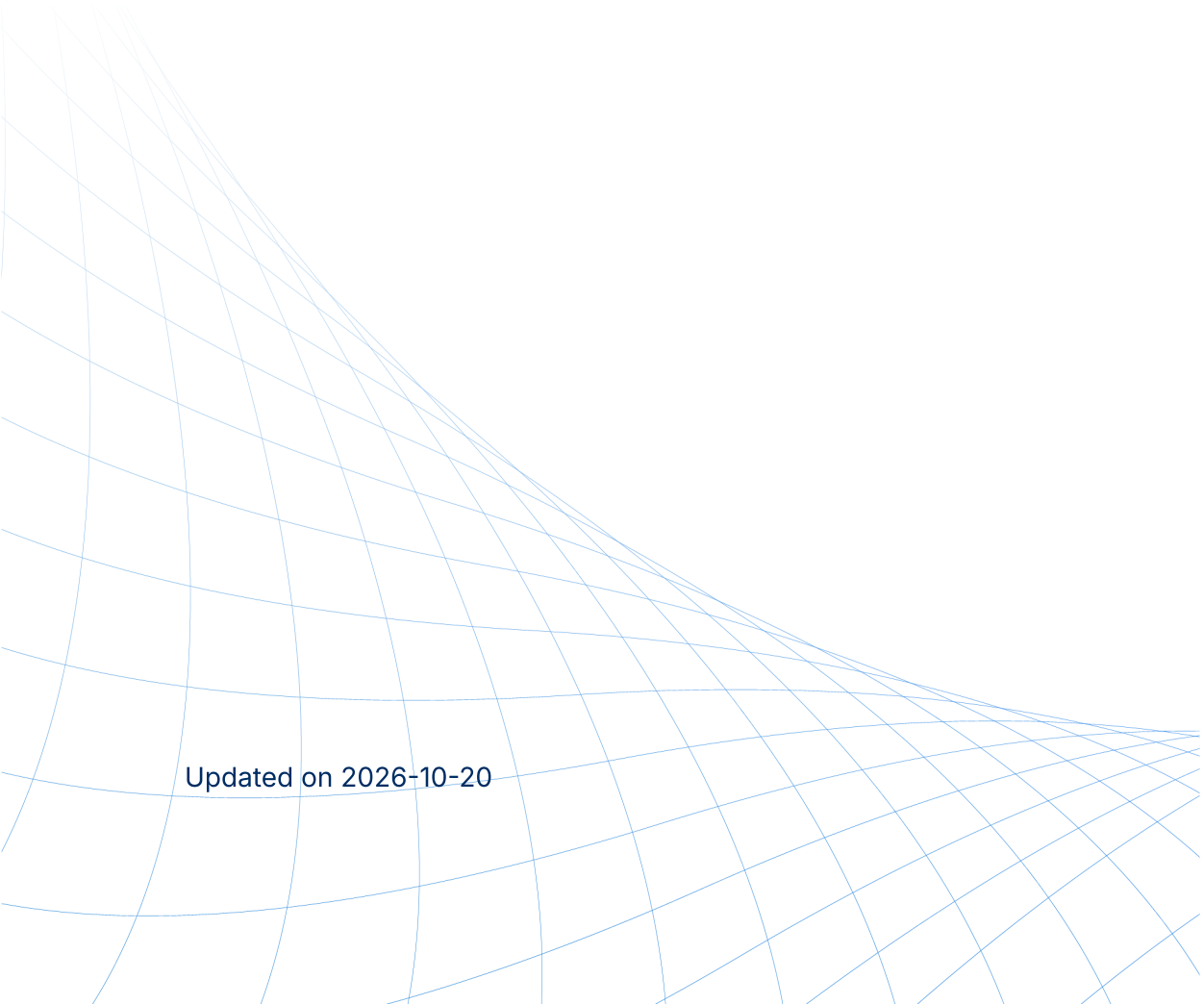
runZero
User Guide
v5.1.261015.1



Updated on 2026-10-20

Copyright © runZero, Inc. 2019-2026

Contents

- User Guide
- runZero
  - Data sources
  - Live inventory
  - Reports
  - Monitoring and alerts
- Quickstart guide
  - 1. Set up an Explorer
  - 2. Run your first scan
  - 3. Configure integrations
  - 4. Review scan results
- Full deployment plan
  - 1. Plan your deployment
  - 2. Initial setup
  - 3. Run discovery and assess
  - 4. Automate and optimize
  - 5. Rollout and monitor
- Creating an account
  - Activating your account
  - Changing your password
  - Updating your profile picture
- Installing an Explorer
  - Installation
  - System requirements
  - Configuration
  - Web screenshots
  - Network communication
  - Restarting an Explorer
  - Removing an Explorer
  - Log management
  - Certificate authorities (CAs)
  - Manual mode
  - Storage locations
  - Container installations
  - Automated installations
  - AWS EC2 installations
  - Automated MSI deployments
  - Installing on a Raspberry Pi
  - Managing Explorers
  - Verifying binaries
- Full-scale deployment
  - 1. Identify key success outcomes
  - 2. Planning your deployment
  - 3. Initial configuration
  - 4. Review assets and exposures
  - 5. Optimization

    - 6. Automation
    - 7. Rollout
    - Additional resources
    - Getting help
    - Types of networks
- Organizations
    - Use cases for organizations
    - Creating organizations
    - Organization configurations
    - Projects
    - Organization and project details
- Sites
    - Use cases for sites
    - Creating sites
    - Site configurations
    - Subnet tagging
    - Importing subnets
    - Importing and exporting sites
    - Sites and Explorers
- Self-hosting runZero
    - Background
    - Requirements
    - Offline mode
    - Installation steps
    - Installation with your own PostgreSQL database
    - runZero updates
    - Managing users
    - CLI service management
    - CLI update management
    - CLI user management
    - CLI organization management
    - Advanced configuration
    - Permissions
    - Backup and restoration
    - Support and debugging
    - Manual migrations
    - Offline mode configuration
    - High-availability configuration
    - Self-hosted troubleshooting
- Data retention
    - Stale asset expiration
    - Offline asset expiration
    - Stale integration attribute expiration
    - Stale vulnerability expiration
    - Scan data expiration
    - Event records
    - AI threads and artifacts
    - Custom integration script versions
    - Data deletion after account termination
- Available roles

    - Superuser
    - Administrator
    - User
    - Billing
    - Annotator
    - Viewer
    - No access
    - Custom roles
- Inviting users
- User details
- Account settings
    - Single sign-on (SSO)
    - Multi-factor authentication (MFA)
    - Disabling support access
    - Idle times and sign in duration
    - Account API keys
    - Roles
    - API clients
    - License information
    - Entity information
    - Audit log
- Custom roles and permissions
    - How permissions work
    - What the built-in roles grant
    - Creating a custom role
    - Assigning custom roles
    - How multiple roles combine
    - Auditing
- Managing user groups
    - Creating user groups
    - Adding users to user groups
    - Setting an expiration date for a user group
    - Viewing users in a user group
    - Viewing user groups assigned to a user
    - Removing users from a user group
    - Deleting user groups
    - Searching for users and user groups
- Bulk importing users
    - Creating the CSV file for importing users
    - Importing users into runZero
    - Verifying users have registered
- Managing external users
    - Inviting an external user into your account
    - Switching clients and organizations as an external user
    - Removing an external user from your account
- Implementing SSO
    - Specific SSO providers
    - Identity provider settings
    - SSO walkthrough
    - Service provider information

    - Common problems
    - Managing SSO group mappings
    - Setting up Microsoft Entra SSO
    - Setting up Okta SSO
- Managing licenses
    - How do I view my license?
    - What counts as recent assets?
    - When does my subscription expire?
    - How do I renew my subscription?
    - How do I convert to the Community Edition?
    - How do I find my invoices?
    - How do I change or cancel my subscription?
- Active scanning
- Traffic sampling
- Inbound integrations
- Active scanning
    - Site
    - Explorer
    - Hosted External Explorer
    - Discovery scope
    - Scan name
    - Scan speed
    - Schedule
    - Scheduling grace period
    - Scan duration limit
    - Advanced scan options
    - Autonomous discovery
    - Initial network scans
    - Identifying gaps in scanning
    - Managing scan templates
    - Scanning with credentials
    - Scanning with SNMP
    - Using custom fingerprints
- Passive sampling
    - Configuring passive sampling
- Protocol support
    - Additional protocols
- Default ports
    - What TCP ports does runZero scan?
    - What UDP ports does runZero scan?
- Scanning IoT and OT
    - Can I safely scan my IoT and OT environments?
    - What makes the Explorer safe for sensitive environments?
    - Where should I deploy the Explorer?
    - OT/ICS vendors runZero fingerprints
    - IoT and OT protocols
- Inbound integrations
    - Enriching runZero results with data from other tools
    - Supported integrations
    - Scan probes or connector tasks

    - Importing integration data
    - Automatic asset merge
    - Removing an integration data source
    - Excluding integration attributes
    - Source names and IDs
    - Amazon Web Services
    - Censys Search & Data
    - Censys Platform
    - Cisco Meraki Dashboard
    - CrowdStrike Falcon
    - Custom Integration Scripts
    - Dragos
    - Google Cloud Platform
    - Google Workspace
    - Microsoft 365 Defender
    - Microsoft Active Directory
    - Microsoft Azure
    - Microsoft Endpoint Configuration Manager (MECM)
    - Microsoft Entra ID
    - Microsoft Intune
    - Miradore MDM
    - NetBox CMDB
- Step 5: Enable unknown assets in recurring sync
    - Qualys VMDR
    - Palo Alto Networks Firewall
    - Palo Alto Prisma Cloud
    - Rapid7
    - SentinelOne
    - Shodan
    - Tailscale
    - Tanium API Gateway
    - Tenable
    - VMware
    - Wiz
- Outbound integrations
    - Using runZero data to enrich other tools
    - Atlassian Insight & Jira Service Management
    - Jira
    - Panther
    - SecurityGate.io
    - ServiceNow Service Graph
    - Splunk Search
    - Sumo Logic
    - Tines
    - Thinkst Canary
- Unified asset inventory
- Total attack surface management
    - External & cloud
    - Internal
    - IoT & OT

    - Inside-out
- Vulnerability detection
- Risk prioritization
- Continuous monitoring
- Rapid responses
- Query library
- Templates
- Findings
- Issues
    - What is an issue
    - Creating an issue
    - Working with an issue
    - Automatic remediation tracking
    - Linking a Jira ticket
    - Permissions
- Reviewing results
    - Task details
    - Dashboard & inventory views
    - Insights from queries
    - Reports
    - Alerts
- Using dashboards
    - Dashboard selection
    - Creating dashboards
    - Customizing dashboards
    - Sharing dashboards
    - Duplicating dashboards
    - Default dashboards
    - Deleting dashboards
    - Widget library
    - Widget types
    - Drill down
    - Printing and exporting
    - Recalculate metrics
    - Themes
    - Display mode
- AI Threads
    - Requirements
    - Threads
    - Tools
    - Artifacts
    - Reporting
    - Limits
    - Retention
- Using the inventory
    - Understanding assets
    - Loading assets
    - Connecting to other systems
    - Viewing services
    - Viewing screenshots

    - Viewing software
    - Viewing vulnerabilities
    - Viewing certificates
    - Viewing wireless networks
    - Viewing users and groups
- Understanding assets
    - Asset fields
- Understanding findings
    - What are findings?
    - Finding categories
    - Findings list
    - Finding details
    - How are findings different from vulnerabilities?
- Asset risk and criticality
    - Defining risk and criticality
    - Assigning asset risk and criticality
    - Asset risk report
- Certificates
    - TLS certificates
- Geolocation
    - Where locations come from
    - GeoIP lookups against MaxMind
    - Cloud region codes
    - Discrete country and city attributes
    - Free-form location strings
    - Manually setting an asset location
    - Searching by location in the asset inventory
    - When locations are refreshed
- Managing ownership
    - Ownership types
    - Default asset owner
    - Assigning owners to assets and vulnerabilities
- Managing tasks
    - Task status values
    - Tabs
    - Task details
    - Scheduled tasks
    - System tasks
    - Dismissing failed tasks
    - Reprocessing tasks
- Tracking goal progress
    - Goal creation
    - Creating saved query goals
    - Creating baseline goals
    - Goal progress calculation
    - Goal events and notifications
- Understanding suppression
    - How suppression works
    - Suppressing and unsuppressing objects
    - Searching for suppressed objects

    - Dashboard widget
    - Suppression events
- Understanding network segmentation
    - runZero multi-homed asset detection
    - Using the bridge report
    - Using the asset route pathing report
- Protocol gateways
    - How it works
    - Supported protocols
    - Enabling enumeration
    - Finding gateways and sub-assets
    - Where sub-assets show up
    - Safety
    - See also
- Understanding fingerprints
    - How fingerprints work
    - Refingerprinting and submitting improvements
    - Fingerprint categories
    - Common fingerprint attributes
    - Using fingerprints in queries
    - Troubleshooting fingerprints
- Managing alerts
    - Using the rules engine
    - Creating alert templates
    - runZero events
- Querying your data
    - Filtering and searching data
    - System and custom queries
    - Creating and editing queries
    - Search query syntax
    - Query examples
    - Inventory keywords
    - Interface keywords
    - Automating queries
- Exporting asset data
- Exporting HP iLO data
    - How to export HP iLO CSV data
    - HP iLO CSV export data
- Network Map
    - Accessing the map
    - 2D and 3D modes
    - Subnet layout
    - Network Intel card
    - Search
    - Tracing paths
    - The Trace panel and per-hop filters
    - Choke points
    - Exporting
    - Sharing a view
- World Map

    - Prerequisites
    - Accessing the map
    - 2D and 3D modes
    - Searching the map
    - Searching the asset inventory by location
    - Exporting
    - Sharing a view
- Organization Overview
    - Generating the Organization Overview Report
    - Email notifications
- RFC 1918 Report
    - RFC1918 coverage report
- External Assets
    - Generating the External Asset Report
    - Email notifications
- Site Comparison
    - Generate a site comparison
    - View how assets change over time
    - View how exposure differs between networks
    - Analyze the results in the site comparison report
    - Search the site comparison report
- External reporting
    - What you can share
    - Sharing a report
    - Share dialog options
    - How recipients view a report
    - Managing reports and snapshots
    - Security and access
- Switch Topology
    - Generating the switch topology report
    - Filtering the switch topology report
    - Limitations of the switch topology report
- Network Bridge Report
    - Accessing the report
    - How bridges are detected
    - Reading the report
    - Querying multi-homed assets directly
    - See also
- Asset Route Pathing
    - Terminology
    - Generating the asset route pathing report
    - Analyzing the report
    - Sharing the report
    - Exporting a dotfile
    - FAQs
- Cryptographic bill of materials
    - Post-quantum status
    - Migration deadlines
    - Interpreting the results
- Leveraging the API

    - API keys and tokens
    - Authentication
    - API rate limiting
    - IP address allowlist
    - runZero API usage examples
    - Additional documentation
- API clients
    - Grants
    - Expiration
    - Creating a user-bound API client
    - Creating an account-bound API client
    - Role locks
    - Managing existing clients
    - Using an API client
    - Credentials created by AI clients
- Using the CLI
    - Scanner
    - runZero Command Line Interface (CLI)
- Model Context Protocol
    - MCP terms
    - Authentication
    - Connecting through OAuth
    - Presenting a credential directly
    - Clients
    - Available tools
    - Prompts and resources
    - Exploring MCP tools, resources, and prompts
- Glossary
    - Terms
- Frequently asked questions
    - Issues and FAQs
    - Identical assets in inventory
    - Scanning routers
    - Scanning VMWare virtual machines
    - Scanning on Windows with Intel AMX
    - Scanning on Windows with EDR or antivirus
    - Explorer not capturing screenshots
    - Explorer security model
    - Browsers supported by the runZero Console
    - Common sign-in issues
    - runZero data formats
    - runZero data dictionary

What is runZero?

runZero

runZero is an attack surface and exposure management platform that helps you see everything on your network, from managed IT systems to unmanaged OT, IoT, cloud, mobile, and remote assets. It combines high-speed active scanning, passive discovery, and API integrations into one continuously updated inventory.

You can run runZero as a hosted service (SaaS) or on-premise. The platform includes one or more Consoles, distributed Explorers that run as lightweight services at your points of presence, and a command-line tool for offline collection and automation. You can work in the web interface, automate through the API, and, on self-hosted deployments, manage operations from the command line.

Data sources

- **Active Scans**: The runZero active scan engine is fast, accurate, and safe for all environments, and it supports many protocols and applications.
- **Passive Traffic Sampling**: The runZero passive traffic sampling engine scales with available resources and works with broadcast traffic, SPAN ports, and encapsulated streams. Any runZero Explorer can sample traffic passively, regardless of its location, configuration, or resources.
- **API Integrations**: runZero has inbound and outbound integrations with major cloud, endpoint, and CMDB providers. You can also build your own with the Custom Integration API and Custom Integration Scripts.

Live inventory

- **Assets**: runZero tracks all assets across the environment: cloud, mobile, endpoint, server, OT, IoT, and everything in between. It correlates and merges assets across data sources into a multi-perspective snapshot of all organization resources. The asset inventory supports deep search, configurable columns, and simple export.
- **Services**: runZero tracks every network service it identifies through active scans, passive discovery, and integrations (where applicable). The services inventory simplifies exposure management tasks and supports deep search and exports.
- **Screenshots**: runZero takes a screenshot of each exposed web service included in active scans. The screenshot inventory lets security teams inspect unknown devices and services by eye.
- **Software**: runZero identifies network-exposed software and imports software records from API integrations. Use the resulting software inventory to find specific packages

and versions across the environment.

- **Vulnerabilities**: runZero reports vulnerabilities based on identified exposures and imports vulnerability data from API integrations. The vulnerability inventory comes in two forms: a detailed per-asset inventory and a grouped view for investigating specific issues.
- **Certificates**: runZero identifies SSL/TLS certificates, including expired and soon-to-expire ones. The certificate inventory includes all relevant metadata, such as issuer, expiration date, and certificate chain.
- **Wireless**: Active scans also enumerate wireless access points within range of the Explorer running the task. The wireless inventory includes the BSSID, SSID, encryption settings, and signal strength.
- **Users and Groups**: runZero imports user and group information from directory services, including Active Directory, Azure Active Directory (Entra ID), and Google Workspace. Search the user and group inventories for accounts with specific attributes, such as expired passwords and excessive group permissions.

Reports

runZero's reports cover everything from layer-2 topology maps to outliers and asset risk. Beyond the pre-defined reports, most attributes of an asset or service can become a grouped report with a single click. For deeper customization, the runZero Export API produces CSV and JSON(L) output with arbitrary search filters, which can feed analytics platforms like Tableau and PowerBI.

Monitoring and alerts

runZero can alert on changes to the inventory, on new results for custom search queries, and on any of dozens of system-level events. Alerts can be delivered in-product, by email, or to a webhook destination, including Slack channels. Alongside alerts, rules, and custom queries, you can define goals and custom dashboard widgets to track progress toward a specific outcome.

To put runZero to work, sign up for an account and follow the getting started guide.

Getting started

To get started, sign up for a runZero account. The default account is a trial of the full runZero Platform. When the trial expires, you can convert to the free Community Edition or buy a subscription.

- Sign up for a runZero account
- See creating an account for activating your account, changing your password, and adding a profile picture.
- With your account set up, pick one of two paths to deploy runZero:
  - Quickstart
  - Full deployment plan

Quickstart guide

Take the quickstart path to use runZero right away. It covers initial setup, running basic scans, and configuring integrations.

1. Set up an Explorer

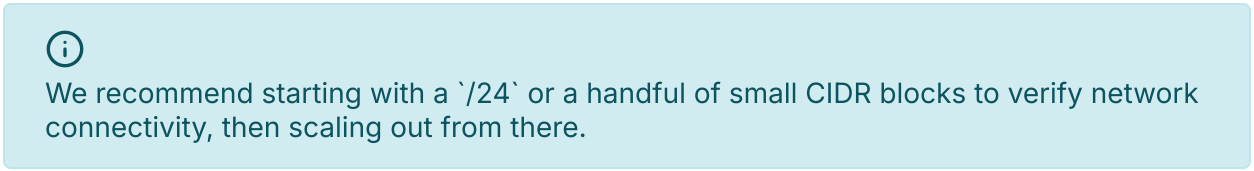
Asset discovery starts with an Explorer:

- Go to Deploy > Deploy Explorers in the runZero Console.
- Review the system requirements and choose the binary for your platform.
- Make sure the Explorer connects to the right organization. Each download link is organization-specific.

2. Run your first scan

With the Explorer installed:

- Go to Scan in the console.
- Click Start standard scan and configure the discovery scope:
  - Choose an Explorer.
  - Set the target subnets or CIDR ranges.
  - Run the scan and monitor progress from the console.



3. Configure integrations

Enrich your data with third-party integrations:

- Go to Integrate.
- Select and configure the supported integrations for EDR, vulnerability management, cloud, and other sources.
  - Each inbound integration has its own documentation.
- Set the required permissions in the integrated systems.
- Run sync tasks to import data and merge it with existing asset information.

4. Review scan results

After the scan completes:

- Check results under Inventory > Assets.

- Use the Query library to spot misconfigurations and other notable findings in the inventory.
- Review reports and customize dashboards to dig deeper.
- A few sample reports to start with:
  - Network Map: interactive 2D/3D topology with attack-path tracing and choke-point ranking.
  - World Map: assets geolocated on a globe by GeoIP, cloud region, and manual `geo*` tags.
  - SNMP default communities
  - SSH authentication methods
  - TLS supported versions



If the `matches:>0` search returns nothing, the metrics may still be calculating. You can check the status of the metrics calculations.

Full deployment plan

If you're planning a production deployment, a little preparation up front gives you a setup that scales. Follow these steps:

1. Plan your deployment

- Set your goals for asset visibility, network segmentation, or compliance.
- Confirm you have adequate resources for Explorers and network access points.
- Learn the runZero-specific terms in the glossary.

2. Initial setup

- Deploy Explorers across your main network segments.
- Configure credentials for access to devices and services (SNMP, EDR, MDM, and so on).
- Verify connectivity against the network communication guidelines.

3. Run discovery and assess

- Run discovery scans on every required segment.
- Validate the data and tune scan settings as needed.
- Find any scanning gaps and adjust configurations.

4. Automate and optimize

- Schedule scans for continuous monitoring.
- Enrich asset data with integrations.
- Set up the alerting and reporting your organization needs.

5. Rollout and monitor

- Deploy to the remaining network segments according to your plan.
- Monitor scan results and network health regularly.
- Document configurations and processes for ongoing management.

If you get stuck, contact support. For step-by-step detail, follow the production deployment documentation.

Creating an account

To get started, sign up for a runZero account. The default account is a trial of the full runZero Platform. When the trial expires, you can convert to the free Community Edition or buy a subscription.

Sign up for a runZero account

Activating your account

After you sign up, we'll email you a link to activate your account. If you don't see it, check your spam folder.

Open the link to reach the **Activation** page and follow the instructions there. You'll provide your name, set a password, specify your location, and accept our privacy policy and terms of service.

Once your account is active, you land directly in the runZero Console. Your new account has administrative access, so you can manage sites, organizations, users, and Explorers.

If anything goes wrong while creating your account, contact support.

Changing your password

Change your password from your account settings. You'll enter your current password before you can set a new one.

