodeZero Tripwires™, NodeZero Overwatch™, NodeZero Guardian™, Offense Informs Defense™, and Pentest Wednesday™ are trademarks of Horizon3 AI Inc.

12.1.2 Third-Party Trademarks

macOS® is a registered trademarks of Apple Inc.

MITRE ATT&CK® and ATT&CK® are registered trademarks of the MITRE Corporation.

VMware® and vSphere® are registered trademarks of Broadcom, Inc.

VirtualBox® is a registered trademark of Oracle America, Inc.

Windows® and Hyper-V® are registered trademarks of Microsoft Corporation.

Other trade names mentioned on this site are the property of their respective owners. Use of any third-party marks does not imply affiliation or endorsement.

For further legal terms and conditions, see our Terms of Service and Trust Center .