All passwords must contain:

- At least 8 characters
- At least 1 uppercase character
- At least 1 lowercase character
- At least 1 number

Updating your profile picture

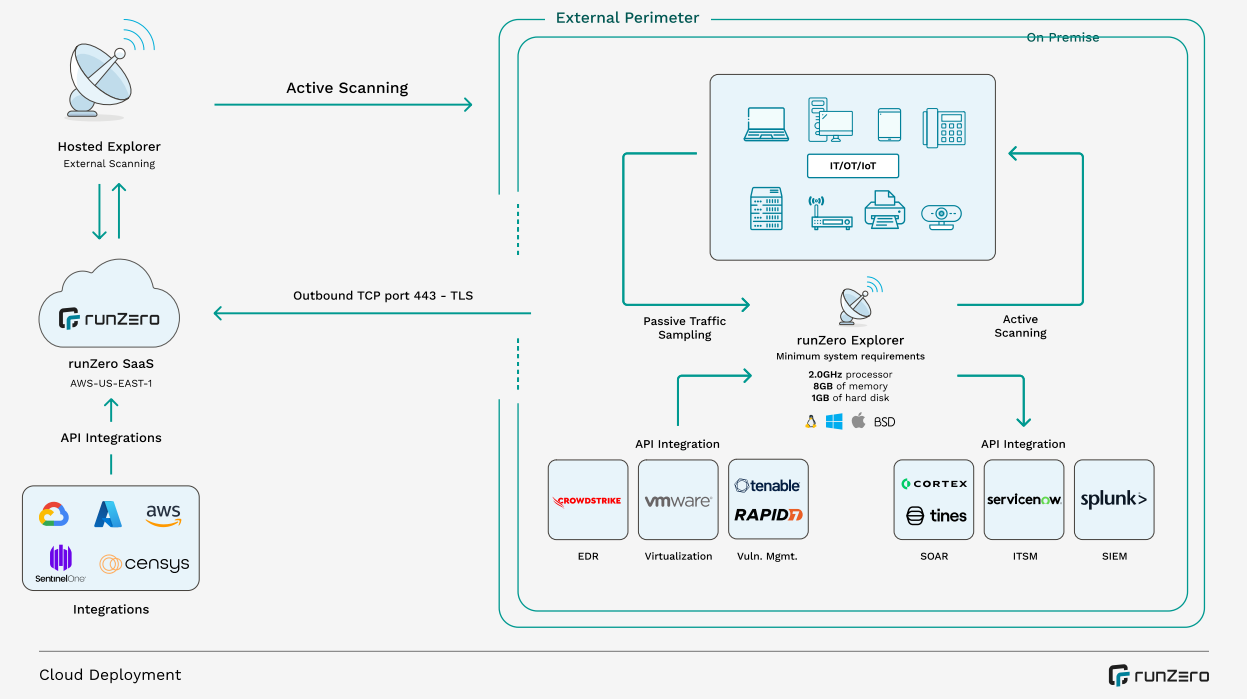
Your profile images come from Gravatar and are tied to your email address. If you don't have a Gravatar account, sign up for one.

Installing an Explorer

runZero needs at least one Explorer in your environment for active and passive network discovery. Install it on a system with reliable connectivity to the network you want to discover. For internal networks, runZero works best on a system with a wired connection rather than wireless.

For external network discovery, nearly any cloud provider with a reliable connection should do. If the Explorer runs in a container or virtualized system, give it direct access to the network (host networking in Docker, bridged networking in VMware, and so on). SaaS customers with a Platform license can use the runZero hosted Explorers at no additional cost.

View an interactive version of this diagram



Installation

Sign in to the runZero Console and switch to the Organization the Explorer should belong to. Then select Deploy in the left navigator and choose the Deploy Explorers sub-menu to reach the downloads.

The download link is specific to your active organization, and **the wrong link can associate the new Explorer with the wrong organization**.

Download the binary for your system from the Explorer download page. For most systems, select the 64-bit (x86_64) architecture. On macOS, pick 64-bit Intel (x86_64) or ARM (Apple M*) to match your hardware. For embedded devices such as the Raspberry Pi 3+, choose the

ARM7 architecture. Windows binaries carry a valid Authenticode signature; validate it before launching the executable.

Installation requires administrative privileges. On Windows, a UAC prompt may appear. On Linux and macOS, make the downloaded binary executable (`chmod u+x runzero-agent.bin`) and run it as root (`sudo` or a root shell). Either way, the Explorer should install itself as a system service, start immediately, and appear as a new entry on the Explorers page. From there, you can manage it as described in Managing Explorers.

System requirements

Windows

- Operating System
  - Windows Server 2016 or newer
  - Windows 10 Build 1604 or newer
- Processor running at 2.0 GHz or faster
- At least 16GiB of memory (8GiB for small environments)
- At least 1GB of free storage space

Limitations

- The Trellix agent for Windows appears to interfere with network scans and traffic capture. Use a system without the Trellix agent for active scans or passive network sampling. Alternatively, switching the Trellix HIDS agent from “block” to “monitor” may allow scans to succeed.

- EDR, antivirus, and endpoint protection products on Windows may also interfere with Explorer scans, packet capture, or process execution. If scans become unreliable, add allowlist exceptions for runZero and npcap/winpcap components, or use monitor-only policy modes while you validate scans. In some environments, you may need to disable these controls on the host for stable operation.

- Windows systems with Intel Advanced Matrix Extensions (AMX) enabled may see Explorer instability or crashes, on physical hardware and virtual machines alike. Disable AMX at the BIOS or host level before installing or running the Explorer. For Windows guests on Hyper-V, enabling **Processor Compatibility** can mitigate this issue.

- Earlier Windows versions may run the Explorer but are not officially supported.

- Windows Explorers run a single scan or passive task at a time because of performance limits in the raw packet driver.

Linux

- Kernel version 3.2 or later
- Processor running at 2.0 GHz or faster
- At least 16GiB of memory (8GiB for small environments)

- At least 1GB of free storage space

Linux ARM devices with limited processing power and memory, such as the Raspberry Pi, can run the Explorer but may struggle to scan larger networks or to run integrations with large amounts of data.

macOS

- macOS 11 (Big Sur) or newer
- Processor running at 2.0 GHz or faster
- At least 16GiB of memory (8GiB for small environments)
- At least 1GB of free storage space

You must use the **curl** download method shown on the deployment page in the Console, because of the Notary requirements introduced in macOS Catalina (10.15).

BSD variants

- Processor running at 2.0 GHz or faster
- At least 16GiB of memory (8GiB for small environments)
- At least 1GB of free storage space

The Explorer requires root access on a recent version of the operating system: FreeBSD 11.2 or newer, or recent versions of NetBSD/DragonFly/OpenBSD.

Configuration



Existing installations may refer to `Rumble` in the directory name, service name, and binary names. `Rumble, Inc` was the previous name of our business; new installations refer only to `runZero`, our new name.

Configure the Explorer with variables in a `config.txt` file in the same directory as the executable. On Windows, create it at `C:\Program Files\runZero\config.txt`; on other platforms, use `/opt/runzero/bin/config.txt`. Besides `config.txt`, the Explorer also supports the legacy `.env` name. If `/etc/runzero/config` is present, it takes precedence.

The file holds one `VAR=VAL` pair per line.

Configuration file locations

Windows: `C:\Program Files\runZero\config.txt`

Other platforms: `/opt/runzero/bin/config.txt`

Web screenshots

The scanner tries any available Chrome, Chromium, or Edge version and reports the browser path and version in the Explorers list in the web console. If it finds no supported browser, install the official Google Chrome package for your operating system to enable web screenshots.

To install the latest Chrome package on Debian-based Linux installations (including Ubuntu):

```
curl -o chrome.deb https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb && \
    sudo apt install ./chrome.deb
```

To install the latest Chrome package on RedHat-based Linux installations (including Fedora, CentOS, Rocky, and Alma):

```
curl -o chrome.rpm https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm && \
    sudo dnf install ./chrome.rpm
```

These variables change web screenshot behavior when set in the environment or the configuration file:

Set `RUNZERO_CHROMIUM_IGNORE_SYSTEM` to `true` to ignore system Chromium paths for screenshots.

Set `RUNZERO_CHROMIUM_AUTOMATIC_INSTALL` to `true` to install Chromium automatically when needed. The download comes directly from the Google and Puppeteer CDNs.

Set `RUNZERO_CHROMIUM_DEBUG` to `true` for detailed Chromium process logging while screenshots are taken.

Set `RUNZERO_CHROMIUM_USER` to a valid system username on non-Windows systems. When the scanner runs as root, the Chromium process runs as this user account (and its default GID).

Set `RUNZERO_CHROMIUM_CACHE_DIR` to the directory for automatic Chromium installations (or installations managed through content updates). It defaults to `/opt/runzero/chromium` on non-Windows and the binary directory plus `\chromium` on Windows. The directory holds a `LATEST.txt` file with a numeric revision, which names a `chromium-[revision]` sub-directory containing a copy of the Chromium installation.

Set `RUNZERO_CHROMIUM_DISABLE_CHROMEDP` to `true` to disable the debug protocol capture mode (now the default).

Set `RUNZERO_CHROMIUM_DISABLE_CHROMECL` to `true` to disable the command-line headless `--screenshot` capture mode (otherwise used as a fallback).

Network communication

The Explorer connects to the runZero Console on TCP port 443 over TLS. This connection carries Explorer registration, job scheduling, status messages, and completed scan jobs. The Console also delivers automatic updates of the Explorer executable. For completely offline environments, the runZero CLI can create scan data files to upload later through the Inventory

`Import` action. The IP addresses involved depend on your deployment model and region. For customers using the SaaS console, these static IPs belong to the runZero consoles.

United States

The console hostname is `console.runzero.com`.

IPv4

- 13.248.161.247
- 76.223.34.198

IPv6

- 2600:9000:a415:cd87:fbe5:476a:3533:69f2
- 2600:9000:a716:ee91:85f9:3c9:48c9:59b9

Germany

The console hostname is `console-eu.runzero.com`.

IPv4

- 15.197.131.232
- 3.33.248.90

IPv6

- 2600:9000:a603:e925:542d:6d40:6897:bc3a
- 2600:9000:a70e:635f:71bd:bb0a:8e43:9466

NAT

Run the Explorer on a system with a full bidirectional network connection, **not** behind a NAT gateway. This includes virtual machines, which should use bridged network adapters.

Juniper DDoS protection is known to break runZero scans.

Proxy support

Some web proxies that perform TLS inspection mishandle Websocket communication, so you'll need to disable TLS inspection for the runZero Explorer to connect. The most common product with this problem is the Sophos (previously Cyberoam) security appliance. Websense users may need to add a bypass rule for `console.runzero.com`.

In most cases, the Explorer handles proxies automatically. On Windows, it reads proxy information from the registry keys used by Chrome, Edge, and IE.

You can also set the proxy with the `HTTPS_PROXY` environment variable. Give `HTTPS_PROXY` a hostname and port (`proxy.example.com:8080`) or just a hostname (`proxy`). The Explorer reads environment variables from your configuration file; the Configuration section shows how to set them. The usual approach is a `config.txt` file in the same directory as the Explorer binary, with the variables in the format described below.

For a SOCKS proxy, set `HTTPS_PROXY` to a `socks5` URL, for example `HTTPS_PROXY=socks5://socks.example.com:1080`.

To override the proxy for AWS connections, set `HTTPS_PROXY_AWS`. It takes priority over any base `HTTPS_PROXY` setting for the Explorer.

The Explorer tries the configured proxy for each probe and, if that fails, tries a direct connection, so both proxy and non-proxy attempts may appear in logs. Using proxies alone to hide or anonymize Explorer connections is not supported.

TLS configuration

You can set the minimum and maximum TLS versions the Explorer uses for outbound communication to the console, through an environment variable or the configuration file. `TLS_VERSION_MIN` and `TLS_VERSION_MAX` accept `1.0`, `1.1`, `1.2`, or `1.3`. The default is a minimum of TLS 1.2 and a maximum of TLS 1.3. If the maximum is set lower than the minimum, the maximum is raised to the minimum. runZero does not recommend TLS versions before 1.2.

This example limits the Explorer to TLS version 1.3.

```
TLS_VERSION_MIN=1.3
TLS_VERSION_MAX=1.3
```

Restarting an Explorer

The easiest way to restart an Explorer is to force a software update from the cloud console. Otherwise, find the service on the host machine and restart it by hand.

On Linux or Mac, run `/opt/runzero/bin/runzero-agent-[uuid] restart`, where `[uuid]` is the ID of the organization the Explorer belongs to.

On Linux, `systemd` can also restart Explorers. First find the name of the Explorer service (runzero-agent-[uuid]):

```
systemctl | grep runzero-agent
```

Then restart the service by that name:

```
systemctl restart runzero-agent-[uuid]
```

A `kill -9` of the Explorer pid should trigger a restart as well.

On macOS, use `launchctl` to restart the Explorer:

```
launchctl kickstart -k runzero-agent-[uuid]
```

As on Linux, `[uuid]` is the organization UUID, visible in the `runzero-agent-*` filename in `/opt/runzero/bin`.

On Windows, restart the Explorer from the Services console like any other background service.

Removing an Explorer

The easiest way to remove an Explorer is from the Explorers page: find the Explorer, then click the *Delete Explorer* trash bin icon under the *Actions* column. This removes the service and terminates the running Explorer process. To remove every online Explorer at once, open the *Manage All Explorers* menu and choose *Uninstall All Online Explorers*. To remove an Explorer without the runZero Console, you have a couple of options.

On Windows, each Explorer appears in Programs and Features (as the runZero Agent) and uninstalls like any other application.

On all platforms, including Windows, the Explorer uninstalls itself when run with the `uninstall` argument from a root or Administrator shell:

Removal on Windows

```
c:\Program Files\runZero\runzero-agent-[uuid].exe uninstall
```

Removal on other platforms

```
/opt/runzero/bin/runzero-agent-[uuid] uninstall
```

Log management

By default, the Explorer logs to a file and to standard output. On Windows, the log file lives in the installation directory (`C:\Program Files\runZero`, or `C:\Program Files\Rumble` for old installations); other platforms log to `/var/log/runzero.log` and `/var/log/runzero.err`. The defaults cap log files at 100MiB, create three backups, and expire logs after 90 days. Change them with these values in the `$BIN/config.txt` file:

- `RUNZERO_AGENT_LOG_MAX_SIZE` sets the maximum log size in mebibytes. The default is **100**.
- `RUNZERO_AGENT_LOG_MAX_BACKUPS` sets how many backup files log rotation creates. The default is **3**.
- `RUNZERO_AGENT_LOG_MAX_AGE` sets the maximum age in days for all files, including backups. The default is **90**.
- `RUNZERO_AGENT_LOG_COMPRESS` controls whether backups are gzip compressed. The default is **false**.
- `RUNZERO_AGENT_LOG_STDOUT` controls whether logs go to standard output. The default is **true**. On Linux, this sends logs to the system log when systemd or upstart starts the Explorer.

On macOS, it produces separate logs under “Log Reports” in the Console application when launchd starts the Explorer.
- `RUNZERO_AGENT_LOG_FILE` controls whether logs go to the log file described above. The default is **true**; set it to `false` to stop writing the log file.

Restart the Explorer for these settings to take effect.

Certificate authorities (CAs)

The Explorer validates TLS connections against the system-installed certificate authorities and an internal CA certificate bundle. By default, both the system roots and the bundled roots apply to all secure TLS connections, including Starlark-based custom integrations. Control this with environment variables, set in the `$BIN/config.txt` file or at the system level:

- Set `RUNZERO_TLS_IGNORE_SYSTEM_ROOTCA` to **true** to ignore the system CA roots.
- Set `RUNZERO_TLS_IGNORE_EMBEDDED_ROOTCA` to **true** to ignore the bundled CA roots.
- Set `RUNZERO_TLS_ADDITIONAL_ROOTCA` to the path of a file with additional CA roots in PEM format.

Manual mode

If the Explorer detects no supported system service manager, such as systemd or upstart, it switches to manual mode: it runs in the foreground and replaces and re-executes its own binary as updates arrive. For temporary installations or container environments, pass the “manual” argument:

```
$ sudo ./runzero-agent.bin manual
```

Storage locations

The Explorer installs into `%PROGRAMFILES%\runZero` on Windows and `/opt/runzero/bin` on all other platforms, and it writes temporary files to the operating system’s default locations. You can override these locations in the `config.txt` file (see the Configuration section above); restart or force-update the Explorer service for the change to take effect. Older installations may still use the `Rumble` versions of these directories and `.env` instead of `config.txt` for environment overrides.

On Windows, the temporary file location is the first non-empty value of `TMP`, `TEMP`, or `USERPROFILE`, falling back to the Windows directory. To override it, add an entry like this to `config.txt`:

```
TMP=D:\Storage\runZero
```

On all other platforms, the temporary file location comes from `TMPDIR`, falling back to `/tmp`. To override it, add an entry like this to `config.txt`:

```
TMPDIR=/home/storage/runzero
```

Scans that fail to upload are kept in the Explorer installation directory. Import them into the platform manually or with the runZero CLI’s `scan --import` and `scan --upload` options.

Container installations

The Explorer can run in standard container environments, with three caveats:

1. Explorer binaries update frequently, and the updates include time-sensitive security coverage. On a read-write container filesystem, the Explorer updates itself but reverts to the older state when the container restarts. We recommend rebuilding the container image frequently, or having the container's entry point fetch and run the latest binary for your organization on startup. On a read-only container, no updates apply, and each task is delayed by up to five minutes while the Console tries to update anyway.

2. The container must run as root with full privileges to access the raw network device. The exact network parameters differ by container runtime, but this typically means a bridged, macvlan, or host (not host-only) networking option when you deploy the container. Without raw network access, many runZero features stop working, including ARP scans, IPv6 neighbor discovery, TCP SYN scans, and passive network sampling through SPAN ports.

3. The container needs a persistent `RUNZERO_AGENT_HOST_ID` environment variable, and only one container with that ID should run at a time. Without it, the Explorer re-registers on every container restart, and if several containers share one identifier, they fight over task status and task processing breaks. Set the host ID to a unique 32-character hexadecimal string.

The `openssl` tool can generate a suitable identifier:

```
openssl rand -hex 16
01b0283809b24511929d0b062bd36109
```

A sample Containerfile to edit and use:

```
#
# Sample Containerfile for running the runZero Explorer in a container, with
# screenshot support.
#
FROM debian:stable

WORKDIR /opt/runzero

# Ensure curl is available and install tools for wireless scanning.
#
RUN apt update && apt install -y curl wireless-tools

# Install Chrome for screenshots.
#
RUN curl -o chrome.deb https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb && \
    apt install -y ./chrome.deb

# Set AGENT_URL to be the download URL for your Linux runZero Explorer. To
# find your URL, go to https://console.runzero.com/deploy/download/explorers
# and click on the first URL box to copy it to the clipboard.
```

```
#
ENV AGENT_URL=https://console.runzero.com/download/explorer/DT[uniqueToken]/[versionID]/runzero-agent-linux-amd64.bin

# This ID is used to track the Explorer even if the container is rebuilt.
# Set it to a unique 32 character hex ID. You can generate one via:
#
# $ openssl rand -hex 16
#
ENV RUNZERO_AGENT_HOST_ID=[UNIQUE-ID]

# If you need to set environment variables to change the Explorer behavior,
# you can do so via the ENV directive. Example:
#
# ENV RUNZERO_AGENT_LOG_DEBUG=true

ADD ${AGENT_URL} runzero-agent.bin

RUN chmod +x runzero-agent.bin

# For full functionality the runZero CLI needs to send and receive raw
# packets, which requires elevated privileges.
USER root

# The argument `manual` tells runZero not to look for SystemD or upstart.
ENTRYPOINT [ "/opt/runzero/runzero-agent.bin", "manual" ]
```

This Containerfile works with podman as well as Docker. Because the Explorer needs root privileges, start the container as root. For best results, run it with the `--privileged` option so the Explorer can listen to network traffic.

Automated installations

The Explorer installs itself automatically when run with root or administrative privileges.

On Linux and BSD, automatic installation depends on a supported init service like systemd or upstart. Without one, the Explorer runs in manual mode instead, overwriting and re-executing itself with each update. To deploy automatically on systems without a supported init service, run the Explorer in the background under the `nohup` wrapper.

On Windows, the Explorer installs automatically when run interactively or with the `updater` parameter. Where MSIs are required, the Explorer MSI wrapper deploys an Explorer from the runZero Console or a local mirror.

AWS EC2 installations

The Explorer can run in an AWS EC2 instance, but you'll need several configuration changes to avoid packet loss when scanning.

- DNS resolution from EC2 to the AWS DNS server has a fixed cap of 1024 packets per second that cannot be raised. Work around it by setting a custom list of non-AWS nameservers in the advanced section of the scan configuration.

- Any Security Group without a `0/0` rule triggers connection tracking, which has an undocumented limit on connections per instance type. Avoid it by adding `0/0 allow` for ICMP/UDP inbound and outbound on the Explorer instance, plus a `0/0 allow` for outbound TCP connections. (runZero scans do not currently require inbound TCP. Callback protocols may need it in the future.)
- Overall packet rates have undocumented limits that depend on instance type. Experiment with instance sizing until scans are consistent at a given scan rate. We hope to gather and share data on appropriate instance sizes soon.

The Explorer should work well on a memory optimized, compute optimized, or general compute instance. Because it can make full use of multi-core systems, you may want to match the number of cores to the number of simultaneous scans. You probably won't want to run larger scans on an instance with less than 32GiB of RAM.

Automated MSI deployments

The runZero Explorer downloads are dynamically generated binaries, which don't always play well with MSI-based installation methods.

To work around this, we provide a shim MSI package for use with automated installers. The package has a valid Authenticode signature, and you can also verify it with the runZero Verifier.



Some parts of the application still use the name "Rumble" for backwards compatibility. New installations use runZero for directory, file, and user names.



Windows hosts with Intel Advanced Matrix Extensions (AMX) enabled may see Explorer instability or crashes, on physical hardware and virtual machines alike. Disable AMX at the BIOS or host level before deployment. For Windows guests on Hyper-V, enabling **Processor Compatibility** can mitigate this issue.



EDR, antivirus, and other endpoint protection software on Windows may interfere with Explorer installation or scan behavior. If problems occur, add allowlist exceptions for runZero and npcap/winpcap components, or use monitor-only policy modes while you validate deployments. In some environments, you may need to disable endpoint protections for stable operation.

Deploy the package with the `URL` parameter set to the organization-specific download URL from the Explorers section of the runZero Console.

```
msiexec /i runzero-explorer-installer-amd64.msi URL=https://console.runzero.com
/download/explorer/DT[uniqueToken]/[versionID]/runzero-explorer-windows-amd64.exe
```

Enter this command as one long line with a single URL parameter; it's wrapped here for display.

The MSI shim verifies that the URL points to a valid runZero Explorer binary, then installs it normally.

Warning: Installing with the `/a` parameter does not work; use `/i` instead.

Binary downloads

Build	SHA256
runZero Explorer Installer MSI x86 64-bit	sha-256
runZero Explorer Installer MSI x86 32-bit	sha-256

Installing on a Raspberry Pi

The runZero Explorer runs discovery scans, and in most cases you can deploy it on an existing system with connectivity to the network you want to discover. Some locations, like retail stores or customer sites, may have no staff or hardware available to install one, which makes remote deployment tricky.

For those sites, install a runZero Explorer on a Raspberry Pi and send the device to the location for them to plug into their network.

What you'll need

- Raspberry Pi 4 Model B (4GB or 8GB), Raspberry Pi Compute Module 4, or Raspberry Pi 400
- At least 1GB of free storage space on your Raspberry Pi's MicroSD card after installing the operating system

Connecting to your Raspberry Pi via SSH

These steps show you how to SSH to your Raspberry Pi and install the Explorer from your terminal.

Step 1. Enable SSH on your Raspberry Pi

Enable SSH on the device before you try to connect.

You can add a file called ssh.txt at the root of the SD card and reboot the Raspberry Pi. The file can be empty. On reboot, the Pi enables SSH if the file is present.

Step 2. Update the password for your Raspberry Pi

With SSH enabled, you must update your credentials.

Log in as the `pi` user and run `passwd` to change the default password. The `passwd` command asks for your current password before it lets you set a new one.

Step 3. Copy the Explorer instructions

- Go to your console and verify you're in the right organization. runZero keys the Explorer download link to the organization you're viewing, which associates the Explorer with that organization.
- Go to the Explorer deployment page, select **Linux Distributions**, and choose Linux ARM 32-bit V7. Raspbian uses a 32-bit kernel by default, even on 64-bit Raspberry Pi hardware.
- Click the instructions at the bottom of the Linux installation page to copy the commands to your clipboard.

Step 4. Install the Explorer

> Some parts of the application still use the name "Rumble" for backwards compatibility. We'll update the documentation as they change.

Connect to the Raspberry Pi over SSH and paste the installation command into the terminal. The URL in the command ties the installation to your active organization, so yours will differ from the example below.

```
me@mac ~ % ssh pi@hostIP
pi@host's password:
pi@raspberrypi:~ $ sudo bash
root@raspberrypi:/home/pi# cd Downloads
root@raspberrypi:/home/pi# curl -o runzero-explorer.bin https://console.runzero.com/download/explorer/DT[uniqueToken]/|
```

The Explorer installs itself in `/opt/runzero` and sets up a systemd service named `runzero-agent-`, which starts automatically on boot.

Next steps

Your Raspberry Pi is now a runZero Explorer box, ready to ship wherever you need it.

You can view and manage your registered Explorers from your console at any time, and Managing Explorers covers what you can do there.

Managing Explorers

The runZero Explorer is a lightweight scan engine for network and asset discovery. You should have at least one Explorer deployed, and once you do, you manage your Explorers from the Deploy page in your runZero web console.

Viewing all Explorers

For each Explorer, you can see:

- The Explorer status (whether it's communicating with runZero)
- The OS it runs on
- Its name
- Any site it's associated with
- Its IP addresses
- The software version it runs
- Whether the installed npcap version is up to date, if the OS is Windows (see upgrading npcap below)
- The CPU architecture of the host machine
- Any tags associated with the Explorer
- The status of its last scan
- Its capabilities, like Chrome support

Screenshot capabilities

To capture screenshots, the Explorer needs Chrome installed. To check whether an Explorer has screenshot capabilities, look for the Chrome icon in the *Capabilities* column:

- Green icon: the Explorer has access to a Google Chrome binary and can take screenshots.
- Red icon: no suitable Chrome binary was found.

If the icon is red, see Explorer not capturing screenshots.

Searching for Explorers

Use the search bar to find Explorers. The query syntax is similar to the other search bars in runZero, with keywords to filter by specific fields:

Keyword	Search by	Example
`arch:`	CPU architecture	`arch:amd64`
`name:`	assigned name	`name:scanner.local`
`address:`	IP address	`address:10.0.1.200`

`capability:`	capabilities	`capability:screenshot` or `capability:aws`
`tag:`	assigned tag	`tag:dev`
`npcap_version:`	npcap version	`npcap_version:1.31`

Explorer actions

Each Explorer has a set of action buttons:

- **Reinstall an Explorer**: runs a reinstall or upgrade of the Explorer. The current Explorer downloads the latest Explorer code from runZero, then runs the install process.
- **Configure an Explorer**: associates the Explorer with a specific site, enables recurring speed tests, and adds tags to it. You can also set the maximum number of concurrent scans allowed; a single Explorer can run multiple tasks at once.
- **Reassign an Explorer**: moves the Explorer to a different organization within your account, or even to a different runZero client account entirely.
- **Remove an Explorer**: asks a running Explorer to uninstall itself from the host machine. If the Explorer is not running, you can still tell runZero to forget it, which is useful when you've decommissioned the machine it ran on or uninstalled it manually. If a forgotten Explorer runs again, runZero adds it back to the registered Explorers list.

Bulk management operations

Bulk operations act on many Explorers at once. They live in the *Manage all Explorers* menu:

- **Update all online Explorers**: tells every Explorer that is up and communicating with runZero to upgrade its software.
- **Forget all offline Explorers**: clears every Explorer that is currently offline and makes runZero forget it. No data is lost, and any Explorer that comes back online rejoins the active list.
- **Uninstall all online Explorers**: tells every online Explorer to uninstall itself from its host system.
- **Automatically assign sites**: goes through every Explorer not currently assigned to a specific site and checks its IP address against the CIDR ranges of the registered subnets of all sites in the current organization. If the address matches exactly one site, runZero assigns the Explorer to that site.

Viewing Explorer details

Click an Explorer's name to open a page of diagnostic information for that Explorer, including its software version, available memory, and network interfaces.

At the bottom of the page is a diagnostics text area. Click *Update Diagnostics* to fetch a fresh list of all sub-processes active within the Explorer. This is useful to send to runZero support when a particular Explorer is giving you trouble.

Internet speed tests

Explorers can run internet speed tests on recurring hourly intervals or on demand. The "Internet speed tests" table shows the most recent results, which can help you monitor the network performance of the Explorer's host machine over time.

To run one now, click the "Start speed test" button; it runs a single speed test and updates the table with the results.

To configure recurring speed tests:

1. From the **Registered Explorers** page, select the Explorer that should run speed tests.
2. In the "Explorer details" card, click the "Edit" button and enter how often the speed test should run, in hours, into the "Speed tests" input.
3. Click **Save** to confirm your configuration and start recurring speed tests at that interval.

Speed tests need an up-to-date Explorer. If yours is behind, click the "Update Explorer" button to bring it to the latest version, then configure the tests.

To download the results table in CSV, JSON, or JSONL format, click the "Download speed test results" button and pick the file type. Every speed test also generates a `speedtest-completed` event in the runZero audit log with its results.

Traffic sampling

You also configure traffic sampling from the Explorer details page.

1. From the **Registered Explorers** page, select the Explorer that should perform traffic sampling.
2. In the traffic sampling card, configure these options:
    - **Site**: the site that assets discovered through traffic sampling are added to.
    - **Discovery scope**: the IP addresses or CIDR networks that traffic sampling observes on this Explorer.
    - **Asset tags** (optional): the tags to apply to assets discovered through traffic sampling.
    - **Excluded hosts** (optional): the IP addresses or CIDR networks that traffic sampling leaves out of the results.
    - **Interfaces**: toggle on the interfaces this Explorer should listen on.
3. Click **Save** to store your configuration and start the traffic sampling task.

To disable traffic sampling later, return to this page, toggle off the selected interfaces, and save. The traffic sampling tasks stop automatically.

Explorer groups

An Explorer group is a logical grouping of Explorers that distribute scheduled tasks among themselves. With a group, you can schedule many tasks without managing which Explorer each one runs on.

Create and manage groups from the Explorer groups page. A new group needs a name, description, and site, plus the Explorers you want in it. Every Explorer in the group must be set to the same site as the group, because any Explorer in the group might run a given task; without that rule, assets could be merged into the wrong site.

To use a group, select it when configuring a scan or integration task, and the group elects the best-fit Explorer for the task.

Upgrading npcap

On Windows, runZero uses a licensed third-party library called npcap for access to raw network traffic. Other software on the Explorer's host machine may also use npcap, and sometimes it has installed an obsolete version, which can cause reliability problems.

runZero flags an obsolete npcap with a warning icon in the list of Explorers.

runZero can't yet reliably upgrade npcap for you, though. Because npcap/winpcap tends to be shared between applications, forcing an upgrade from the runZero side can break other services (EDRs, Wireshark, and so on).

To upgrade npcap manually:

1. Stop any running runZero services from the Windows Services app. Look for "runZero Network Discovery Explorer".
2. Stop any other running software that uses npcap.
3. Uninstall Winpcap and any npcap installations from the Windows Control Panel.
4. Reboot the computer.

runZero restarts automatically and installs the latest npcap.

Verifying binaries

runZero generates the Console, CLI, and Explorer download binaries dynamically. Dynamic binaries make it easy to deploy Explorers that connect back to the right organization, but they complicate independent integrity validation. Windows binaries have a valid Authenticode signature, and every binary also carries a secondary, internal signature. The **runZero Verifier** checks that signature: it confirms whether a given binary contains a valid internal signature, in addition to any existing Authenticode signatures.

Download the latest verifier from the bottom of this page, along with the PGP signature file for your architecture.

The runZero Verifier is always signed by PGP Key ID `60EBAAE9AEF08C6D`.

To validate the signature of the Windows 64-bit Verifier, you need a GPG client. Then run:

```
C:\> curl -s https://www.runzero.com/.well-known/security.pub.asc | gpg --import
C:\> gpg --verify runzero-verifier-3.1.0-windows-amd64.exe.asc
```

A successful validation shows a valid signature by key ID `9B5DAFF7D43349298A3039BD60EBAAE9AEF08C6D`.

```
gpg: Signature made Sun 07 Aug 2022 11:33:15 AM CDT
gpg:                using RSA key 9B5DAFF7D43349298A3039BD60EBAAE9AEF08C6D
gpg:                issuer "security@runzero.com"
gpg: Good signature from "runZero Security <security@runzero.com>" [unknown]
```

This warning is expected and does not mean the signature has a problem:

```
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
```

With the Verifier itself validated, use it to check the signature of any runZero binary:

```
C:\> runzero-verifier-3.1.0-windows-amd64.exe runzero-explorer-3.1.0-windows-amd64.exe
runzero-explorer-3.1.0-windows-amd64.exe: VALID SIGNATURE
```

A failed validation shows the error `Invalid or missing signature`, and the verifier exits with status 1.

Binary downloads

Windows

Build	PGP sig	SHA hash
runZero Verifier x86 64-bit	pgp signature	sha-256
runZero Verifier x86 32-bit	pgp signature	sha-256

Linux

Build	PGP sig	SHA hash
runZero Verifier x86 64-bit	pgp signature	sha-256
runZero Verifier x86 32-bit	pgp signature	sha-256

Additional Linux builds

Build	PGP sig	SHA hash
runZero Verifier ARM v5 32-bit	pgp signature	sha-256
runZero Verifier ARM v6 32-bit	pgp signature	sha-256
runZero Verifier ARM v7 32-bit	pgp signature	sha-256
runZero Verifier ARM 64-bit (aarch64)	pgp signature	sha-256
runZero Verifier PPC 64-bit Little Endian	pgp signature	sha-256

runZero Verifier MIPS 32-bit Big Endian	pgp signature	sha-256
runZero Verifier MIPS 32-bit Little Endian	pgp signature	sha-256
runZero Verifier MIPS 64-bit Big Endian	pgp signature	sha-256
runZero Verifier MIPS 64-bit Little Endian	pgp signature	sha-256
runZero Verifier S390X	pgp signature	sha-256

macOS

Build	PGP sig	SHA hash
runZero Verifier x86 64-bit	pgp signature	sha-256
runZero Verifier ARM 64-bit	pgp signature	sha-256

BSD variants

FreeBSD

Build	PGP sig	SHA hash
runZero Verifier x86 64-bit	pgp signature	sha-256
runZero Verifier x86 32-bit	pgp signature	sha-256
runZero Verifier ARM v6 32-bit	pgp signature	sha-256
runZero Verifier ARM v7 32-bit	pgp signature	sha-256

NetBSD

Build	PGP sig	SHA hash
runZero Verifier x86 64-bit	pgp signature	sha-256
runZero Verifier x86 32-bit	pgp signature	sha-256
runZero Verifier ARM v5 32-bit	pgp signature	sha-256
runZero Verifier ARM v6 32-bit	pgp signature	sha-256
runZero Verifier ARM v7 32-bit	pgp signature	sha-256

Dragonfly

Build	PGP sig	SHA hash
runZero Verifier 64-bit	pgp signature	sha-256

OpenBSD

Build	PGP sig	SHA hash
runZero Verifier 64-bit	pgp signature	sha-256

Full-scale deployment

Start with the standard deployment plan and add tasks as you need them. It's broken into six stages that help you plan your requirements, execute the deployment, and optimize your environment around runZero's best practices.

1. Identify key success outcomes

Total attack surface visibility

- Active discovery on all internal assets
- Active discovery on all externally facing assets
- Passive discovery and enrichment in key network segments
- Integrate with all cloud providers and other relevant data sources

Additional resources

- Overview
- RFC1918 coverage playbook
- Scanning OT playbook

Full-spectrum exposure detection

- Rapid Response findings and asset-level pivoting
- Network misconfiguration findings and control coverage gaps
- Vulnerability enrichment and inside-out findings

Additional resources

- Overview
- Rapid Response blog
- Gaps in EDR playbook
- Gaps in VM playbook

Risk prioritization and insights

- Custom dashboards
- Rules and alerts
- Setting asset criticality and ownership

Additional resources

- Overview
- Custom dashboards documentation
- Alerting playbook

Compliance, reporting, and KPIs

- Comply with asset inventory and discovery requirements of relevant frameworks
- Comply with secure configuration requirements of relevant frameworks

- Comply with malware protection requirements of relevant frameworks
- Comply with vulnerability management requirements of relevant frameworks

Additional resources

- Overview
- Compliance alignment documentation
- NYDFS dashboard playbook

2. Planning your deployment

These first tasks help your team identify target results, get ahead of blockers, and avoid misconfiguring runZero.

Tasks

- Identify the organizational stakeholders
  - Administrators (who will set up runZero)
  - Integration owners (who will provide credentials for each integration)
- All users take the runZero 101 training
- Administrators take the runZero 201 training
- Decide whether you need to self-host (docs | video)
- Decide whether users will sign in with SSO or local accounts.
  - If SSO, start any internal process needed to get the runZero app created in your provider.
- Identify known networks and subnets for discovery, plus any other inventory sources (docs | video)
  - If you have them, prepare CSV files to bulk import subnets.
  - Configure firewalls to let the Explorer IPs scan the entire network on all ports.
- Define organizations based on your RBAC requirements, including organization-level data retention and expiration thresholds (docs | video).
- Decide where to deploy Explorers.
  - Explorer Groups let you combine Explorers logically to simplify task scheduling.
- Identify the inbound integrations you need. Prioritize connectors for platforms such as EDR and CSPM to get full vulnerability and misconfiguration coverage.
- Get the team familiar with runZero search so they can plan initial asset searches and reporting requirements.

3. Initial configuration

Once your plan is in place, configure runZero and run your initial scans. These tasks are in priority order, so working through them in sequence saves you from reconfiguring later.

Tasks

- Deploy the self-hosted console, if you need one (docs | video)

- Set up organizations and configure expiration thresholds for stale data, integrations, and vulnerabilities.
- Set up sites and define subnets for discovery (video)
    - **Sites do not necessarily correspond to physical locations within runZero.** Sites represent distinct networks, which may have overlapping IP space.
    - This also includes public IP space and domains for external scanning.
- Install Explorers (video)
    - If planned, configure Explorer Groups to organize scanners logically and simplify task distribution.
- Define private network addresses
    - If you use public IP ranges internally, set custom private IP ranges in Account Settings.
- Run your initial scan (docs | video)
- Configure credentials and inbound integration connections
    - Cloud inventory sources
    - Endpoint protection
    - Vulnerability management
    - Directory services
    - Custom integrations

4. Review assets and exposures

With initial discovery done and integrations connected, review the results. Paired with runZero's reporting features, that review helps you expand scan scope, prioritize risk, understand your network better, and identify exposures such as misconfigurations and actively exploited vulnerabilities.

Tasks

- Review results and exposure overview
    - Review the **Risk Management Dashboard** (the new default dashboard) for a centralized view of risk, with insights, trend data, and asset breakdowns (docs).
        - Check the **Latest Rapid Response alerts** widget, a carousel of the five most recent Rapid Response posts that shows the matches in your inventory (docs).
        - Create and share multiple custom dashboards for different use cases or teams (e.g., Compliance or Vulnerability Management) (docs).
    - Review the Asset Inventory (docs). It correlates and merges assets across all data sources (scan, passive, and integrations) into a single source of truth.
    - Review the Asset Detail View, which consolidates each asset's attributes, vulnerabilities, and software (docs).
- Identify and prioritize exposures with Findings
    - Review the **Findings** section. It groups vulnerabilities and misconfigurations into prioritized risk categories (e.g., **Internet Exposure**, **Open Access**, **End-of-Life**, **Certificates**, **Vulnerability**, and **Best Practice Violations**) (docs).

- Dig deeper with queries
    - Find risky assets with the Queries library (docs):
        - Learn the query syntax (docs).
        - Apply vulnerability records to queries (e.g., for novel internal findings) (docs).
        - Filter assets and services by their **Finding Code** (e.g., `finding_code:rz-finding-internet-exposed-database`) to target remediation (docs).
        - Search for assets exposed to **Known Exploited Vulnerabilities (KEV)** lists (CISA KEV, VulnCheck) and use **EPSS scores** to set priority.
- Track long-term initiatives
    - Use Goals to track them (docs).
    - Use **Baseline Goals** to measure progress against specific inventory subsets (e.g., a goal to remediate all expired certificates or critical vulnerabilities in the `os_eol:<now` subset).
- Review reporting
    - Review reports (docs).
    - Identify gaps in scanning using reports (docs).
    - Understand network segmentation (docs).

5. Optimization

After your initial analysis, optimize your scans and configuration to follow best practices.

Tasks

- Configure SNMP credentials (video)
- Optimize scans by adjusting scan rates and other settings (docs | video)
    - This clickthrough covers more of the configuration options
- Confirm that default credential and vulnerability checks are enabled on all scans.
    - The Templates page lists current coverage.
- Configure asset ownership to speed up investigations.

6. Automation

Now that your scans are optimized and your data analyzed, you can automate the routine work: recurring scans, automated queries, and alerts for the team.

Tasks

- Schedule recurring scan tasks and any inbound integration tasks
- Automate queries and configure alerts to match your use cases (video)
- Use rules to automate tagging and asset criticality (walkthrough)

- Set up alerts for system events such as an Explorer going offline or a scan failing.
- Configure outbound integration connections to enrich other IT and security tools
  - CMDB
  - SIEM
  - SOAR
  - If runZero currently has no standard outbound integration for a tool you use, the API documentation explains how to export runZero data.

7. Rollout

As the deployment wraps up, confirm that every user has completed training and that everyone who would get value from runZero has access to the platform.

Tasks

- Add users
- Confirm all users are trained on runZero
  - Training and core documentation
    - 101 user training
    - 201 administrator training
    - Use case library
    - Search
    - Reporting
    - Exporting assets
  - runZero playbooks
- Identify other teams with an interest in the asset inventory data, such as:
  - Enterprise security team
    - Security teams typically use runZero to build a complete asset inventory, find gaps in their vulnerability scanning and endpoint protection, and discover potential vulnerabilities.
  - IT Operations team
    - IT Operations teams typically use runZero to build a complete inventory of all assets across on-premises and cloud infrastructure, then turn to search and reporting to identify misconfigurations and report on those assets.
  - Penetration testing team
    - Penetration testing teams typically use runZero to run internal and external reconnaissance, identify vulnerable targets, and find paths to those targets with search and reporting.

Additional resources

After deployment and training, these resources can answer the questions you still have and help you get more out of runZero:

- runZero playbooks
- Use case library
- API documentation
- Glossary

- runZero FAQs

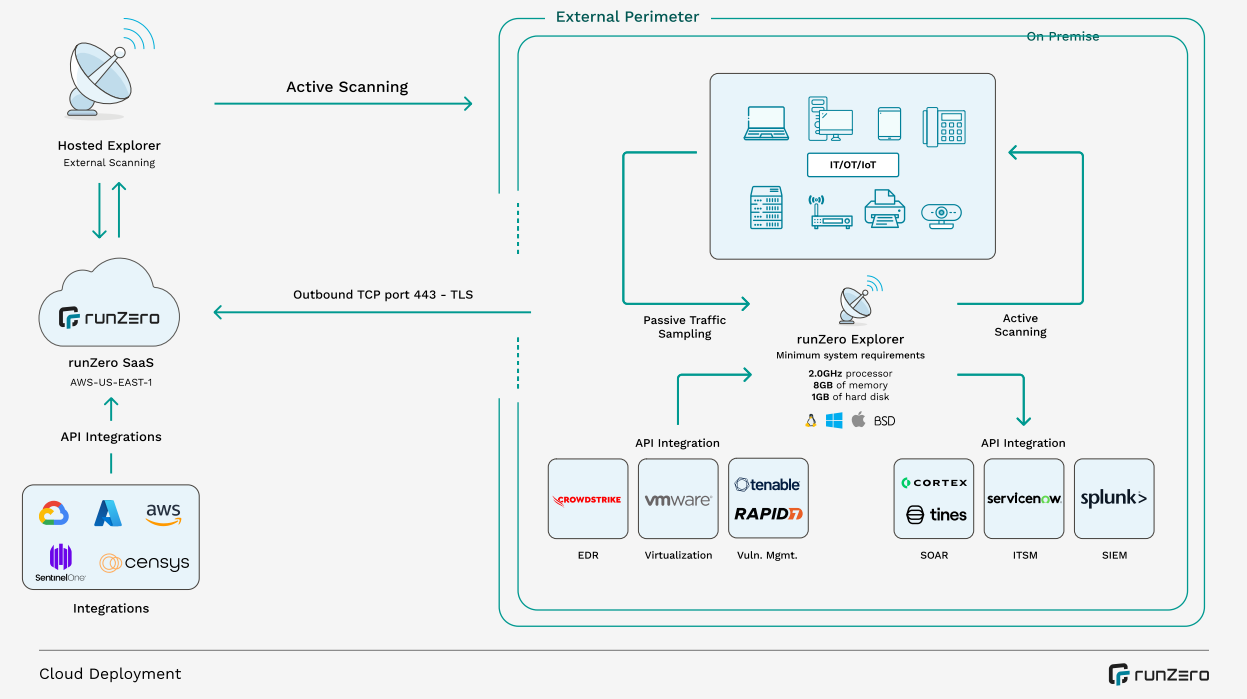
Getting help

If you need help at any point in the process, book a session with a runZero Customer Success Engineer.

Types of networks

These network examples give you a starting point for planning your runZero implementation. Each one describes a standard network type and a potential configuration for it. Every network has its own nuance, so expect your implementation to differ in places.

Discovery runs through Explorers and scanners. By default, you deploy one Explorer to run discovery on as much of the network as possible and, if needed, add more for areas the primary Explorer cannot reach. You can also use a scanner for offline environments with no internet connectivity. Once you've matched your network to one of these models, follow the standard deployment plan to roll it out.



SaaS company

Network characteristics

- Flat on-premises network for offices
- Multi-cloud environment
- Hybrid workforce, remote and in office

Sample runZero implementation

- Explorers
  - Corporate network Explorer with access to all on-premises networks and private connectivity to cloud services
  - One Explorer per VPC that isn't routable from on-premises networks
- Organizations
  - Single organization
- Sites
  - Primary corporate site
  - One site per VPC

Large corporation

Network characteristics

- Corporate offices with many connected branches
- Multiple M&A transactions in flight at any time, onboarding new offices every year
- Multiple data centers for internal and externally facing applications

Sample runZero implementation

- Explorers
  - Corporate network Explorer that can reach all on-premises networks
  - One Explorer per site with low bandwidth or legacy firewalls, proxies, etc.
  - One Explorer per M&A transaction to assess new risks and keep data segmented
- Organizations
  - Corporate assets
  - One project per M&A transaction until the deal closes and the assets are merged
- Sites
  - Primary corporate site
  - Possibly separate sites if branch offices have overlapping IP space

Retail company

Network characteristics

- Flat on-premises network for corporate offices
- Data centers
- Hundreds of retail locations

Sample runZero implementation

- Explorers
  - Corporate network Explorer that can reach all on-premises networks

    - One Explorer per retail location with low bandwidth or legacy firewalls, proxies, etc.
    - One Explorer per data center
- Organizations
    - Single organization
- Sites
    - Primary corporate site
    - One per retail site if it has overlapping IP space

Manufacturing company

Network characteristics

- Flat on-premises network for corporate offices
- OT environment completely disconnected from the internet
- Field service agents VPN in from varying locations

Sample runZero implementation

- Explorers
    - Corporate network Explorer that can reach all on-premises networks
- Scanners
    - Manufacturing plant with no connection to the corporate networks
- Organizations
    - Single organization
- Sites
    - Primary corporate site
    - One per OT site if it has overlapping IP space

Telecommunications company

Network characteristics

- Flat on-premises network for corporate offices
- Segmented data centers
- IoT devices scattered across the country
- Field service agents VPN in from varying locations

Sample runZero implementation

- Explorers
    - Corporate network Explorer that can reach all on-premises networks
    - One for each segmented lab
    - One for each data center
- Organizations
    - Corporate network
    - Production network

    - One per lab
- Sites
    - One primary site per organization

Managed security service provider

Network characteristics

- Many customers spinning up and down across engagements
- Mostly small, flat networks
- Some customers with sensitive OT environments

Sample runZero implementation

- Explorers
    - One per customer
- Scanners
    - Only for customers with offline networks
- Organizations
    - One project per customer to start
    - Promote the project to an organization for long-term use
- Sites
    - One per customer, or more if a customer has overlapping IP space

Academic organization

Network characteristics

- Many buildings and networks spread around the campus with varying connectivity
- Multiple data centers managed by different departments
- Multiple research and development labs disconnected from the rest of the network and from the internet

Sample runZero implementation

- Explorers
    - Main network Explorer that can reach all on-premises networks
    - One Explorer per building with low bandwidth or legacy firewalls, proxies, etc.
    - One Explorer per data center
- Scanners
    - One for each disconnected lab
- Organizations
    - Main assets
    - One for each lab
- Sites
    - One primary site per organization, or more if buildings have overlapping IP space

Organizations

An organization is a distinct entity: your business, a specific department within it, or one of your customers. All actions, tasks, Explorers, scans, and other objects that runZero manages belong to a specific organization and are isolated from each other.

Switch your active organization with the dropdown at the top right of the runZero Console. If your default role is Viewer or higher, select **All Organizations** from the dropdown to see the data for all of your organizations in the Dashboard and Inventory. The Queries page also supports the All Organizations view. Pages that do not support it stay hidden until you select a single organization.

runZero Community Edition is limited to a single organization.

Use cases for organizations

It's common to segment your assets across multiple organizations for one of these reasons:

- **RBAC** limits users to the assets in the organizations they're added to, and in those organizations' children.
- **Environments** such as development and production can stay in separate pools.
- **Customers** of a service provider generally get one organization each.

Creating organizations

Create, modify, and deactivate organizations from the organizations section of the console: click the Organizations link under Global Settings.

Organization configurations

- **Name** is the name of the organization.
- **Parent organization** makes this a sub-organization, with nesting limited to three levels. Permissions are inherited based on most privilege.
- **Description** describes the organization.
- **Stale asset expiration** sets the number of days before stale assets are removed from the inventory. It applies to online and offline assets not seen within that many days.
- **Offline asset expiration** sets the number of days before offline assets are removed from the inventory. It applies to offline assets not seen within that many days.
- **Stale integration attributes expiration** sets the number of days before stale integration attributes are removed from the inventory. You can optionally keep the latest set of integration attributes per source, even when it is older than the threshold.
- **Stale vulnerability expiration** sets the number of days since last seen before stale vulnerabilities, including archived ones, are permanently removed from the inventory.
- **Enable vulnerability archiving** (default: enabled) applies to all vulnerabilities being removed from the active inventory. When disabled, vulnerabilities are deleted instead of archived. Archived metadata (such as suppression state and the original first seen date)

is not retained if the vulnerability returns. Disabling this setting disables `Stale vulnerability archive`.
- **Stale vulnerability archive** (default: 0, disabled) sets the number of days since last seen before stale vulnerabilities are archived and removed from the active inventory. Set it lower than the stale vulnerability expiration.
- **Scan data expiration** sets the number of days before completed scans are removed from the platform. It applies to the scan task metadata as well as the raw data files and change reports.

Projects

A project is a special type of organization for temporary use. Projects behave like organizations and can have sites defined within them. The differences: projects cannot be parents of organizations or other projects, they automatically become read only after 30 days, and they are automatically deleted after 90 days. Organizations let you customize expiration times; project expiration times are fixed.

Your runZero license includes up to five times as many project assets as licensed live assets. For example, a license for 1,000 live assets allows up to 5,000 project assets in total. The license information page in the runZero console shows the total number of assets in organizations and projects.

If you decide to keep a project indefinitely, convert it into an organization.

Organizations support custom data retention settings that you can edit to fit your requirements.

Organization and project details

Click the name of an organization or project to open its details page. It shows basic details about the organization or project, along with its place in the organization hierarchy.

Sites

Every organization has at least one site and may have several. A site is a distinct network segment, usually defined by addressing or accessibility. Sites in runZero do not have to match physical sites or locations. They represent distinct networks that may have overlapping address space. Multiple sites can use the same RFC1918 space, which is common in retail, and their assets stay distinguishable in the inventory.

By default, your account has a single organization, which contains a single site, named Primary. If you delete the only site in an organization, runZero creates a replacement automatically, and it does the same if you remove the last organization. You can rename organizations and sites at any time.

Because sites represent separate networks, scanning the same devices from two sites gives you two copies of the resulting assets, one set for each site's network.

All analysis in runZero happens at the site level. The Network Map, for example, builds its topology from a single site's devices, so you'll likely want to avoid splitting routers and non-router assets into separate sites unless they are truly on separate, disconnected networks.

Use cases for sites

For flat networks, where every IP address can reach any other address on the network, a single site is usually enough, and it keeps you from accidentally creating duplicate assets by scanning the same devices from multiple sites. Sites are recommended for complicated, sprawling, and highly-segmented environments.

Two situations can call for multiple sites:

- With overlapping IP space, sites let you distinguish identical IPs that belong to different machines.
- In a highly complex network, you can use sites as an organizational tool, though they are not required.

Creating sites

Site creation must be enabled on your account before you can add new sites. If you do not see a **New Site** button, email support@runzero.com to have it enabled.

To create a site, click the **New Site** button at the top of the sites page.

Site configurations

- **Name** is the name of your site.
- **Description** can help identify the purpose of the site.
- **Default scan scope** is pre-populated when you create scans for this site.
- **Default scan exclusions** are pre-populated when you create scans for this site.

- **Registered subnets** can automatically tag assets, services, screenshots, and software that fall within each subnet.

Subnet tagging

Tagging by subnet works a little differently from tagging assets directly or through a task.

When a subnet carries a specific tag, only assets in that subnet get it. runZero removes the tag from any asset outside the subnet, even if a task or a user set it.

Importing subnets

You can create or update site subnets by importing a CSV file as part of the site configuration. A sample CSV for import:

```
cidr,tags,description
192.168.0.0/16,purpose=endpoints,"Headquarters endpoint subnet ""quoted text"""
10.0.0.0/24,purpose=wifi,"Headquarters wireless subnet, with comma"
2001:db8::/32,"foo=bar,foobar",ipv6 subnet
1.2.3.4/16,,
```

Importing and exporting sites

You can create or update site configurations by importing a CSV file. Import your CSV from the sites page. The CSV format should include these columns:

```
name,description,scope,exclusion,subnet_ranges,subnet_tags,subnet_descriptions
```

You can also export your site configurations as a CSV from the sites page.

Sites and Explorers

You can tie sites to specific Explorers, which can help limit traffic between low-bandwidth segments. The site configuration lets you define a default scan scope and an optional list of excluded scan scopes. Scans of the site can use these fields as their scope.

To tie an Explorer to a site, go to the Explorers page, click the Explorer you want, and then click configure. A dropdown with your site options appears. For more on Explorers, see Managing Explorers.

Self-hosting runZero

Background

The self-hosted version of runZero runs the entire platform on-premises or in your own cloud environment. It is functionally identical to the hosted service, has a fully-offline mode, and sends no inventory data back to runZero.

Self-hosting is less common, but your company might choose it for a few reasons:

- ISO compliance requirement
- Other compliance requirement
- Prefer data on-premise

Self-hosting must be enabled for your account. It's available with a runZero Platform license or during an initial trial for a platform license, but not with the free Community Edition of runZero. Contact your runZero sales representative for details.

Requirements

Check your system against these requirements before you install.

Hardware requirements

Recommended production system requirements:

- 12 CPU Cores at 2 GHz or faster
- 1TB of local disk storage
- 128 GB of RAM

Minimum production system requirements:

- 4 CPU Cores at 2 GHz or faster
- 100 GB of local disk storage
- 32 GB of RAM (more for large sites)

Minimum testing system requirements:

- 2 CPU Cores at 2 GHz or faster
- 20 GB of local disk storage
- 16 GB of RAM (more for large sites)

Sample customer deployments

Up to 1M assets:

- Two virtual machines (one for console and one for database)
- 32 CPU Cores at 2 GHz or faster
- 1TB of local disk storage
- 128 GB of RAM

Up to 500K assets:

- 32 CPU Cores at 2 GHz or faster
- 1TB of local disk storage
- 128 GB of RAM

Up to 50K assets:

- 16 CPU Cores at 2 GHz or faster
- 1TB of local disk storage
- 64 GB of RAM

Software requirements

- PostgreSQL 14 or newer (17+ preferred; the installer can install it for you)

Supported operating systems

- Ubuntu 18.04 and newer running on x86_64
- Red Hat Enterprise Linux 7.x and newer running on x86_64
- CentOS Linux 7.x and newer running on x86_64
- Oracle Linux 7.x and newer running on x86_64
    - 8.x must be 8.4+ with UEK 5.4+ or kernel 4.18+
    - 7.x must be 7.9+ with UEK 5.4+ or kernel 3.10+
- Debian Linux 9.x and newer running on x86_64

Windows Subsystem for Linux is not supported.

A minimal install of Ubuntu or any RHEL derivative is enough for the runZero console; you don't need a GUI. The default disk partitioning scheme works, but consider LVM if you plan to resize the disk later.

Note about Debian

Debian Linux uses the `su` command instead of `sudo` by default, and you must become root with `su -`, not just `su`, so that `PATH` includes system administration commands such as `useradd`. If the installer fails with `executable file not found in $PATH`, this is the most likely reason.

Connectivity

The self-hosted runZero platform needs a connection to the runZero SaaS backend over TCP port 443 (TLS) for online updates. The IP addresses and hostnames depend on your deployment model and region.

United States

The console hostname is `console.runzero.com`.

IPv4

- 13.248.161.247
- 76.223.34.198

IPv6

- 2600:9000:a415:cd87:fbe5:476a:3533:69f2
- 2600:9000:a716:ee91:85f9:3c9:48c9:59b9

Germany

The console hostname is `console-eu.runzero.com`.

IPv4

- 15.197.131.232
- 3.33.248.90

IPv6

- 2600:9000:a603:e925:542d:6d40:6897:bc3a
- 2600:9000:a70e:635f:71bd:bb0a:8e43:9466

During installation and later software updates, the platform also needs to reach these sites over TCP port 443:

- apt.postgresql.org

Platform address & TLS configuration

The system running the runZero platform should have a static IP address.

Explorers must be able to validate their HTTPS connection to the platform with a TLS certificate. By default, the runZero platform installer sets up a self-signed certificate for the console's IP address, and downloaded Explorers come preconfigured with that URL and certificate so they can make a verified connection.



Some parts of the application still use the name "Rumble" for backwards compatibility. We'll update the documentation as they change.

Because Explorers pick up the console address and certificate at download time, changing the IP address or DNS name later means redeploying the Explorers so they receive the new address. Renewing a self-signed certificate also means redeploying them.



We recommend a real certificate from an internal or public CA for your runZero console deployment. With one in place, Explorers don't need reinstalling when the server's IP address changes or the certificate is renewed. As long as the DNS name stays the same and the new certificate still validates, Explorers keep working.

To use a real certificate from an internal or public CA, assign the console server a DNS name and set `RUNZERO_CONSOLE` in the configuration to the fully qualified domain name. Point `TLS_CERT` and `TLS_KEY` at the certificate and private key files, then restart the console with `runzeroctl restart`. Explorers downloaded after that come preconfigured with the right URL and certificate.

Behind a reverse proxy, the proxy may need extra configuration to pass the `Origin` header through to the runZero Console. CSRF protection requires this header, and some features (such as the data tables throughout the product) stop working if it is missing or incorrect.

Offline mode

The self-hosted version of runZero can run in offline mode. In this mode the console doesn’t use the runZero update service, so you must apply offline updates manually. Enable it on an isolated network, or whenever you don’t want your self-hosted runZero console making any connection to the internet. Offline mode also disables certain DNS probes that could reflect responses to the internet during a scan.

Installation steps

For offline installs, see offline installation. For installs that use your own database credentials, see Installation with your own PostgreSQL database.

The installer:

- Sets up PostgreSQL and creates a dedicated user with a strong password.
- Generates TLS certificates for your IP address in `/etc/runzero/certs`.
- Generates a configuration file at `/etc/runzero/config` with some defaults.
- Creates a `systemD` or RC service for the runZero platform.

Step 0: Start from a clean environment

If an earlier run of the installer hit a problem with disk space, memory, or repository verification, you may need to clean up its temporary files before retrying the install.

Remove any temporary files left from the previous install:

```
# rm -rf /tmp/update-tmp* /tmp/runzero-updates*
```

On RHEL and CentOS systems, clear any PostgreSQL remnants:

```
# dnf remove postgresql
# rm -rf /var/lib/pgsql/
```

Use the installer binary to purge any leftovers:

```
# ./runzero-install.bin uninstall --purge
```

Step 1: Download and run the installer

- Go to https://console.runzero.com/deploy/download/platform.
- Copy the command from the download page and run it in your terminal. It downloads the installer and marks it executable. The download path is uniquely keyed.
- Run the installer.
- The console installs to `/opt/runzero` by default. To change that, use the `--install-directory` option.

Step 2: Check disk space

The runZero console stores data in the `storage` directory under the install directory, `/opt/runzero/storage` by default. The `storage` directory must have room for all unexpired scan data and asset data. To move it, set the `RUNZERO_STORAGE_PATH` variable in the runZero configuration file `/etc/runzero/config`.

The temporary directory also needs plenty of space for software and content updates. If the `TMPDIR` environment variable is set, runZero uses that directory; otherwise it uses `/tmp`.

Step 3: Initialize the admin user

Installing the runZero platform gives you the runZero CLI, `runzeroctl`. To initialize an admin user, run:

```
runzeroctl initial [email address]
```

Step 4: Sign in to your self-hosted console

If everything is set up correctly, you can sign in to your console at `https://YourInternalIPAddress`.

You may need to allow HTTPS through the Linux system firewall. Example commands:

Ubuntu Linux: `sudo ufw allow https/tcp`

RHEL/CentOS/Oracle: `sudo firewall-cmd --add-service=https`

To make a `firewall-cmd` change permanent across reboots, run the command a second time with the `--permanent` flag.

Installation with your own PostgreSQL database

By default, runZero installs and configures a PostgreSQL user and database for you. To supply your own, install with the manual database option described below.

Requirements

- PostgreSQL 13 or newer (16+ preferred)
- Password authentication must be enabled.
- Two extensions are required: pg_trgm and uuid-ossp. Depending on where you get your PostgreSQL packages, these may ship in a `contrib` package rather than with the main PostgreSQL server install.

This PostgreSQL example enables the extensions and adds a database and user:

```
CREATE DATABASE runzero TEMPLATE='template0' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8';
CREATE USER runzero WITH PASSWORD 'YOURPASSWORD';
GRANT ALL PRIVILEGES ON DATABASE runzero TO runzero;
\connect runzero;
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

To install the self-hosted runZero platform with your own database credentials:

1. Run this install command as root:

```
./runzero-platform-[VERSION]-linux-amd64.bin install --manual-database
```

2. Add your database details to the runZero configuration in `/etc/runzero/config`. The line to edit is:

```
DATABASE_URL=postgres://runzero:{DB_PASSWORD}@127.0.0.1:5432/runzero?sslmode=disable
```

Change it to match your credentials. You need to set the user, password, host, port, and database name, in this format:

```
DATABASE_URL=postgres://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}?{DB_OPTIONS}
```

4. Verify that the self-hosted runZero platform can connect to your database: `sudo runzeroctl database verify`
5. Once the database is configured and verified, restart the self-hosted runZero platform service: `sudo systemctl restart runzero-console`

runZero updates

Update the self-hosted runZero platform before first use. You can use the `runzeroctl` command to download the update and then restart the service once the update completes. For offline updates, see CLI update with offline mode.

You can update the platform and scanners together or separately with the CLI update management commands.

Managing users

Manage users in your self-hosted runZero platform console at `https://YourInternalIPAddress/team` or with the runZero CLI. Tasks include:

- Adding, deleting, and listing users
- Resetting passwords
- Changing default roles
- Viewing details
- Changing organization roles

CLI service management

Start the runZero service

Starts the runZero platform service.

```
runzeroctl start
```

Stop the runZero service

Stops the runZero platform service.

```
runzeroctl stop
```

Restart the runZero service

Restarts the runZero platform service.

```
runzeroctl restart
```

Install the runZero platform

Installs the runZero platform service and its dependencies, such as PostgreSQL. Creates a systemd service, generates cron jobs, and writes a configuration file to `/etc/runzero/config`.

```
runzeroctl install
```

Uninstall the runZero platform

Stops the runZero platform service, removes it from systemd, and removes the generated cron jobs. Your PostgreSQL database and data stay in place.

```
runzeroctl uninstall
```

Purge the runZero platform

Stops the runZero platform service, removes it from systemd, and removes the generated cron jobs. It also deletes your runZero database and removes the runZero directories `/etc/runzero` and `/opt/runzero`.

```
runzeroctl purge
```

To uninstall and purge everything except the database and your PostgreSQL settings, use this flag:

```
runzeroctl uninstall --purge --ignore-database
```

Run the runZero platform manually

Starts the runZero platform manually and writes logs to standard output.

```
runzeroctl server
```

Verify your database is reachable

Tries to connect to your database using your self-hosted runZero platform configuration, and either succeeds or shows an error.

```
runzeroctl database verify
```

CLI update management

Update the runZero platform and scanners

Updates the runZero platform service and the runZero scanners. The optional `force` parameter forces the update even when the current installation is already the latest version.

```
runzeroctl update [--force]
```

Update the runZero platform

Updates just the runZero platform service. The optional `force` parameter forces the update even when the current install is already the latest version.

```
runzeroctl update-platform [--force]
```

Update the runZero scanners

Updates just the runZero scanners. The optional `force` parameter forces the update even when the current installation is already the latest version.

```
runzeroctl update-scanner [--force]
```

CLI user management

Create the initial administrator account

Creates the initial admin user for a new installation. You must provide an email address.

```
runzeroctl initial <email>
```

List user accounts

Lists all users with their email address, full name, and current roles.

```
runzeroctl user list
```

Add a user account

Creates a new user account under the initial administrator user. You must provide an email address.

```
runzeroctl user add <email>
```

Delete a user account

Deletes a user account. You must provide an email address. This cannot be undone.

```
runzeroctl user delete <email>
```

Get user details

Shows the details for a user account, such as full name, date created, last sign-in IP, last sign-in time, last activity, default organization role, and current roles. You must provide an email address.

```
runzeroctl user details <email>
```

Set a user role

Sets a user's role to the role provided. You must provide the email and the role; the organization is optional. Without an organization, this sets the user's default role.

```
runzeroctl user set-role <email> [organization name or organization ID]:<role>
```

Reset a user password and MFA

Generates and applies a new password for the specified user and prints it to the terminal. You must provide an email address. The reset also clears any associated MFA tokens.

```
runzeroctl user reset <email>
```

CLI organization management

List all organizations

Lists all organizations by name and ID.

```
runzeroctl organization list
```

Advanced configuration

The file at /etc/runzero/config holds the settings described below. After editing it, apply the changes with `runzeroctl restart`.

Email server

runZero sends user account invitations and notifications through an SMTP server. The default configuration expects an SMTP server on localhost that needs no authentication:

```
SMTP_SERVER=127.0.0.1:25
SMTP_AUTH_METHOD=none
```

With plaintext SMTP servers, runZero uses STARTTLS automatically and validates the certificate. If your internal SMTP server lacks a valid TLS certificate, disable verification with:

```
SMTP_TLS_NOVERIFY=true
```

For transport-layer TLS instead of STARTTLS, set:

```
SMTP_TLS=true
```

If the server requires authentication, configure these three settings:

```
SMTP_AUTH_METHOD=plain
SMTP_AUTH_USER=YourUsername
SMTP_AUTH_PASS=YourPassword
```

The supported values for `SMTP_AUTH_METHOD` are `plain`, `login`, and `none` (no authentication).

Emails come from `noreply@runzero.com` by default. To change the sender, set the `FROM_EMAIL` option:

```
FROM_EMAIL=runzero@yourcompany.int
```

Hostname and port

The RUNZERO_CONSOLE variable is the address that users and deployed Explorers use to connect to the platform. runZero also uses it to build inbound links, configure deployed Explorers, and generate the default self-signed TLS certificate.

```
RUNZERO_CONSOLE=https://{IP ADDRESS OR HOSTNAME}:443
```

Changing it may require regenerating the TLS certificate and redeploying Explorers.

```
runzeroctl generate-certificate
```

To run runZero on a different port, set CONSOLE_PORT. This is the port the console listens on; users and Explorers still connect to the RUNZERO_CONSOLE value, so in most cases the two should match.

```
CONSOLE_PORT=443
```

TLS configuration

runZero generates a self-signed TLS certificate and serves all web requests using HTTP over TLS. The standard configuration uses a self-signed certificate stored in the filesystem:

```
TLS=true
TLS_CERT=/etc/runzero/certs/cert.pem
TLS_KEY=/etc/runzero/certs/key.pem
```

The certificate and key file are PEM encoded, and you can replace them with any valid certificate. Any new certificate must list the value of RUNZERO_CONSOLE among its Subject Alternative Names.

The key file must be unencrypted (not password protected). If it has a passphrase, remove it with OpenSSL. For example:

```
# RSA key
openssl rsa -in key.pem -out newkey.pem
# DSA key
openssl dsa -in key.pem -out newkey.pem
```

Behind a TLS-terminating reverse proxy (AWS ELB, nginx, etc), you can disable TLS at the application level:

```
TLS=false
```

The web interface is then reachable over plain HTTP, but Explorers refuse to connect to a plain HTTP port, and features like WebAuthn MFA only work when the site is accessed through TLS.

You can also set specific TLS versions and ciphers.

TLS versions are chosen by minimum and maximum:

```
TLS_VERSION_MIN=1.2
TLS_VERSION_MAX=1.3
```

TLS ciphers may be chosen by profile name:

- **default**: a set of strong ciphers, a good fit for most configurations
- **nist80052**: a set of strong ciphers, approved in NIST 800-52r2
- **nist80052-aes256**: a set of strong ciphers, approved in NIST 800-52r2, restricted to AES-256 variants

TLS 1.3 ciphers work differently, so if you need a specific set of ciphers, set both TLS_VERSION_MIN and TLS_VERSION_MAX to `1.2`. For example, to restrict runZero to only the NIST 800-52r2 approved ciphers using AES-256:

```
TLS_VERSION_MIN=1.2
TLS_VERSION_MAX=1.2
TLS_CIPHERS=nist80052-aes256
```

TLS ciphers may also be chosen as a comma-separated list of cipher constants.

Certificate authorities (CAs)

The server validates TLS connections against the system-installed certificate authorities and an internal CA certificate bundle. By default, both the system roots and the bundled roots are considered for every secure TLS connection.

- Set `RUNZERO_TLS_IGNORE_SYSTEM_ROOTCA` to `true` to ignore the system CA roots.
- Set `RUNZERO_TLS_IGNORE_EMBEDDED_ROOTCA` to `true` to ignore the bundled CA roots.
- Set `RUNZERO_TLS_ADDITIONAL_ROOTCA` to the path of a file containing additional CA roots in PEM format.

Database configuration

runZero keeps all platform data in a PostgreSQL database, except raw scan files, change reports, and images processed from scans. By default, runZero configures a local PostgreSQL server on the same system, with a random password and without TLS encryption:

```
DATABASE_URL=postgres://runzero:{DB_PASSWORD}@127.0.0.1:5432/runzero?sslmode=disable
```

If you prefer a separate database, any PostgreSQL server running 12.x or newer should work. TLS (sslmode=require) should be enabled whenever the database server is not local.

For high-throughput environments, you can change the default database pool (connection count):

```
DATABASE_POOL_COUNT=50
```

Proxy configuration

runZero makes outbound connections to receive platform updates (in online mode), to reach third-party APIs, and to deliver webhooks for notifications. If those connections need a proxy server, set:

```
HTTPS_PROXY=host:port
```

You can set a separate proxy for AWS connections; it takes precedence over `HTTPS_PROXY`:

```
HTTPS_PROXY_AWS=host-for-aws-services:port
```

Besides standard HTTP proxies, you can use a SOCKS proxy by specifying a URL like:

```
HTTPS_PROXY=socks5://socks.example.com:1080
```

Storage configuration

runZero stores raw scan data, change reports, and images retrieved from assets in local file storage. The storage directory must be owned by the `runzero` user and mounted below the /opt/runzero path.

```
RUNZERO_STORAGE_MODE=local
RUNZERO_STORAGE_PATH=/opt/runzero/storage
```

Files in the storage directory are split into two groups, assets and scans. To rename them, set:

```
ASSET_BUCKET=assets
SCAN_BUCKET=scans
```

To use AWS S3 for file storage instead, set:

```
RUNZERO_STORAGE_MODE=s3
ASSET_BUCKET=company-runzero-assets
```

```
SCAN_BUCKET=company-runzero-scans
```

For a non-AWS backend that is compatible with the S3 API, use the same AWS and bucket variables, override `AWS_REGION`, and set `AWS_ENDPOINT_URL_S3` or `RUNZERO_STORAGE_ENDPOINT` to the endpoint. Reach out to runZero support if you run into trouble with the endpoint configuration.

If the S3 buckets are in a different region than the environment, set `RUNZERO_STORAGE_REGION` to the buckets' region.

With S3, AWS must also be configured with at least the `AWS_REGION` variable set, even when a non-AWS backend is enabled.

Secret configuration

runZero secures the platform with three randomly generated secret tokens. Each is a hexadecimal string made from 16 random bytes, and OpenSSL can generate compatible values:

```
$ openssl rand -hex 16
```

The authentication key for local storage HMAC operations can be rotated as long as you restart the service afterwards:

```
RUNZERO_STORAGE_KEY_SECRET={SECRET_32_HEX}
```

The session secret key signs and validates browser session cookies. You can rotate it, but doing so invalidates all existing web sign-ins:

```
SESSION_SECRET={SECRET_32_HEX}
```

The DB key encrypts sensitive fields (user password hashes). It cannot be rotated without breaking password authentication: if it changes, users must reset their password from the command line or the web interface using email before they can sign back in:

```
DB_KEY={SECRET_32_HEX}
```

AWS configuration

The AWS region is required:

```
AWS_REGION=us-east-1
```

The Access Key ID and Secret must be valid and belong to a user with read-write access to the S3 buckets and read-only access to Secrets Manager.

```
AWS_ACCESS_KEY_ID=AKIA....
AWS_SECRET_ACCESS_KEY=SECRET....
```

runZero can load almost any configuration setting from AWS Secrets Manager at startup. The Secrets Manager entries should match the key names in the configuration file. Set the secret name with:

```
AWS_SECRETS_MANAGER_KEY=runzero/production
```

Most AWS CLI environment variables are also available alongside these variables for finer tuning.

These settings change where the Explorer and scanner binaries are stored. They should still live under /opt/runzero, or the service can't load them:

```
RUNZERO_RELEASE_DIR=/opt/runzero/agent/
RUNZERO_SCANNER_RELEASE_DIR=/opt/runzero/scanner/
```

Content security policy

With a non-standard S3 configuration (or an S3-like deployment such as minio), the Content Security Policy headers must allow external image loads.

Set CSP_IMAGES to one or more (comma-delimited) external image sources:

```
CSP_IMAGES=https://*.custom-storage-backend.lan
```

These CSP settings are also available:

```
CSP_SCRIPTS=https://*.myscripts.lan
CSP_FONTS=https://*.myfonts.lan
CSP_STYLES=https://*.mystyles.lan
```

Logging

The self-hosted runZero console writes its logging output to standard output. On Linux, systemd picks this up and stores it in the journal, where you can query it with the `journalctl` tool. For example, this command shows the most recent hour of logs, newest first:

```
journalctl --unit=runzero-console --since=-1hour --reverse
```

You can pipe the journalctl output to a text file to send to runZero support.

You can configure the systemd logging subsystem to send log messages to a local syslog daemon as well as the journal. Once logs are in syslog, you can forward them across the network to remote logging servers using the standard syslog protocol. Some Linux distributions, such as RHEL, forward logs from systemd to syslog by default. Others, such as Ubuntu and Debian, don't include a syslog daemon in their default minimal server installs.

To make systemd send logs to syslog, use the ForwardToSyslog option in `/etc/systemd.conf`. Alternatively, some syslog daemons can read the systemd journal themselves; rsyslog, for example, has imjournal.

runZero writes its logs in CEE-enhanced JSON format, which works with rsyslog, syslog-ng, DataDog, and other common logging tools. For rsyslog, the mmjsonparse module can filter the logs on individual JSON fields and forward them to ElasticSearch or other JSON databases.

Set `LOG_MAX_LENGTH` in runZero's config file to cap the length of each log line, in bytes of UTF-8 text. A value of 0 means no limit, and other values below 480 are treated as 480. When truncating, runZero tries to preserve the most useful logging fields, and the result stays valid JSON. The limit applies before systemd or syslog adds anything to the start of the line.

Set `LOG_FORMAT` to `text` to turn off the CEE and JSON format and log in plain text. For example:

```
LOG_FORMAT=text
LOG_MAX_LENGTH=512
```

HTTP timeouts

Three configuration variables set the built-in web server's HTTP idle, read, and write timeouts. The defaults are below and don't usually need changing.

```
HTTP_IDLE_TIMEOUT_MINUTES=3
HTTP_READ_TIMEOUT_MINUTES=60
HTTP_WRITE_TIMEOUT_MINUTES=720
```

Concurrent processing

The runZero Console can process more than one completed task at a time when `RUNZERO_CRUNCHER_INSTANCES` is set to a value greater than 1. Tasks run concurrently only when they belong to different organizations, or to different sites within the same organization when the task doesn't require cross-site asset merging. Most third-party connectors and integrations require cross-site merging, so they can't take advantage of concurrent site processing within one organization. Resource requirements for concurrent task processing scale linearly with the instance count.

This example lets the console process up to four concurrent tasks across all organizations:

```
RUNZERO_CRUNCHER_INSTANCES=4
```

Custom JavaScript

The self-hosted runZero Console can run your own JavaScript on its web pages. Add your JavaScript to /opt/runzero/etc/custom.js. The custom.js file doesn't exist until you create it in the /opt/runzero/etc folder.

Then enable the feature by adding this variable to your configuration:

```
RUNZERO_CUSTOM_JS=true
```

HTTP headers

You can disable the security headers sent by the runZero Console as needed with these options:

```
# Disable Strict-Transport-Security
RUNZERO_DISABLE_HSTS=true

# Disable X-Frame-Options
RUNZERO_DISABLE_HXFO=true

# Disable X-Content-Type-Options
RUNZERO_DISABLE_HXCTO=true

# Disable X-XSS-Protection
RUNZERO_DISABLE_HXXP=true

# Disable Referrer-Policy
RUNZERO_DISABLE_HRP=true

# Disable Content-Security-Policy
RUNZERO_DISABLE_HCSP=true

# Disable CSRF-Protection
RUNZERO_DISABLE_CSRF=true
```

API availability

This setting controls whether public APIs such as /health are enabled:

```
# Disable all unauthenticated API endpoints (/health)
RUNZERO_DISABLE_PUBLIC_APIS=true
```

Disabling entitlements

You can force-disable any license entitlement for the entire instance by setting `RUNZERO_DISABLE_<ENTITLEMENT>` to `true`, where `<ENTITLEMENT>` is the uppercased entitlement name. This overrides the license: it can only remove an entitlement, never grant one, and it applies to every account and organization on the instance.

For example, to disable autonomous (autoscan) discovery everywhere:

```
RUNZERO_DISABLE_AUTOSCAN=true
```

The full set of supported variables is below, one per entitlement. Each is commented out; set the value to `true` to disable that entitlement.

```
# Asset ownership
# RUNZERO_DISABLE_ASSET_OWNERSHIP=true

# Risky assets
# RUNZERO_DISABLE_RISKY_ASSETS=true

# Passive monitoring, traffic sampling, and PCAP import
# RUNZERO_DISABLE_PASSIVE_MONITORING=true
```

```
# Manage Explorer groups
# RUNZERO_DISABLE_EXPLORER_GROUPS=true

# Goal tracking
# RUNZERO_DISABLE_GOAL_TRACKING=true

# Unlimited organizations (not just one)
# RUNZERO_DISABLE_ORGS_UNLIMITED=true

# Unlimited projects (not just one)
# RUNZERO_DISABLE_PROJECTS_UNLIMITED=true

# Continuous scanning and variable TTL
# RUNZERO_DISABLE_SCANS_UNLIMITED=true

# Concurrent scanning on Explorers
# RUNZERO_DISABLE_SCANS_CONCURRENT=true

# Subnet ping and host ping (sampling) modes
# RUNZERO_DISABLE_SCANS_SAMPLING=true

# Advanced scan targeting
# RUNZERO_DISABLE_SCANS_TARGETING=true

# Autonomous (autoscan) discovery scanning
# RUNZERO_DISABLE_AUTOSCAN=true

# AI features
# RUNZERO_DISABLE_AI=true

# Organization API access
# RUNZERO_DISABLE_API_ORG=true

# Account (client) API access
# RUNZERO_DISABLE_API_ACCOUNT=true

# Adjust organization data retention settings
# RUNZERO_DISABLE_DATA_RETENTION=true

# Rule engine processing
# RUNZERO_DISABLE_RULES_BASIC=true

# Basic reporting
# RUNZERO_DISABLE_REPORTS_BASIC=true

# Disable external (shareable, email-verified) report links
# RUNZERO_DISABLE_DISABLE_EXTERNAL_REPORTING=true

# ServiceNow export APIs
# RUNZERO_DISABLE_INT_SERVICENOW=true

# Splunk sync APIs
# RUNZERO_DISABLE_INT_SPLUNK=true
```

```
# Cisco export APIs
# RUNZERO_DISABLE_INT_CISCO=true

# HP export APIs
# RUNZERO_DISABLE_INT_HP=true

# Censys search import
# RUNZERO_DISABLE_INT_CENSYS=true

# CrowdStrike import
# RUNZERO_DISABLE_INT_CROWDSTRIKE=true

# Microsoft 365 Defender import
# RUNZERO_DISABLE_INT_MS365DEFENDER=true

# Miradore import
# RUNZERO_DISABLE_INT_MIRADORE=true

# Nessus import
# RUNZERO_DISABLE_INT_NESSUS=true

# Rapid7 Nexpose import
# RUNZERO_DISABLE_INT_RAPID7=true

# SentinelOne import
# RUNZERO_DISABLE_INT_SENTINELONE=true

# VMware API import
# RUNZERO_DISABLE_INT_VMWARE=true

# Tenable vulnerability management
# RUNZERO_DISABLE_INT_TENABLE=true

# Prisma Cloud integration
# RUNZERO_DISABLE_INT_PRISMA=true

# Tenable Security Center integration
# RUNZERO_DISABLE_INT_TENABLE_SECURITY_CENTER=true

# Rapid7 InsightVM vulnerability management
# RUNZERO_DISABLE_INT_INSIGHTVM=true

# Qualys vulnerability management
# RUNZERO_DISABLE_INT_QUALYS=true

# Shodan integration
# RUNZERO_DISABLE_INT_SHODAN=true

# Azure AD import
# RUNZERO_DISABLE_INT_AZURE_AD=true

# LDAP import
# RUNZERO_DISABLE_INT_LDAP=true
```

```
# Intune import
# RUNZERO_DISABLE_INT_INTUNE=true

# Google Workspace import
# RUNZERO_DISABLE_INT_GOOGLE_WORKSPACE=true

# Wiz import
# RUNZERO_DISABLE_INT_WIZ=true

# Meraki integration
# RUNZERO_DISABLE_INT_MERAKI=true

# Microsoft MECM integration
# RUNZERO_DISABLE_INT_MECM=true

# Tanium integration
# RUNZERO_DISABLE_INT_TANIUM=true

# NetBox integration
# RUNZERO_DISABLE_INT_NETBOX=true

# Dragos integration
# RUNZERO_DISABLE_INT_DRAGOS=true

# Jira integration
# RUNZERO_DISABLE_INT_JIRA=true

# Palo Alto Networks integration
# RUNZERO_DISABLE_INT_PAN=true

# Manage directory users
# RUNZERO_DISABLE_DIRECTORY_USERS=true

# Manage directory groups
# RUNZERO_DISABLE_DIRECTORY_GROUPS=true

# Cloud connections (AWS, GCP, Azure)
# RUNZERO_DISABLE_INT_CLOUD=true

# Create custom integrations
# RUNZERO_DISABLE_INT_CUSTOM_INTEGRATIONS=true

# Custom integration scripts
# RUNZERO_DISABLE_INT_CUSTOM_INTEGRATION_SCRIPTS=true

# Advanced reports
# RUNZERO_DISABLE_REPORTS_ADVANCED=true

# Self-hosted platform
# RUNZERO_DISABLE_SELFHOSTED=true

# Bulk user management
# RUNZERO_DISABLE_BULK_USER_MANAGEMENT=true
```

```
# Group expiration
# RUNZERO_DISABLE_GROUP_EXPIRATION=true

# SSO group mapping
# RUNZERO_DISABLE_GROUP_MAPPING=true

# Hosted external scans (hosted zones)
# RUNZERO_DISABLE_HOSTED_ZONES=true

# Rebrand
# RUNZERO_DISABLE_REBRAND=true

# Knowledge base sidebar
# RUNZERO_DISABLE_KNOWLEDGE_BASE=true

# Customizable dashboards
# RUNZERO_DISABLE_CUSTOM_DASHBOARDS=true

# Multiple customizable dashboards
# RUNZERO_DISABLE_MULTIPLE_DASHBOARDS=true

# Query builder
# RUNZERO_DISABLE_QUERY_BUILDER=true

# Metadata minimization
# RUNZERO_DISABLE_METAMIN=true

# Offline mode
# RUNZERO_DISABLE_OFFLINE=true

# Certificate inventory
# RUNZERO_DISABLE_CERTIFICATE_INVENTORY=true

# Findings inventory
# RUNZERO_DISABLE_FINDINGS=true

# Risk management event rule types
# RUNZERO_DISABLE_RISK_EVENTS=true

# Baseline goals
# RUNZERO_DISABLE_BASELINE_GOALS=true

# Immediate analysis recalculation on rapid response publish
# RUNZERO_DISABLE_IMMEDIATE_RAPID_RESPONSE_RECALC=true

# Create new sites
# RUNZERO_DISABLE_CREATE_SITES=true

# Suppress vulnerabilities, vulnerability groups, and findings
# RUNZERO_DISABLE_SUPPRESSION=true

# Issues
# RUNZERO_DISABLE_ISSUES=true
```

```
# Archive vulnerabilities
# RUNZERO_DISABLE_ARCHIVE_VULNERABILITIES=true
```

Unofficial CPEs

When runZero fingerprints an asset's operating system, it generates a CPE. If the NIST database has no official match, runZero generates an unofficial CPE by default. To turn this off, set `RUNZERO_GENERATE_UNOFFICIAL_CPE` to `false`:

```
RUNZERO_GENERATE_UNOFFICIAL_CPE=false
```

Unofficial CPEs include `r0_unofficial` in the `other` field of the CPE by default. You can change this to any alphanumeric-constrained tag of up to 32 characters:

```
RUNZERO_UNOFFICIAL_CPE_TAG=custom_unoffical_tag
```

Rewriting `cve.org` URLs

To rewrite the `cve.org` links in the product UI to point at an internal mirror or another site, set the environment variable `RUNZERO_REPLACE_URL_PREFIX_CVEORG`. For example:

```
RUNZERO_REPLACE_URL_PREFIX_CVEORG=https://cve.mirror.local/cve/
```

With this example, `cve.org` URLs in the product interface are rewritten to `https://cve.mirror.local/cve/$1`, where `$1` is replaced with the CVE identifier, such as `CVE-12345`.

Permissions

Installing and managing the self-hosted platform from the command line requires **root** access.

The platform service (`runzero-console`) runs as **root** and spawns a worker subprocess that runs as the **runzero** user account inside a chroot environment (/opt/runzero). All substantive work happens in that isolated subprocess. Older installations use `rumble` instead of `runzero` in directory, file, and user names.

The self-hosted platform uses these filesystem locations:

/etc/runzero

Path	Owner	Permission	Notes
`/etc/runzero`	root	0700	Configuration files and certificates
`/etc/runzero/config`	root	0600	A plain-text configuration file
`/etc/runzero/certs`	root	0700	A directory containing the TLS certificate and key
`/etc/runzero/certs/cert.pm`	root	0600	The TLS certificate in PEM format

`/etc/runzero/certs/key.pm`	root	0600	The TLS certificate private key in PEM format

/opt/runzero

Path	Owner	Permission	Notes
`/opt/runzero/tmp`	runzero	0755	A temporary directory
`/opt/runzero/storage`	runzero	0700	Contains asset and scan artifacts
`/opt/runzero/console`	root	0755	Contains the platform executable
`/opt/runzero/console/runzero-console.bin`	root	0755	The platform executable
`/opt/runzero/agent`	root	0755	Contains the Explorer binaries
`/opt/runzero/agent/runzero-agent-*`	root	0755	The Explorer binaries
`/opt/runzero/scanner`	root	0755	Contains the CLI binaries
`/opt/runzero/agent/runzero-scanner-*`	root	0755	The CLI binaries
`/opt/runzero/proc`	root	0755	Contains copies of system /proc files
`/opt/runzero/proc/cpuinfo`	root	0644	A copy of /proc/cpuinfo
`/opt/runzero/proc/meminfo`	root	0644	A copy of /proc/meminfo
`/opt/runzero/proc/version`	root	0644	A copy of /proc/version
`/opt/runzero/etc`	root	0755	Contains copies of system files
`/opt/runzero/etc/resolv.conf`	root	0644	A copy of /etc/resolv.conf
`/opt/runzero/etc/hosts`	root	0644	A copy of /etc/hosts
`/opt/runzero/etc/ca-certificates.crt`	root	0644	A copy of the system root CA store
`/opt/runzero/etc/runzero`	runzero	0700	Contains instance identifiers
`/opt/runzero/etc/runzero/cruncher.id`	runzero	0700	A unique ID to identify the cruncher instance
`/opt/runzero/etc/runzero/hub.id`	runzero	0700	A unique ID to identify the hub instance
`/opt/runzero/config`	root	0700	Unused today

Backup and restoration

You can back up and restore your runZero installation and data to preserve your configuration.

runZero data backup

To back up a self-hosted installation, archive the file system and the database.

The file system archive includes these paths:

- `/etc/runzero`
- `/opt/runzero`
- `/lib/systemd/system/runzero-console.service`
- `/etc/systemd/system/multi-user.target.wants/runzero-console.service`
- `/usr/bin/runzeroctl`

A sample file system backup command:

```
# tar zcvf runzero-backup-fs.tar.gz /etc/runzero/ /opt/runzero/ \
/lib/systemd/system/runzero-console.service \
/etc/systemd/system/multi-user.target.wants/runzero-console.service \
/usr/bin/runzeroctl
```

Back up the PostgreSQL database separately. A sample command:

```
# sudo su - postgres
$ pg_dumpall -f runzero.sql && gzip runzero.sql
```

runZero data restoration

To restore the runZero install:

1. Stop any running runZero service:

```
# runzeroctl stop
```

2. Unpack the filesystem archive:

```
# tar -C / -zxvf /path/to/runzero-backup-fs.tar.gz
```

3. Restore the PostgreSQL database:

```
# sudo su - postgres
$ dropdb runzero; gzip -dc runzero.sql.gz | psql
```

4. Restart the runZero service:

```
sudo systemctl restart runzero-console
```

Support and debugging

The `runzeroctl` command includes a debugging tool that collects diagnostics from your server and assembles them into a zip file you can send to support. If support asks for it, run:

```
runzeroctl diagnostics run-script
```

Data is written to `/opt/runzero/collector`.

To examine the script before running it, save a copy to the current directory instead:

```
runzeroctl diagnostics write-script
```

For more on the diagnostics collection script, see Self-hosted troubleshooting.

Manual migrations

Starting with version `4.0.240221.0`, the runZero self-hosted upgrade runs migrations before restarting the service. If you run an older version and want to avoid downtime while upgrading a single-node self-hosted installation, use these steps:

1. Get the self-hosted download link from the runZero SaaS.
2. Download the file manually to your self-hosted systems:

```
$ curl -o platform.bin https://console.runzero.com/...../runzero-platform.bin
```

3. Mark the file executable and run it with the `task db:migrate` parameter:

```
$ chmod u+x platform.bin; ./platform.bin task db:migrate
```

4. Once the migrations finish, install the update as usual:

```
$ runzeroctl update
```

Offline mode configuration

Supported operating systems

The runZero self-hosted console runs on the same operating systems whether you install online or offline: Ubuntu 18.04+, RHEL 7+, CentOS 7+, Oracle Linux 7+, and Debian 9+, on x86_64. See Supported operating systems in Self-hosting runZero for the full list.

Offline installs need PostgreSQL packages available locally. The PostgreSQL.org RPM bundles linked below cover RHEL/CentOS 8, 9, and 10. On older or other distributions, supply PostgreSQL through your own offline mirror, or use `--distro-packages-only` to install the PostgreSQL version your OS provides (it must be PostgreSQL 14 or newer).

Offline installation

The install command accepts these flags: `--offline`, `--postgres-version`, `--distro-packages-only`, and `--postgres-rpm-directory`.

`--offline`
: Configures the installation to run without internet access. The installer may still need network access during installation to fetch dependencies. It uses the upstream postgresql.org packages by default; the options below turn that off.

`--postgres-version`
: Selects a major version of PostgreSQL (14-18). If your operating system doesn't ship the default version (18), you may need the PostgreSQL.org packages instead, which for an offline installation may mean downloading them ahead of time.

`--distro-packages-only`
: Installs the platform without third-party repositories. The installer looks for the PostgreSQL package in the operating system's configured repositories (local or remote). If no repository is reachable, you can point it at a directory of packages on the file system with the option below. OS packages may be limited to older versions of PostgreSQL.

`--postgres-rpm-directory [directory]`
: Installs from PostgreSQL RPM files you supply (requires RHEL or CentOS). The installer uses the RPMs in the specified directory to satisfy the PostgreSQL dependencies.

The console installs to `/opt/runzero` by default. To change that, use the `--install-directory [directory]` option.

PostgreSQL RPMs required for `--postgres-rpm-directory`

Installing PostgreSQL requires four RPMs:

- postgresql18
- postgresql18-server
- postgresql18-contrib
- postgresql18-libs

x86_64 Intel

- RHEL/CentOS 10 RPMs
- RHEL/CentOS 9 RPMs
- RHEL/CentOS 8 RPMs

64-bit ARM

- RHEL/CentOS 10 RPMs
- RHEL/CentOS 9 RPMs
- RHEL/CentOS 8 RPMs

Example install commands for offline mode

First, download the runZero platform.

RHEL/CentOS 8, 9, or 10

1. Download the RPMs listed above and store them in a directory.
2. Run this install command as root:

```
./runzero-platform-[VERSION]-linux-amd64.bin install --offline
    --postgres-rpm-directory [RPM_DIRECTORY]
    --postgres-version=18
```

Ubuntu 18.04+, Debian 9+, RHEL 7+, CentOS 7+, or Oracle Linux 7+

1. Run this install command as root:

```
./runzero-platform-[VERSION]-linux-amd64.bin install --offline --distro-packages-only
```

On Debian, become root with `su -`, not just `su`. On distributions whose default PostgreSQL is older than 14, supply a newer PostgreSQL through `--postgres-rpm-directory` (RHEL/CentOS 8/9/10) or your own offline mirror.



Some parts of the application still use the name "Rumble" for backwards compatibility. We'll update the documentation as they change.

Enabling offline mode for existing installs

- Open `/etc/runzero/config` in an editor.
- Find `OFFLINE=` and change it to `OFFLINE=true`.
- Restart the runZero service with `runzeroctl restart`.

CLI update with offline mode

Update the self-hosted runZero platform before first use.

Update the runZero platform and scanners with an offline update

- Go to https://console.runzero.com/deploy/download/platform.
- Copy the command from the download page and run it in your terminal, or run one of these commands against the zip archive you downloaded:

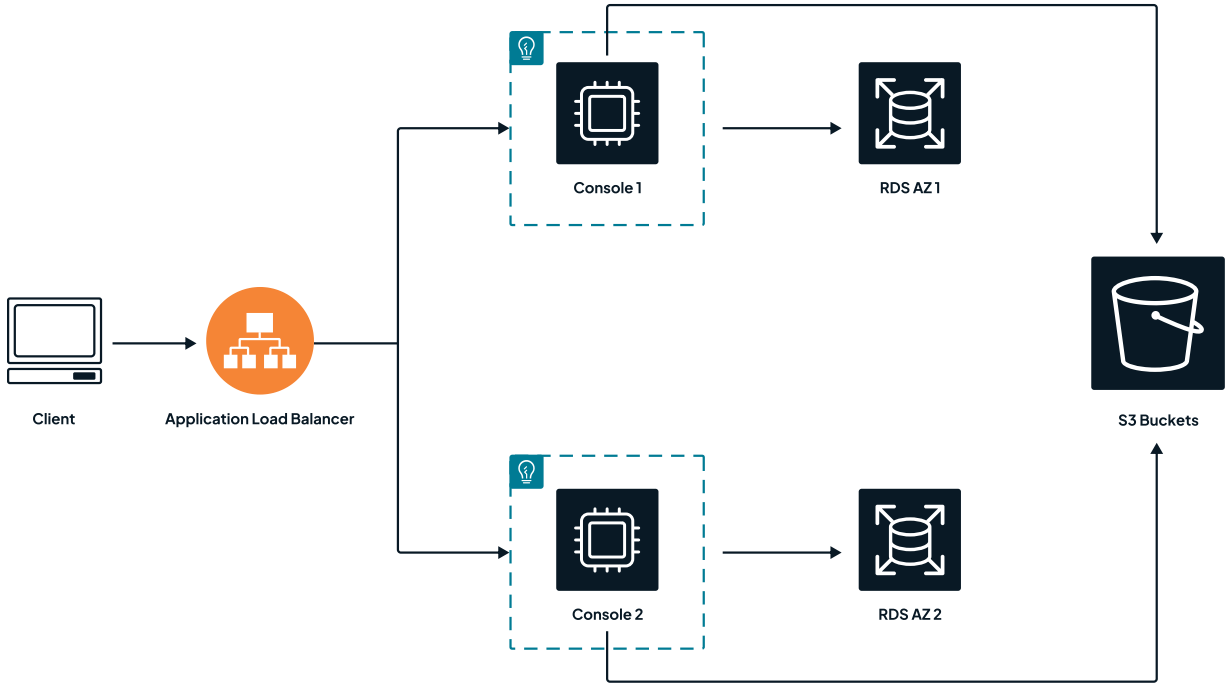
```
runzeroctl update runzero-platform-update-[VERSION].zip

runzeroctl update --offline --zip-file-path runzero-platform-update-[VERSION].zip
```

Change the version to match the zip archive you downloaded.

High-availability configuration

You can configure a self-hosted runZero installation for high-availability: a load balancer directs traffic to multiple console servers, which share a PostgreSQL cluster and a storage backend. The diagram below shows an example architecture on AWS with two availability zones.



In the diagram, an application load balancer (ALB) terminates TLS and points to a target group of two runZero servers in separate availability zones. Both servers use a multi-availability-zone PostgreSQL RDS instance configured for the same two zones, and both point to the same set of S3 storage buckets.

Installation steps

Installing and configuring a high-availability instance of runZero takes these steps:

1. Deploy an application load balancer.
2. Install and configure the first server node.
3. Install and configure subsequent server nodes.

Deploy a load balancer with TLS termination

An application load balancer gives the runZero cluster its high-availability interface: it receives TLS connections and forwards the HTTP requests to the available servers. The load balancer should be highly available on its own. Any load balancer that can proxy large HTTP requests

and websockets should work. It should add an X-Forwarded-For header to the requests it sends on to the runZero servers, and it should detect unhealthy nodes so that failover happens automatically. You can use the /health endpoint on the runZero servers for its health checks.

Prepare the first runZero server node

To install and configure the first runZero node:

1. Install the runZero platform.
2. Configure the console URL and TLS/XFF settings.
3. Configure the database settings.
4. Configure the shared storage settings.
5. Verify your configuration.
6. Create the initial user account.
7. Add the runZero server to the load balancer and verify the connection.

Install the runZero platform

Some parts of the application still use the name "Rumble" for backwards compatibility. We'll update the documentation as they change.

1. Go to the platform download page in the runZero Console.
2. Copy the URL of the platform installer from the download page. The download path is uniquely keyed to your license.
3. Download the installer with Wget or cURL:

- `wget -O runzero-platform.bin https://console.runzero.com/download/platform-combined/<UNIQUE KEY>/62e41615/runzero-platform-vX.X.X-linux-amd64.bin`
- `curl -o runzero-platform.bin https://console.runzero.com/download/platform-combined/<UNIQUE KEY>/62e41615/runzero-platform-vX.X.X-linux-amd64.bin`

4. Make the installer executable:

```
# chmod u+x runzero-platform.bin
```

5. Run the install command with the manual database option:

```
# ./runzero-platform.bin install --manual-database
```

Configure the console URL and TLS/XFF

The high-availability settings live in `/etc/runzero/config`. Start with the console URL and the TLS/XFF settings.

- Set the RUNZERO_CONSOLE variable to the hostname or IP address of the load balancer:

```
RUNZERO_CONSOLE=https://{IP ADDRESS OR HOSTNAME}:443
```

- Disable TLS and trust X-Forwarded-For headers with these two lines:

```
TLS=false
RUNZERO_TRUST_XFF=true
```

Configure the database

- Edit the DATABASE_URL value to match this format:

```
DATABASE_URL=postgres://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}?{DB_OPTIONS}
```

An example DATABASE_URL for AWS RDS looks like:

```
DATABASE_URL=postgres://rumble:[password]@rumbledb.rds-id.us-east-1.rds.amazonaws.com:5432/rumble?sslmode=require
```

To send read-only queries to a replica, set DATABASE_REPLICA_URL with the same syntax. For RDS Aurora, use the read-only endpoint as the replica URL. Without replicas, only DATABASE_URL is required, and on RDS Aurora it should point to the writer endpoint.

With any other type of PostgreSQL clustering, make sure DATABASE_URL points to the highly-available endpoint for the active writer.

Configure shared storage

The shared storage backend holds asset data, raw scan data, and reports. runZero supports both mounted NFS and object storage.

- To use NFS, mount a read-write NFS share at /opt/runzero/storage and select local storage mode with these options. The bucket names are created under the NFS mount base directory.

```
RUNZERO_STORAGE_MODE=local
RUNZERO_STORAGE_PATH=/opt/runzero/storage
ASSET_BUCKET=runzero-assets
SCAN_BUCKET=runzero-scans
```

- To use AWS S3 as the shared storage, either configure an IAM instance role with read-write access to two S3 buckets, or put the AWS credentials in the configuration file.
  - To set AWS credentials specifically for runZero, use this syntax:

```
AWS_ACCESS_KEY_ID=AKIA…
AWS_SECRET_ACCESS_KEY=[SECRET]
```

- Select S3 storage and name the two buckets:

```
AWS_REGION=<your-region>
RUNZERO_STORAGE_MODE=s3
ASSET_BUCKET=<company>-runzero-assets
SCAN_BUCKET=<company>-runzero-scans
```

For a non-AWS backend that is compatible with the S3 API, use the same AWS and bucket variables, override AWS_REGION, and set AWS_ENDPOINT_URL_S3 to the endpoint. Reach out to runZero support if you run into trouble with the endpoint configuration.

Most AWS CLI environment variables are also available alongside these variables for finer tuning.

Verify the configuration

1. After saving the configuration file, verify that the self-hosted runZero platform can connect to your database:

```
sudo runzeroctl database verify
```

2. Once the settings are configured and verified, restart the self-hosted runZero platform service:

```
sudo systemctl restart runzero-console
```

Create the initial user account

- Run `runzeroctl initial <your-email>` to create the initial user account.

Add the runZero server to the load balancer and verify the connection

- Point the load balancer at the new runZero server's IP on port 80 (unless you changed it).
- Browse to the load balancer URL over HTTPS using the configured DNS name.
- Verify that the user account you just created can sign in.

Installing and configuring subsequent nodes

- Download and install the runZero platform binary on each node as above, with the `--manual-database` option.
- Copy the configuration file from the first node to `/etc/runzero/config` on each subsequent node.
- After saving the configuration file, verify that the self-hosted runZero platform can connect to your database:

```
sudo runzeroctl database verify
```

- Restart the platform service:

```
sudo systemctl restart runzero-console
```

Spread the runZero nodes across multiple availability zones and add them to the load balancer's target group. To confirm that requests reach each node, review the syslog entries on that node.

AWS advanced configuration

runZero can load configuration settings from AWS Secrets Manager at startup. The Secrets Manager entries should match the key names in the configuration file. Set the secret name with:

```
AWS_SECRETS_MANAGER_KEY=rumble/production
```

For runZero to reach Secrets Manager, the configuration file must set AWS_REGION, and the instance needs either an IAM instance role with permission to read the Secrets Manager key or AWS credentials specified directly in the configuration file. With Secrets Manager in use, only the AWS options need to be in the configuration file; runZero loads the rest of the settings from the Secrets Manager key. That means copying the settings from the generated configuration file into the Secrets Manager key-value pairs.

Many settings must be identical across the runZero cluster for web requests and authentication to work, including the session secrets, the DB encryption key, and the console URL.

Self-hosted troubleshooting

The runZero console includes a diagnostics collection script that gathers the performance statistics, log files, system configuration, and profile debug capture needed to troubleshoot a self-hosted environment, then compresses them into one archive you can send to runZero support. Collecting that data by hand means running many different commands and finding many files. The script checks permissions and dependencies first, and it works the same way on a system running your self-hosted console or an Explorer.

Requirements

Run the script on a Linux system with these dependencies installed:

Dependency	To install (Ubuntu)	To install (RHEL, CentOS, Fedora, Oracle Linux)
`zip`	`apt install zip`	`dnf install zip`
`unzip`	`apt install unzip`	`dnf install unzip`
`lsscsi`	`apt install lsscsi`	`dnf install lsscsi`
`iostat`	`apt install sysstat`	`dnf install sysstat`
`curl`	`apt install curl`	`dnf install curl`

Run it as `root` or with `sudo` so it can read the system files and the `/opt/runzero` and `/etc/runzero` directories.

Running the script

Using runzeroctl

To run the script through `runzeroctl`, use:

```
runzeroctl diagnostics run-script
```

Output goes to `/opt/runzero/collector`.

Manually

To run the script by hand, first write a copy of it to the current directory:

```
runzeroctl diagnostics write-script
```

The copy is named `runZero-collector.sh`. Run it with:

```
sudo ./runZero-collector.sh
```

The script also has a `--help` option.

The script performs these tasks:

- Check for the required permissions.
- Check for the dependency commands (lsscsi / zip / unzip / iostat / curl).
- Check that the operating system is Linux.
- Check that either a runZero Explorer or runZero console is installed.
- Collect system configuration information.
- Collect system performance statistics.
- Collect database configuration information.
- Collect some basic statistics from the database.
- Collect runZero debug information from the console or log files from the Explorer.
- Create a zip file of the data collected and the log from the script.

Logging

The script logs every action to a file in `/tmp` and tells you the file name. When it finishes, it moves the log into the `/opt/runzero/collector` directory. Installing a runZero Explorer or runZero console doesn't create that directory, so the script creates it if needed and places all of its files there.

Data retention

You can set data retention periods for each organization. The organization settings page gives you three ways to control how runZero manages your asset and scan data. runZero expires data in a nightly batch job, using the current settings of each organization in your account.

The runZero Platform keeps data for up to 1 year by default. For active subscriptions, the retention period can be increased to a maximum of 3 years. The runZero Community Edition is limited to 30 days of retention.

Stale asset expiration

Stale asset expiration purges out-of-date information from runZero: assets that haven't been seen in the specified number of days are deleted automatically. The deletion task runs once per day.

The limit applies to online and offline assets alike. If an asset was online when last scanned but hasn't been scanned (or imported through a connector) in the specified number of days, its information is stale and runZero deletes it. It's like running the asset search `last_seen:>Xdays` (where X is a number) and deleting the results.

You can set stale asset expiration at both the Organization and Site levels; the lower value wins. To disable it, set the number of days to zero.

Offline asset expiration

Offline asset expiration clears out information about assets that no longer exist on the network: assets that are offline and haven't been seen in the specified number of days are deleted automatically. Online assets are untouched. The deletion task runs once per day.

This setting applies to offline assets discovered by runZero scans. Assets with only connector/import data fall under stale asset expiration.

The asset state (`alive:false`) determines whether an asset is offline. The `last_seen` value then measures the asset's age against the configured threshold; on its own, that value doesn't classify an asset as offline. The cleanup also requires scan-backed assets (`scanned:true`) with a runZero scan source.

This is useful for guest wireless networks or networks with short DHCP leases. It's closest to running the asset search `last_seen:>Xdays AND alive:false AND scanned:true` (where X is a number) and deleting the results.

Stale asset expiration runs separately and can remove the same assets sooner when its threshold is lower.

To disable offline asset expiration, set the number of days to zero.

Stale integration attribute expiration

Stale integration attribute expiration clears out data from integrations that no longer report an asset: attributes the integration hasn't reported in more than the specified number of days are deleted automatically, and assets left with no data sources are removed too. The deletion task runs once per day.

You can keep the latest attribute set per integration, even when it is older than the threshold.

To disable stale integration attribute expiration, set the number of days to zero.

Stale vulnerability expiration

Stale vulnerability expiration removes out-of-date vulnerability data: vulnerabilities not updated in more than the specified number of days are deleted automatically. The deletion task runs once per day.

To disable stale vulnerability expiration, set the number of days to zero.

Scan data expiration

Scan data expiration controls how long runZero keeps the raw data and metadata for Scan and Import tasks. If your organization has a maximum data retention policy for cloud services, use this setting to force the removal of any data older than a given number of days. The default of 365 keeps one year of historical scan data.

Event records

Account-level event records (under Alerts → Events) are always kept for one year. To keep copies longer than that, export and archive them through the API.

AI threads and artifacts

Data from the AI section of the console has fixed retention windows that you can't change:

- **AI threads** (conversations, private to the user who started them) are removed 180 days after their last use. Every new turn resets the clock, so an active thread never expires; renaming one does not.
- **Unshared AI artifacts** are removed with the thread that produced them, attached files included. **Shared artifacts** are kept indefinitely while shared; unsharing one restores the thread-derived expiry.
- **AI run records** (the job and its streamed event log) are temporary: about 15 minutes while a run is active and 24 hours after it ends. The durable results are the thread and its artifacts.

Cleanup runs daily. Users can delete their own threads and artifacts at any time, and deleting a thread removes its unshared artifacts immediately.

Custom integration script versions

Custom integrations keep the most recent 25 versions of their script. Older versions are pruned as new ones are saved, and deleting a custom integration deletes its version history.

Data deletion after account termination

When your runZero Platform subscription ends, runZero deletes your data according to the table below.

Data Type	Default	Maximum
Stale assets (devices that have not received any new data from active, passive, or integration tasks)	180 days, unless requested earlier by emailing support[at]runzero.com	3 years (solely as set by you in runZero Platform prior to account termination)
Task data (raw data associated with each discovery task)	1 year, unless requested earlier by emailing support[at]runzero.com	3 years (solely as set by you in runZero Platform prior to account termination)
All other data (including data contained in support tickets, administrative data, etc.)	Upon written request by you to runZero at support[at]runzero.com	n/a
Any of the above data types archived as a copy on backup systems	1 year from date original data point is deleted	n/a

If you downgrade from a runZero Platform subscription to the free Community Edition, runZero deletes your data according to the table below.

Data Type	Default	Maximum
Stale assets (devices that have not received any new data from active, passive, or integration tasks)	30 days, unless requested earlier by emailing support[at]runzero.com	3 years (solely as set by you in runZero Platform prior to account termination)
Task data (raw data associated with each discovery task)	10-30 days, unless requested earlier by emailing support[at]runzero.com	3 years (solely as set by you in runZero Platform prior to account termination)

All other data (including data contained in support tickets, administrative data, etc.)	Upon written request by you to runZero at support[at]runzero.com	n/a
Any of the above data types archived as a copy on backup systems	1 year	n/a

Managing access

runZero supports multiple concurrent users with different roles. You set roles per user, both as a default and per organization. The built-in roles are administrator, user, billing, annotator, viewer, and no access. A superuser role also exists for managing global settings.

You can also define custom roles that grant exactly the permissions a team needs, and assign them anywhere a built-in role can be assigned.

The console URLs in this article use `console.runzero.com` as an example. Use the URL for your runZero deployment instead.

When a user holds several roles, the most privileged one wins. For example, a user who has user access through a group and admin access assigned directly gets admin privileges.

Available roles

Superuser

The first user created in the runZero console is a superuser. Superusers manage global settings such as subscriptions and SSO parameters, and the console shows their access level as “everything”.

If you are a superuser, you can promote another user: check their row and click the *Promote to superuser* button.

If you use SSO authentication, configure at least one superuser with a strong password and MFA as a backup, so you can still change the SSO settings if you need to.

We strongly recommend more than one superuser, particularly if you use MFA. If an MFA token is lost or a superuser leaves your organization, another superuser can fix the problem.

Administrator

Administrators can modify any aspect of an organization. Only they can permanently delete bulk data, create additional organizations, and reset settings for other users.

User

Users have full access to an organization: they can update sites, modify assets, schedule scans, and use most other functionality. They cannot reset other users’ security credentials, bulk delete data, or delete an organization.

Billing

Billing users manage the licensing, billing, and entity settings for the account. They cannot see any asset data.

Annotator

Annotators have the same permissions as a viewer, plus the ability to add tags to assets. They have no other write access within an organization, so they cannot modify or remove existing tags. Only a runZero user or administrator can change an existing tag.

Viewer

Viewers have read-only access to an organization, including inventory data, findings, queries, and reports. They cannot interact with tasks, modify settings, or update assets. They also cannot install explorers or download the CLI scanner, and they have no access to client API

tokens or organization-level API tokens. Viewers cannot generate some reports, including the Organization overview report, the External assets report, and the Organization and site comparison report. Previously generated versions of these reports remain visible.

No access

The *no access* role is usually set as a default. An account with no access as its default can reach only the organizations you grant it. If you grant none, the user can only manage their own account settings.

The no access global role lets you create a single-organization user, such as a customer or third party who needs access to the inventory of one specific organization. For consulting use cases, a single-organization user gives clients visibility into their own environment at no additional cost.

The *no access* role also works as the account default when you place no limits on who can sign in through your SSO system. New users sign in and request access, and you then grant their newly created account access to the appropriate organizations.

Custom roles

When no built-in role fits, define a custom role with the exact permissions you want. You create custom roles once for the account on the Roles page, then use them as a user's default role, a per-organization role, a group role, the SSO enrollment default, or locked to an API client. Custom roles and permissions covers the permission model and how multiple roles combine.

Inviting users

To add a team member, open the Your team page and click the *Invite user* button to send an invitation.

The **Your team** menu entry has several submenus.

- *Users* shows all users in the current client account.
- *Restricted* lists users who by default have no access to any organization.
- The entry named for the current organization (the one selected in the organization selector at the top of the screen) shows only the users with access to that organization.
- *External* is where you invite users from other runZero client accounts.
- *Groups* lists the available user groups. Use groups to set the access and permissions users have within each organization.

User details

On the Users page, click a user to open their details page, which lists their effective access to each organization in three sections:

- *User access* lists organizations the user was granted directly.
- *Group access* lists organizations the user reaches through membership in a group that has access to them.
- *SSO group access* lists the same kind of group access, for groups assigned through SSO group roles.

To edit a user's directly assigned permissions, click the gear icon at the right of their row in the user listing, or the button at the top right of their user details page.

The gear icon edits the explicitly assigned roles. The user details page shows the resulting access levels.

Account settings

The *Account* page is available to superusers. Its settings apply to all users and organizations within the account.

Single sign-on (SSO)

runZero supports SSO through SAML2. If you use a SAML2-compatible single sign-on (SSO) implementation, configure your Identity Provider (IdP) on the SSO Settings page so permitted users can sign in to the runZero console.

Multi-factor authentication (MFA)

runZero supports multi-factor authentication, also known as two-factor authentication or 2FA. Physical hardware keys such as Google TitanKey and Yubico YubiKey work through the WebAuthn standard.

> ⚠
> The WebAuthn standard does not accept an IP address as the relying party identifier, so you cannot configure multi-factor authentication on a self-hosted console unless you access the console by domain name.

Configure MFA policies for your account on the Account settings page. If you require multi-factor authentication, users without an MFA token must set one up the next time they sign in. You can require it for all users or only for non-SSO users. The second option is useful when your SSO server already enforces MFA.

Once a user registers one or more MFA tokens, they must use one of them every time they sign in.

Changing the account settings to no longer require MFA does not alter the MFA status of existing accounts. Those accounts keep the MFA tokens they registered and must still use one to sign in. To disable MFA for a user, that user must clear their own token registration on their user settings page by clicking the red “Unlink” text next to the token ID in the bottom right.

Disabling support access

Check the box labeled *Disable support access to your account*, and runZero support staff can no longer switch to your account.

Disabling support access may make it harder for runZero support to answer your questions. In some cases we may need you to turn it back on before we can help.

Idle times and sign in duration

You can set a maximum idle session time in minutes. Users whose web browsers don't access runZero for that long are considered idle and signed out.

You can also set a maximum sign in duration. Users then have to sign in again at least that often.

Account API keys

The Account API is a REST API for account-level operations such as adding and removing organizations and sites, adding users, and reading the system event log. The *Generate API Key* button on the Account page creates a token that grants access to the Account API.

Roles

The *Roles* page lists the built-in roles and any custom roles defined for the account, with the permissions each one grants and how many users, groups, and API clients hold it. Superusers and administrators of every organization can create and change custom roles there.

API clients

The *API clients* page manages account-bound API clients: credentials with an explicit set of permissions and organizations, not tied to any user account. Users manage their own user-bound API clients from My API clients in their settings.

License information

The *License* page shows your runZero software license: how many assets you are licensed for, how many assets you have across all organizations, and when the license renews.

Entity information

The *Entity* page is where you update information about the legal entity runZero is licensed to. Keep that information current if your company changes its name or location, because we use it to calculate taxes and to comply with the applicable regulations.

Audit log

The *Audit log* page shows the history of system events relevant to the superuser, such as login events, which the organization Events page does not show. runZero events lists the full catalog of event actions.

Custom roles and permissions

Every role in runZero is a set of permissions. The built-in roles are fixed sets that cover the common cases. A custom role is a set you define yourself, so you can grant exactly the access a team needs instead of rounding up to Administrator or down to Viewer.

You define a custom role once for the whole client account, then assign it anywhere a built-in role can go: as a user's default role, as a per-organization role for a user or a group, as the SSO enrollment default, or locked to an API client.

The console URLs in this article use `console.runzero.com` as an example. Use the URL for your runZero deployment instead.

How permissions work

A permission is a verb applied to an area. An area is a family of records or settings, such as inventory or alerts. A verb says what you may do to it.

Verbs

Verbs are ranked, and a higher verb includes every lower one. Granting `write` on an area also grants `annotate` and `read` on that area.

Verb	What it allows
Read	Read, search, list, view detail, and export existing records. Nothing changes.
Annotate	Change user-supplied metadata on a record you can already read: tags, comments, ownership, criticality, and issue status. It never changes scan-derived facts or configuration.
Write	Create, change, and delete records, and run operations that change data.
Manage	Change the settings that govern an area for everyone else, and create or change credentials, permissions, and organization structure.

Only inventory and findings have anything that can be annotated. Every other area goes straight from read to write.

Areas

Area	Covers	Verbs available
`inventory`	Assets, services, software, certificates, and sites.	read, annotate, write, manage
`findings`	Vulnerabilities and issues.	read, annotate, write, manage
`tasks`	Scan tasks, scan templates, and raw task artifacts.	read, write, manage

`events`	The activity record. Append-only, so read is the only verb.	read
`alerts`	Alert rules, notification channels, and alert templates.	read, write, manage
`explorers`	Explorers, explorer groups, and hosted zones.	read, write, manage
`sources`	Integration sources and custom integrations.	read, write, manage
`queries`	The saved query library, including the Rapid Response feed.	read, write, manage
`directory`	Directory user and group records collected by integrations and scans. Harvested data, so read is the only verb.	read
`users`	Console users, user groups, and invitations.	read, write, manage
`organizations`	Organizations, their settings, and their hierarchy.	read, write, manage
`credentials`	The credentials runZero uses to authenticate scans and integrations.	read, write, manage
`tokens`	API keys, export tokens, and API clients.	read, manage
`ai`	AI Hub, AI threads, and query translation.	read, write, manage
`reports`	Reports, external reporting, and goals.	read, write, manage
`account`	Account settings and license information.	read, manage
`content`	Reference content runZero ships, such as vulnerability match rules and scan checks. Read-only for everyone.	read

Four distinctions are worth reading twice, because the names are similar and the access is not:

- `users` is console identity and `directory` is collected data. `users` covers the people who sign in to your console. `directory` covers the user and group records that scans and integrations bring back from LDAP, Entra ID, and similar sources. Granting one does not grant the other.
- `credentials` is scan credentials and `tokens` is API credentials. `credentials` covers the secrets runZero uses to log in to your systems. `tokens` covers the keys and clients that let something log in to runZero. A credential that can create another credential is a way to escalate privilege, so keep `tokens` restricted.
- `tasks` includes raw scan artifacts. A raw artifact includes the inventory and directory records the scan collected, so `tasks:read` reaches that data without a separate `inventory` or `directory` grant.
- Permission is not the same as visibility. A permission says what you may do; the organizations attached to the assignment say where. Both apply on every request.

What the verbs mean in practice

The same permissions drive the console, the API, and MCP. If a role cannot write inventory, the console hides the buttons that write inventory, the matching API routes refuse the call, and the AI client is not offered the MCP tools that would change inventory. No surface enforces a permission the others skip.

Worked examples for the areas where the line between verbs matters most:

Permission	What it allows
`inventory:read`	Search and view assets, services, software, certificates, and sites, and export them.
`inventory:annotate`	The above, plus tagging assets, setting ownership and criticality, and adding comments.
`inventory:write`	The above, plus editing and deleting assets, and moving them between sites.
`inventory:manage`	The above, plus bulk deletion and the inventory settings that apply to everyone.
`tasks:read`	View scan and integration task history, and download raw task artifacts, which include the inventory and directory data those scans collected.
`tasks:write`	Start, stop, and delete scans.
`tasks:manage`	The above, plus creating and changing scan templates.
`findings:annotate`	Change issue status and suppression state without changing the underlying findings.
`sources:write`	Configure and run integration sources.
`sources:manage`	The above, plus custom integrations.
`users:read`	See who is on the team and what access they have.
`users:manage`	Invite, change, and remove console users and groups. This is the permission that makes someone an administrator of an organization.
`tokens:read`	See which API credentials exist.
`tokens:manage`	Create, change, and revoke API keys, export tokens, and API clients, and approve MCP connections.
`credentials:manage`	Create, change, and delete the credentials used to authenticate scans, and list them through the API and MCP. Secret values are never returned by any permission.

An export always requires the same permission as reading the data in the console. No separate export permission exists to grant or withhold.

What a custom role cannot grant

Some privileges are structural and never part of a role's permissions:

- Superuser: global settings, subscriptions, and SSO configuration.
- Billing: licensing, billing, and entity settings, which stay with the built-in Billing role.
- Account settings: the `account` area is reserved for built-in roles, so a custom role cannot grant it.
- Managing roles: a custom role can never authorize changes to roles, including its own permissions.

What the built-in roles grant

The built-in roles are the same permission model expressed as fixed sets. Use this table as a starting point when deciding which permissions a custom role should include. A dash means no access to that area.

Area	Administrator	User	Annotator	Viewer	Billing
`inventory`	manage	write	annotate	read	-
`findings`	manage	write	annotate	read	-
`tasks`	manage	manage	-	-	-
`events`	read	read	read	read	-
`alerts`	manage	manage	read	read	-
`explorers`	manage	write	read	read	-
`sources`	manage	write	read	read	-
`queries`	manage	read	read	read	-
`directory`	read	read	read	read	-
`users`	manage	read	read	read	-
`organizations`	manage	read	read	read	-
`credentials`	manage	-	-	-	-
`tokens`	manage	read	-	-	-
`ai`	manage	write	write	write	-
`reports`	manage	write	read	read	-
`account`	read	read	read	read	manage
`content`	read	read	read	read	-

Superuser is not in this table because it is a flag on a user rather than a role.

Creating a custom role

Manage custom roles from the **Roles** page under **Account** in the left navigator, at console.runzero.com/account/roles. If the page is missing from your navigator, either you do not administer every organization or custom roles are not enabled for your account. The page lists the built-in roles as read-only rows alongside your custom roles, with the permissions each one grants and, for the people who manage roles, how many users, groups, and API clients hold the role.

Superusers and all-organization administrators can open the Roles page, and they are the only people who can create, change, or delete a role. Everyone else can still read the role catalog, which is how the role dropdowns and effective-access views elsewhere in the console explain what each role allows.

To create one:

1. Click **Create role**.
2. Give it a **Name**. Names are 2 to 64 characters of letters, digits, spaces, underscores, periods, and hyphens, and must start with a letter or digit. They are unique within the account without regard to case, and cannot reuse a built-in role name or label.
3. Add a **Description**. This is the help text shown to whoever assigns the role, so describe what the role is for rather than restating the permissions.
4. Set the **Permissions**. Choose a verb for each area you want the role to grant, and leave the rest at no access. The `account` area is not offered.
5. Save. A role must grant at least one permission.

The copy action on an existing row starts a new role from that role's permissions, which is the quickest way to build "Viewer, plus one thing".

Editing, renaming, and deleting

Editing a role changes access immediately for everyone who holds it, with no separate publish step.

Renaming a role rewrites every assignment that refers to it, including SSO enrollment defaults and API client locks, so nothing is orphaned. runZero refuses to rename onto a name that still has assignments rather than merge the two.

Deleting a role leaves its name on the assignments that used it, but the role stops contributing any permission. Those assignments display as disabled, and effective access views show "No access" and mark the role as disabled. Other roles assigned to the user continue to apply. The delete confirmation lists the users, groups, API clients, and organizations affected.

Plan for two consequences:

- An API client locked to a deleted role cannot authenticate until a role with that name exists again.
- Recreating a role with the same name re-enables every assignment that still refers to it. The create form warns you when a name still has assignments and asks you to confirm.

This is deliberate, so you can undo a mistaken delete, but it means the name itself retains authority. Treat recreating a deleted role name with the same care as granting the access directly.

Assigning custom roles

A custom role appears in every role dropdown that a built-in role appears in, with its description as help text. Nothing about how you assign roles changes.

To a user

On the Your team page, use the gear icon on a user's row, or the button on their user details page, to edit their permissions. You can set a custom role as the user's default role in every organization, or per organization.

The user details page shows effective access per organization, split into direct user access, group access, and SSO group access, so you can see what a role resolves to for that person.

To a group

User groups take custom roles the same way, either as the group's default role in every organization or per organization. Group membership can come from the console or from SSO group mappings.

As the SSO enrollment default

The **Default role in any organization** field on the SSO settings page accepts custom roles as well as built-in ones. Users provisioned through SSO enroll with that role.

If the role was deleted by the time a new user enrolls, they enroll with no access rather than inheriting something unintended, and runZero records the event. Recreating the role does not retroactively change users who already enrolled, though it does re-enable their stored assignment.

To an API client

You can lock an API client to a role, so the credential's access follows later changes to that role, or use a role as the starting point for a one-off set of permissions. See Role locks.

How multiple roles combine

A user can hold roles through several routes at once: their own default role, a per-organization role, a group's default role, and a group's per-organization role.

Permissions add together. In a given organization, the user gets the best built-in role that applies plus every custom role that applies, unioned. Where two roles grant different verbs on

the same area, the higher one wins. A custom role can only add access, and never reduces what a built-in role already grants. If someone has Viewer by default and a custom role granting `tasks:write` in one organization, they can start scans in that organization and remain a viewer everywhere else.

A custom role can bring an organization into view. If a custom role is the only thing granting a user access to an organization, that organization appears in their organization picker and in their effective access.

Custom roles never confer structural privileges. No combination of custom roles makes someone a superuser, gives them billing access, or lets them manage roles. See What a custom role cannot grant.

Each role slot contains one role. Assigning a new role to a user's default slot replaces the role that was there. To give someone several distinct sets of permissions, combine a default role with per-organization roles, or with group membership.

Auditing

The audit log records role changes as `role-created`, `role-updated`, and `role-deleted` events. An update records which fields changed, including the previous name and permissions. A delete records how many assignments it disabled.

On the Roles page, the usage columns and the usage view list the users, groups, and API clients assigned each role, and whether it is the SSO enrollment default.

Managing user groups

User groups let you treat users who need the same set of permissions as one unit. A user group explicitly sets the organizational role for its members, which determines what they can do within each organization. You can assign both built-in roles and custom roles to a group, either per organization or as a single role across all organizations. SSO group mappings can also apply single sign-on settings to groups.

> ⚠
> **Conflicting permissions**
>
> runZero always grants the role with the highest permissions level. For example, an account with a viewer role for all organizations joins a user group that has a user role for all organizations. That user now has user-level permissions for all organizations. If the user group expires, the user's role reverts to their account-level role.

A user group can also have an expiration date, which gives specific users time-bound access to organizations within runZero. When the date passes, the user reverts to their account-level permissions. Without an expiration date, the user group settings stay in effect indefinitely.

Creating user groups

runZero administrators and superusers can create user groups.

1. Go to Your team > Groups and click Add Group.
2. Enter a name for the user group.
3. Choose the default role for the group. This sets the access level for every organization you have.
4. Set per-organization roles if specific organizations need different access levels.
5. Set an expiration date if the permissions should be time-bound. When the date passes, group members revert to their user-level permissions. Without an expiration date, the group stays in effect indefinitely.
6. Go to the **Add users** tab and search for the users you want to add, by username or email.
7. Save the user group. It appears on the Groups page.

Adding users to user groups

To add users to several groups at once, use the *Edit group membership* button on the Users page. The *Edit group membership* window lists every user group each selected user is in, and changes you make there apply to all selected users. Only runZero administrators and superusers can add users to user groups.

1. Go to the Users page.
2. Select the users you want to add to a group.

3. Click the **Edit group membership** button.
4. Choose the user groups to add the users to.
5. Save your changes.

Setting an expiration date for a user group

runZero administrators and superusers can set an expiration date for a user group.

1. Go to Your team > Groups.
2. Find the user group and click its name to open the config page.
3. Set the expiration date. After it passes, members revert to their account-level permissions.
4. Save the user group. The Groups page shows the expiration date.

Viewing users in a user group

1. Go to Your team > Groups.
2. Find the *Users* column in the *User groups* table, which shows each group's user count.
3. Click the count to list the users assigned to the group.

Viewing user groups assigned to a user

1. Go to Your team > Users.
2. The Groups column shows how many groups each user belongs to.
3. Click a number to list those groups.

Removing users from a user group

runZero administrators and superusers can remove users from a user group.

1. Go to Your team > Groups.
2. Find the user group and click its name to open the config page.
3. Go to the *Users* tab.
4. Remove the users you no longer want in the group. Their permissions revert to their account-level ones.
5. Save your changes.

Deleting user groups

runZero administrators and superusers can delete user groups.

1. Go to Your team > User groups.
2. Select the user group you want to delete.
3. Click the **Delete** button.
4. Confirm you want to delete the user group.

Searching for users and user groups

On the Users page or Groups page, you can search the table with these keywords:

Keyword	Description	Example
`id`	User's ID.	`id:123456789`
`name`	User's name.	`name:john`
`expires_at`	Time or date the user group expires.	`expires_at:>2weeks`
`created_at`	Time or date the user group was created.	`created_at:>2weeks`
`updated_at`	Time or date the user group was last updated.	`updated_at:>1year`
`has_expiration`	Whether the group has an expiration date.	`has_expiration:true`
`created_by_id`	ID of user who created the user group.	`created_by_id:123456789`
`created_by_email`	Email of the user who created the user group.	`created_by_email:user@example.com`
`group_id`	The user group ID.	`group_id:123456789`
`group_name`	The user group's name.	`group_name:group1`

The group ID appears in the URL of the group config page, `https://console.runzero.com/groups/<groupid>/edit`.

The `group_id` keyword is only available for the users table; for the groups table, use `id`.

Bulk importing users

runZero administrators can add multiple users at once with a bulk import instead of adding them one at a time. To bulk import users, create a CSV (comma separated values) file with the user information, such as each user's first name, last name, email, role, and organizational access.

A bulk import only adds new users; it does not update existing ones. If the file contains a user who already exists in the system, the import does not complete. Remove all duplicate users from your CSV file and import it again.

The high-level steps:

1. Create a CSV file with your user information.
2. Import the CSV file into runZero.
3. Verify users have registered their accounts.

Creating the CSV file for importing users

Start from the provided template or create the CSV file from scratch. Either way, follow these guidelines and include the fields below.

- The first row is the header row, and the file must include it. It names the fields, which must follow this order: `first_name`, `last_name`, `email`, `all_orgs_role`, `org_roles`.
- runZero requires the `first_name`, `last_name`, and `email` fields. The organization-level fields, `all_orgs_role` and `org_roles`, are optional.
- Write `org_roles` values as `orgName=role`, and separate multiple definitions with pipes.
- If you leave out `all_orgs_role`, users get the no-access role in all organizations.
- If you leave out `orgs_role`, users get the role in `all_orgs_role`. If you leave out both, every user gets the no-access role and must contact the runZero administrator to change their account access.

User fields for the CSV file

The fields you can include:

Field	Description	Example
`first_name`	The user's first name.	`sarah`
`last_name`	The user's last name.	`smith`
`email`	The user's email address.	`ssmith@company.org`
`all_orgs_role`	The role assigned to the user for all organizations. This is optional.	user
`org_roles`	The role assigned to the user for a specific site. Use pipes to separate the organizations. This field is optional.	org1=user

The resulting CSV file looks like this:

```
first_name,last_name, email,all_orgs_role,orgs_role
Sarah,Smith,ssmith@company.org,user,org1=annotator|org2=annotator
```

Importing users into runZero

1. Go to your Team Import page.
2. Choose the CSV file with your user information and upload it.
3. Import the file.

runZero alerts you to problems with the file. If the file contains an existing user, for example, the import does not complete until you remove all duplicate users and import it again.

Verifying users have registered

Each imported user receives an email invitation to join your team. The email contains a link to register their email address, which prompts them for the name and password for their account. Once the account exists, they can sign in to the runZero console.

The Teams page shows each user's status and last sign in. Users who completed registration show an *Activated* status.

Managing external users

You can invite external users into your runZero instance, where they can view the organizational data you make available to them. External users are useful for consultants, value-added resellers, and managed service providers who want to share runZero data with partners and clients.

If you are a superuser, you can invite any user with a runZero account to join yours. They receive the invitation by email. After they accept it and sign in to runZero, a new menu next to their organization switcher lists every client they can access, including yours.



If the person you want to invite has no runZero client account, they need to sign up for one before you can invite them as an external user. A free runZero Community Edition account qualifies.

External users appear only on the External users page, separate from your other user lists. They sign in with their existing credentials and switch between the clients and organizations they have access to.

Inviting an external user into your account

1. Go to **Your team > External** and click Invite external users.
2. Choose the default role the user gets in all organizations.
3. Enter the user's email address. The user must already have a runZero account.
4. Customize the user's access per organization. When the global and per-organization permissions conflict, runZero grants the higher one.
5. Add a custom email subject or body if you want. The default template tells the user you have invited them to your account and includes a link to activate the invitation. The sender is your name and the subject line is "Invitation to join [client] by [your name]."
6. Send the email.

Once they activate the invitation, they can sign in to runZero, switch clients, and view your organizational data.

Switching clients and organizations as an external user

External users with access to another account see a client switcher (a client dropdown menu) next to their organization switcher at the top right of the runZero web interface. Changing the client changes the organizations they can access and view. The account that granted them access decides which organizations and permissions they have within that client account.

Removing an external user from your account

Removing an external user takes away their access to your data, and they can no longer choose your client. Go to the External users page, select the user account, click **Remove user account**, and confirm. The user returns to their primary client account and can no longer reach your runZero data.

Implementing SSO

If you use a SAML2-compatible single sign-on (SSO) implementation, configure your SSO Identity Provider (IdP) on the SSO Settings page so permitted users can sign in to the runZero console.

runZero's SSO implementation is designed for common SAML providers and needs minimal configuration. A few requirements apply:

- Your users authenticate to a single domain such as `example.com`, not to multiple domains or a domain with many subdomains.
- You configure that domain name in the SSO identity provider settings in runZero. This applies even to self-hosted runZero deployments.
- Your SAML IdP should send something that looks like an email address in the standard NameID parameter. It doesn't need to be a valid email address, but it should be a unique value with the syntax of one (`user@example.com`). The field name `NameID` is case sensitive.
- If the `NameID` does not look like an email address, runZero checks the fields `email`, `user.email`, `emailaddress` and `email address` for a suitable ID. These field names are not case sensitive.
- runZero looks for the user's full name in the fields `name`, `gecos`, `user.name` and `displayname`. If none is present, it looks for a first name in `first_name`, `firstname`, `given_name`, `user.firstname`, `givenname` or `first name`, and for a last name in `last_name`, `lastname`, `family_name`, `user.lastname`, `surname`, `sn`, or `last name`. These field names are not case sensitive.

You must be a superuser to manage runZero SSO settings.



Once a user signs in with SSO, they can no longer sign in with a password, even if they could before. We strongly recommend that you set up and keep a non-SSO superuser account, so you can update the settings if SSO stops working for any reason.

Specific SSO providers

The Azure AD and Okta pages cover those providers in detail. The basic configuration steps are:

1. Add runZero as an application.
2. Set up SSO in runZero.
3. Provision users to the runZero app.

For other SSO providers, the following sections explain the settings you need.

Identity provider settings

To open the identity provider settings, choose *Your team* in the left navigator, then click the SSO Settings button at the top right of the page.

The identity provider settings form takes the details of your SSO identity provider.

The easiest way to configure SSO is with XML metadata, if your SSO service provides it. You usually set up runZero as an application on your SSO provider, then download an XML file. Open the XML in a text editor and paste it into the box at the bottom of the runZero identity provider settings form. If the XML decodes successfully, the form fills in the main fields for you.

Single sign on mode

Choose whether to allow regular runZero accounts, SSO-provisioned accounts, or both. Allowing both lets you keep backup administrator accounts in case your SSO provider is unavailable.

Domain name

runZero uses the domain name to find the correct SSO IdP settings when a user chooses to sign in via SSO, so it can redirect them to the right SSO provider. For example, if the domain is `example.com`, the runZero sign-in page expects users to start by entering an address of the form `username@example.com`.

The SSO provider also passes the domain back after sign in, and runZero uses it to fetch the SSO IdP settings it needs to verify and process the data.

The domain is part of the ACS URL in the sign in request sent to the SSO provider. If you change the domain, you need to reconfigure the SSO provider to accept the new ACS URL.

Default role in any organization

The default role given to users the first time they sign in through SSO. It accepts the built-in roles, any custom role defined for the account, and Superuser.

No access is a common choice when anyone in your identity provider can sign in. New users then start with no organization access, and you grant access to the appropriate organizations after they first sign in.

If a custom role set here is deleted, users who enroll afterwards are created with no access rather than inheriting something unintended. Recreating the role restores the assignment for those users.

Roles assigned through SSO group mappings apply on top of this default.

Issuer URL

The issuer URL, often called the entity ID, is a URL your SSO provider uses to identify itself in data passed back to runZero. Get this value from your SSO provider. runZero does not normalize trailing slashes, so the value must match exactly.

Sign in URL

The sign-in URL is where runZero redirects users to begin the sign-in process at your SSO provider.

Certificate

The certificate is the PEM encoded CA certificate runZero uses to verify the signature on the data from your SSO provider.

If you paste multiple CA certificates into the box, your SSO provider needs to include a `KeyInfo` element in the data it passes back, to say which certificate runZero should check the signature against.

SSO walkthrough

Configuring and debugging SSO settings is easier when you know how the SAML sign-in process works.

SAML SSO authentication runs on messages passed through the user's web browser. runZero and the SSO identity provider (IdP) never connect to each other directly, in either direction.

A normal sign-in starts when the user opens the runZero console sign-in page, chooses **Sign in via SSO**, and enters their email address. runZero uses the domain name of that address to find the correct SSO identity provider settings in its database.

runZero then redirects the user's web browser to the Sign in URL from those settings, adding parameters to the URL as the SAML specifications require. The SSO IdP verifies those parameters.

The user signs in on the SSO IdP website. The IdP then redirects the browser back to runZero, to the Assertion Consumer URL (ACS URL), along with a digitally signed blob of encoded data about the authenticated user. The ACS URL appears on the service provider information tab of the SSO settings. It includes the domain name, which is how runZero finds the correct set of SSO settings to process the data.

runZero decodes the data and checks its digital signature against the certificate from the identity provider settings. If the certificate is valid, runZero trusts the decoded information about the user, provisions a runZero user account if necessary, and signs the user in to a new runZero session.

Service provider information

After you fill out the identity provider settings, switch to the service provider information tab for the values your SSO identity provider needs from runZero.

The Assertion Consumer URL, sometimes called the SSO URL, is where the SSO identity provider redirects a person's web browser after they sign in successfully. The URL contains the domain from the identity provider settings, which runZero uses to find the right set of SSO settings and verify the data from the SSO provider. If you change the domain in the identity provider settings for any reason, the ACS URL changes, and your SSO provider needs the new URL.

The user sign-in URL is a URL you can visit to begin the sign-in process. It displays the computed identity provider URL to help with debugging.

Common problems

A user signs in on the SSO provider, but runZero refuses the sign in

The most common cause is an Assertion Consumer URL on the IdP that is missing the domain name set in the runZero SSO identity provider settings, or that has the wrong domain name.

A correct ACS URL looks like `https://console.runzero.com/auth/example.com/saml20/process`, where `example.com` is the domain name configured in the runZero SSO identity provider settings.

Missing x509 Element

The full error message is "invalid SAML response: error validating response: Missing x509 Element".

This error can occur if you entered multiple CA certificates in the certificate field of the runZero SSO settings but your SSO IdP didn't include a `KeyInfo` element saying which certificate to check the signature against. Either configure your IdP to send `KeyInfo` naming the CA certificate to use, or use a single CA certificate.

Managing SSO group mappings

⚠
Only runZero administrators can map users to user groups automatically with SSO attributes and custom rules.

SSO group mapping maps your SAML attributes to user groups in runZero. A user group explicitly sets the organizational role, which determines what users can do within each organization. To set up a mapping, you name the SSO attribute and the value to match. When a user's attribute matches, runZero applies the group settings to that user, so SSO users land in the right runZero groups.

For example, your IT team may need administrator privileges. Create a user group with the administrator role, then create an SSO group mapping from the SAML attribute that identifies your IT team to that group. When someone from the IT team signs in to runZero, they get the appropriate access and permissions automatically, with no pre-provisioning of their account. After evaluating all SSO group mapping rules, runZero grants the user the highest privilege assigned for each organization. Where custom roles are involved, the permissions of every role that applies add together.

Creating SSO group mappings

Before you create an SSO group mapping, set up SSO for your organization and create user groups, because a mapping needs both.

Only runZero superusers can create SSO group mappings.

1. Go to Your team > SSO settings > Group mappings > Add group mapping.
2. In the **SSO attribute** field, enter the attribute to check for matching values. Your SSO configuration defines these values.
    - For Azure AD SSO, the **SSO attribute** field must match the claim name from Azure AD.
3. In the **SSO value** field, enter a comma separated list of values the attribute can match. On a match, runZero applies the group permissions.
4. Click the **Group** dropdown and choose the user group to assign on a match. The dropdown lists every user group you have created.
5. Save the SSO group mapping. It applies the next time the user signs in to runZero.

Changes do not reach users who are currently signed in until they sign out and back in. To apply the SSO group mappings immediately, you can force users to sign out.

Forcing a user to sign out

Permission changes take effect only after the user signs out and back in to runZero. Forcing users to sign out ends their current session, so their updated permissions apply as soon as they sign back in. Only superusers and admins with access to all organizations can force sign-outs.

Go to the Teams page, select the users you want to sign out, and click the sign-out button.

Viewing SSO group mappings

The Group mappings page lists every SSO group mapping. From there you can create, edit, or delete mappings.

Viewing SSO group mapping assignments

To see which SSO groups a user belongs to, go to the Users page. The *Groups* column shows the number of user groups and SSO groups for each user, with the SSO group count in

parentheses.

Click the gear icon under actions to open the user's settings. The access summary tab lists all of their organizations and roles.

Deleting a group mapping

1. Go to the Group mappings page.
2. Select the group you want to delete and click the **Delete** button. All users provisioned through the mapping revert to their account-level permissions.

Searching for SSO group mappings

On the Group mappings page, you can search the table with these keywords:

Keyword	Description	Example
`id`	User's ID.	`id:123456789`
`sso_attribute`	User's SSO attribute.	`sso_attribute:department`
`sso_value`	User SSO attribute value.	`sso_value:securityteam`
`created_at`	Time or date user group was created.	`created_at:>2weeks`
`updated_at`	Time or date user group was last updated.	`updated_at:>1year`
`created_by_email`	Email of user who created the group.	`created_by_email:user@example.com`
`group_id`	User group ID.	`group_id:123456789`
`group_name`	User group's name.	`group_name:group1`

To find a group ID, go to the group config page and enable the ID column from the Columns menu above the data grid.

The `group_id` keyword is only available for the Users table; for the groups table, use `id`.

The `group_name` keyword is only available for the Users table.

Setting up Microsoft Entra SSO

Superusers can configure single sign-on to the runZero Console through an external identity provider (IdP), so your single sign-on (SSO) solution handles authentication and user access control for the runZero Console. SSO is available by default but not required to sign in to the console. You can make it required or disable it completely.

The high-level steps to set up Microsoft Entra (formerly Azure AD) SSO for runZero:

1. Add and configure runZero as an Entra app.
2. Download the SSO configuration metadata in XML format.

3. Set up SSO in runZero.
4. Add users to your runZero app in Entra ID.

Requirements

Before you set up SSO for the Microsoft Identity Platform:

- Verify that you have administrator privileges for Azure AD.
- Verify that you are a superuser in runZero. Look for the yellow star in your account status.

Step 1: Add and configure runZero as an Azure app

First, add runZero as a non-gallery application to your Azure AD setup and configure the settings for runZero as an Azure AD application.

1. In Azure, go to **Enterprise Applications > New Application > Create your own application**.
2. Under the **What are you looking to do with your application?** section, choose the **Non-gallery application** option.
3. Name your application something like `runZero` and add it.
4. Go to **Azure Active Directory > Enterprise applications** and open the new runZero application.
5. Select the **Single sign-on** tab and choose **SAML** as the sign-on method.
6. For the fields on the **Configure App Settings** page, go to https://console.runzero.com/team/sso/sp and copy the service provider details:
    - Entity ID
    - Single sign-on URL
    - SSO callback (ACS) URL
7. Enter the values in the matching fields in the Azure AD portal.
8. **Do not** set a value for “Sign on URL (Optional)” or “Relay State (Optional)”.
9. To use group information from Azure for group membership in runZero, configure the IdP to send group information.

Step 2: Download the SSO configuration metadata

While editing your application settings, download the SSO configuration metadata in XML. You need it to set up SSO in runZero.

1. On the **Configure App Settings** page, find the **SAML Signing Certificate** section.
2. Locate the XML download link under the Federation Metadata URL.
3. Download the file. The next step uses its contents.

Step 3: Set up Entra ID SSO in runZero

With the metadata XML in hand, configure the Entra ID SSO settings in runZero.

1. Go to https://console.runzero.com/team/sso/idp to open the SSO IdP provider settings page in runZero.
2. Choose a mode to enable SSO:
    - **Allowed** enables SSO, but users can still sign in without it.
    - **Required** makes users sign in with SSO. Only superusers can sign in without SSO.
3. Enter the domain name associated with SSO authentication. This is likely your company domain (companyabc.com).
4. Choose a default role for SSO users. This is the role every new user gets when their account is created.
5. Copy the XML you downloaded from Azure and paste it into the Metadata XML field on the runZero SSO IdP page.
6. Apply your SSO settings. The remaining IdP fields fill in automatically.
    - The issuer URL looks something like `https://sts.windows.net/00000000-0000-0000-0000-000000000000/`, where the UUID at the end is your unique Microsoft Active Directory (tenant) ID, listed under **App registrations > Overview > Endpoints**.
    - The sign-in URL looks something like `https://login.microsoftonline.com/00000000-0000-0000-0000-000000000000/saml2`, with the zero UUID replaced by your unique tenant ID.
    - The certificate is Microsoft's PEM encoded certificate, extracted automatically from the XML.
    - On the Microsoft side, the redirection URL for runZero should be `https://console.runzero.com/auth/<domain>/saml20/process`, where <domain> is the domain specified in the runZero SSO settings.

Step 4: Add users to the runZero app in Azure

With setup complete, go to the runZero app in the Azure portal to add users and assign their access. Users you add to the runZero app appear on the Team members page in runZero once they have signed in.

Step 5: Update SSO group mappings to match any configured Azure groups (if applicable)

If you created user groups within Azure, update your SSO group mappings in runZero to associate those Azure groups with runZero user groups. That way your users get the appropriate access and permissions when they sign in to runZero.

When you set up the SSO group mappings, the **SSO attribute** field must match the claim name from Entra ID.

Setting up Okta SSO

Superusers can set up single sign-on to the runZero Console through an external SAML identity provider (IdP) such as Okta. The IdP then handles authentication and user access control, and your users sign in without typing credentials.

Setting up single sign-on (SSO) with Okta takes these high-level steps:

- Add runZero as an application in Okta.
- Set up SSO in runZero.
- Add users to the runZero app in Okta.

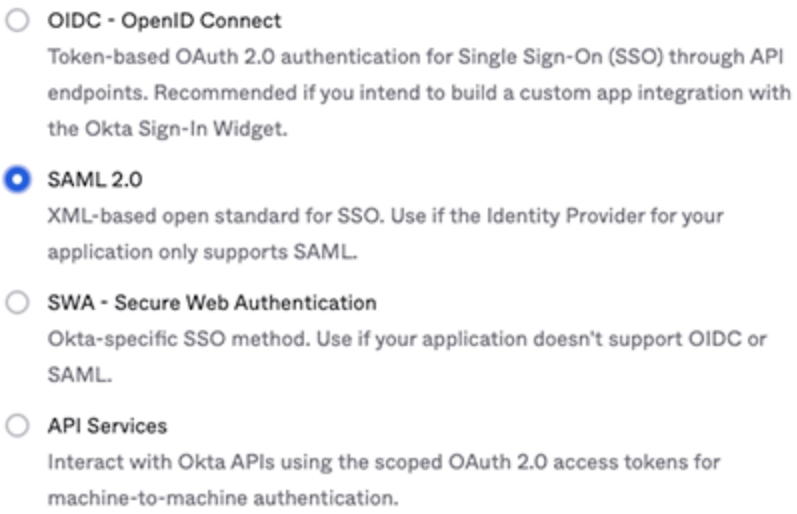
Requirements

Before you can set up Okta SAML:

- Confirm that you have administrator privileges for Okta.
- Confirm that you are a superuser in runZero. Look for the yellow star in your account status.

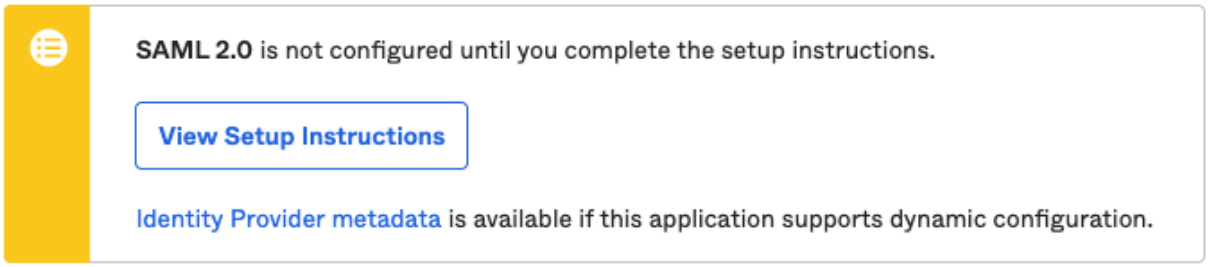
Step 1: Add and configure runZero as an Okta app

1. Go to **Okta > Applications > Create App Integration**. In the **Create a new app integration** window, select `SAML 2.0` as your sign-in method.



2. In the general settings, name the app `runZero`. You can also add a logo so users can pick out the runZero app.
3. For the SAML settings, open the service provider information page in runZero to find the URLs you need.
    - **Single sign on URL** is the assertion consumer service (ACS) URL shown in runZero.
    - **Audience URI or SIP Entity ID** is the runZero entity ID, or SAML audience: `https://console.runzero.com`.
        - Include the leading 'https://' when you enter this field in Okta.
4. For the remaining settings, such as the attribute statements, the Okta documentation explains how to configure them.

5. Once the SAML settings are done, Okta asks for feedback on how you will use the app. Answer or skip it to finish creating the app.
6. After Okta creates the app, go to the **Sign On** tab for the runZero app and open the SAML 2.0 instructions. You'll need these details in the next step.



Step 2: Set up SSO in runZero

1. Go to the SSO setup page in runZero and choose a mode to enable SSO:
   - **Allowed** enables SSO but still lets users sign in without it.
   - **Required** makes users sign in with SSO. Only superusers can sign in without it.
2. Enter the domain name associated with SSO authentication. This is likely your company domain (companyabc.com).
3. Choose a default role for SSO users. runZero assigns this role to every new user when it creates their account.
4. Copy the fields from Okta into runZero.
   - **Issuer URL** takes the **Identity Provider Issuer URL** from Okta. It looks something like `http://www.okta.com/<ID>`.
   - **Sign-in URL** takes the **Identity Provider Single Sign-On URL** from Okta. It looks something like `http://<okta-instance>/app/<app-name>/<ID>/sso/saml`.
   - **Certificate** holds the X.509 certificate from Okta. Paste its entire contents.
5. Apply your SSO settings.

Step 3: Add users to the runZero app in Okta

With setup complete, go to the runZero app in Okta to add and manage user access. Once added, your users can reach runZero from your SSO sign-in URL.

Step 4: Update SSO group mappings to match any configured Okta groups (if applicable)

If you created user groups in Okta, update your SSO group mappings in runZero to map each Okta group to a runZero user group. Your users then get the right access and permissions when they sign in to runZero.

Managing licenses

As a runZero superuser or billing user, you can view and manage your organization's licensing, plan, and billing information.

If you aren't a superuser or billing user, ask your organization's superuser or billing user for help with licensing and billing. These users are tagged with a yellow star.

How do I view my license?

As a superuser or billing user, go to Account > License to view your runZero licensing information.

The License page shows:

- How many organizations, sites, Explorers, and user accounts you have.
- How many recent assets you have across all organizations.
- How many project assets you have.
- How many recent assets and project assets your license supports in total.

What counts as recent assets?

runZero counts how many assets were seen during the last 30 days. If a scan has seen an asset in the last 30 days, it counts as recent. If third party data reports the device was seen in the last 30 days, it also counts.

License limits are the same for every asset type. Online or offline, scanned or unscanned, and whatever the source of the data, the asset counts the same way. When an asset has multiple data sources with different last-seen dates, the most recent date applies.

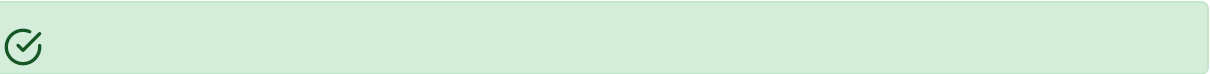
runZero recalculates the recent asset count in the background, so after you delete assets the number may take a short while to update.

When does my subscription expire?

Go to Account > License to see when your subscription or license expires.

Find the line: `This is a runZero [edition] subscription that expires at [date and time].`

An expired subscription shows `This is a runZero [edition] subscription that expired on [date and time].` instead, and you can no longer run discovery scans. To keep using runZero, renew your subscription.



The Community license never expires. It's part of our free tier, and you can keep using it as long as you have no more than 100 recent assets.

How do I renew my subscription?

To renew, contact us by email or through our online form.

How do I convert to the Community Edition?

If you no longer need the full platform, you can convert it to the Community Edition. This plan is on our free tier and supports up to 100 recent assets.

How do I find my invoices?

Superusers and billing users can view all invoices in the runZero Console on the License page.

How do I change or cancel my subscription?

Contact us to change or cancel your subscription.

Network discovery

runZero gathers asset data through unauthenticated active scanning, passive traffic sampling, and inbound integrations.

Active scanning

The runZero Explorer and scanner run unauthenticated active scanning against the networks you specify, using the configuration you set. They use several network protocols to discover and fingerprint the assets connected to the network.

You can run an active scan once or on a schedule, and scan templates keep the settings consistent across scan tasks.

Traffic sampling

An Explorer can passively sample traffic in your environment to identify the assets communicating across your networks. You configure traffic sampling per Explorer on the Explorer details page. When a scan task arrives, the Explorer automatically pauses traffic sampling to handle it, then resumes parsing sniffed traffic once the scan completes.

Traffic sampling can also ingest a PCAP file and pull asset data from it.

Inbound integrations

Inbound integrations with many platforms enrich your runZero asset context and give you insight into coverage and capabilities. You configure an integration as either a connector or a scan task.

Active scanning

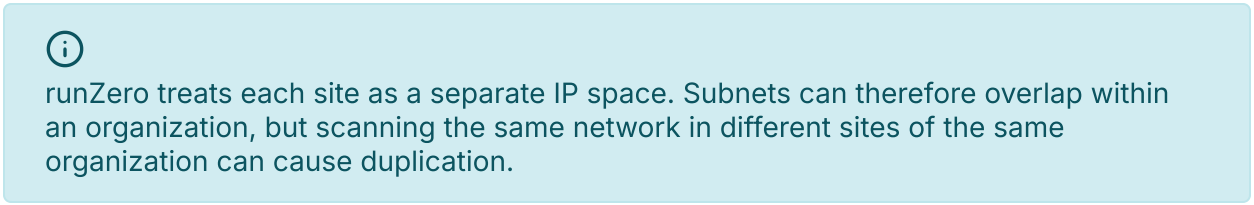
An active scan finds every responsive device on a network, fingerprints each one, and fills the asset, services, screenshot, and software inventory. Regular scans of internal and external networks are a basic part of network management. You configure a scan by site, Explorer, and scan scope, and the scope can include IP ranges, domain names, ASNs, and even entire country codes.

A new scan has many parameters you can set, from a schedule to the advanced options below. To start one, log in to the runZero Console, select Scan from the Data sources section of the navigation menu, and choose “Start Standard Scan”. You can also launch scans from the Inventory views.

Site

runZero organizes information into organizations and sites. An organization is a distinct entity that keeps its data separate and holds a collection of sites. Sites model segmented networks, particularly independent networks that use the same private IP address ranges.

For example, you might have several physical locations, each with its own local network on the 10.0.0.0/8 private IP range. Define each one as a site with its own Explorer, and runZero treats the networks and assets as completely independent, even when similar systems appear at the same IP addresses in each.



Because scan analysis happens at the site level, the boundaries you define for a site become the default scope for its scans.

Explorer

From the Explorers registered to the site, select the Explorer that will run the scan. It must be able to communicate directly with the networks and addresses in your discovery scope, and ideally it reaches every address in the scope with no firewall in the way. Stateful firewalls and VPN gateways may interfere with discovery.

Hosted External Explorer

runZero Platform users can scan public IP space with runZero-hosted Explorers. When you create a scan, choose a Hosted External Explorer in the ‘Run task with’ dropdown. With this

option, the discovery scope must use public IP addresses or ranges, or resolve to public IP space.

Discovery scope

The discovery scope defines which IP addresses the scan covers. The keyword “defaults” uses the site settings, and you can change the scope per scan as well. The scope should include at least one IP address or hostname. You can write IPv4 address ranges in most standard formats:

- `10.0.0.1`
- `10.0.0.0/24`
- `10.0.0.0/255.255.255.0`
- `10.0.0.1-10.0.0.255`

You can specify individual IPv6 addresses, but not IPv6 ranges.

Hostnames in the scope are resolved at runtime by the assigned Explorer. If a hostname resolves to multiple IP addresses, the scan covers all of them. A hostname can also carry a mask, which expands around each resolved address. For example, if `example.com` resolves to both `1.2.3.4` and `5.6.7.8`, the input `example.com/24` becomes `1.2.3.0/24` and `5.6.7.0/24`. IPv6 addresses returned from hostname resolution are scanned if the Explorer has a valid IPv6 address and a route to the target.

runZero load balances the scan across as many subnets as possible, in a quasi-random order.

Discovery keywords

These keywords work in both scan scopes and exclusions.

- **asn4** selects the IPv4 ranges associated with a given AS number: `asn4:<AS number>`.
- **country4** selects the IPv4 ranges associated with a given two-character country code: `country4:<ISO code>`.
- **public** and **private** select the IPv4 and IPv6 addresses of assets in the current organization: `public:<mode>` and `private:<mode>`. Set the mode to `all`, `primary`, or `secondary` to choose which IP addresses are used. The `public` keyword selects all non-reserved IP addresses on organization assets, and the `private` keyword selects all RFC-1918 and private use IP addresses on organization assets.
- **domain** automatically selects the publicly-known hostnames for a given domain name: `domain:<domain name>`. The `domain:<domain>` keyword is available to cloud-hosted users.

Scan name

Give your scan task a name so it’s easier to track.

Scan speed

Set the maximum packet rate for the whole discovery process, in network packets per second. 500 is conservative, 3000 works for most LANs including WiFi, and 10000 or more may help large sites with fast connectivity.

Scan speed directly affects how long the scan takes. An approximate formula:

    time in seconds = hosts x ports x attempts / scan speed

The discovery scope mostly determines the number of hosts. The number of ports is around 500 by default, and the scan makes three connection attempts.

The advanced scan options can change the number of hosts and ports scanned, and the maximum host rate and group size can also affect speed; see the advanced scan options below.

The formula leaves out the time spent taking screenshots, following web server redirects, and processing the scan data.

Schedule

You can set a date and frequency for your scan task. Dates and times follow your browser’s advertised timezone.

A scan scheduled to start in the past launches immediately, then repeats at the specified time at the frequency you selected.

Scheduling grace period

Set how many hours the scan waits for an available Explorer before giving up. With a zero or negative value, the scan retries indefinitely until an Explorer becomes available.

Scan duration limit

Set a limit in hours on scan duration; if scanning is still in progress when the time runs out, runZero cancels the scan. This limit does not cover processing time.

If you set this to 0, runZero applies no limit.

Advanced scan options

On the Advanced tab you can view and change more scan settings, such as network exclusions, scan speed, the ports the TCP scan covers, and which probes are enabled. The defaults should work for most organizations but may need tuning for slow networks or unreliable links.

Maximum host rate

Beyond the overall scan rate in packets per second, you can also cap the rate at which packets go to any single host IP address. This helps when you have devices that are quick to overload under network traffic. The default should be safe for most systems.

Max group size

When runZero scans your network, it spreads the load across many IP addresses at once. The max group size sets how many IP addresses can be actively scanned at the same time, allowing for hosts that take a while to respond to probes. It needs to be at least as large as the overall scan speed, or it would hold the scan below the speed you set. If you enter a value lower than the overall scan speed, runZero raises it automatically at scan time.

Max group size is most useful with stateful network devices that can track only a limited number of connections at once, since it restricts how many active TCP sessions a runZero scan creates.

Max TTL

The IP standards define a maximum hop count for packets: the Time To Live or TTL in IPv4, and the Hop Limit in IPv6. Every device that processes a packet must decrease the TTL or Hop Limit by one, and when the value reaches zero, the router receiving the packet must discard it. Max TTL sets the maximum hop limit for scan traffic.

ToS

The IP standards define a Type of Service or ToS for packets. IPv4 calls it the Type of Service or ToS, and IPv6 calls it the Traffic Class or TC. Switches and routers use the ToS or Traffic Class to prioritize network traffic, and the lower bits of the IPv4 ToS are also used for congestion control. This option sets the ToS or Traffic Class for scan traffic. The ToS/Traffic Class setting applies only to the basic discovery probes, not to all traffic runZero sends. Some protocols, such as SNMP, and integrations, such as VMware, do not set the ToS/Traffic Class fields on their packets. If all scan traffic must carry the correct ToS or Traffic Class, tag it through settings on the managed switch port instead.

TCP ports

The **Included TCP ports** and **Excluded TCP ports** fields override the default scan ports. The string “defaults” looks up the current default port list at scan time. The current port list is:

1	7	9	13	17	19	21	22	23	25	37
42	43	49	53	69	70	79	80	81	82	83
84	85	88	102	105	109	110	111	113	119	123
135	137	139	143	161	179	222	264	280	384	389
402	407	442	443	444	445	465	500	502	512	513
515	523	524	540	541	548	554	587	617	623	631
636	664	689	705	717	743	771	783	789	830	873
888	902	903	910	912	921	949	990	993	995	998
1000	1024	1030	1035	1080	1083	1089	1090	1091	1098	1099
1100	1101	1102	1103	1128	1129	1153	1158	1199	1211	1220
1234	1241	1260	1270	1300	1311	1337	1352	1433	1434	1440
1443	1468	1494	1502	1514	1521	1530	1533	1581	1582	1583
1604	1610	1611	1723	1755	1801	1811	1830	1883	1900	1911
2000	2002	2021	2022	2023	2024	2031	2049	2068	2074	2082
2083	2100	2103	2105	2121	2181	2199	2207	2222	2224	2323
2362	2375	2376	2379	2380	2381	2404	2443	2455	2525	2533
2598	2601	2602	2604	2638	2809	2947	2967	3000	3001	3003
3033	3037	3050	3052	3057	3071	3080	3083	3128	3142	3200
3210	3217	3220	3260	3268	3269	3273	3299	3300	3306	3311
3312	3351	3389	3460	3478	3500	3502	3628	3632	3690	3780
3790	3817	3868	3871	3872	3900	4000	4092	4322	4343	4353
4365	4366	4368	4369	4406	4433	4443	4444	4445	4567	4659
4679	4730	4786	4840	4843	4848	4900	4949	4950	4987	5000
5001	5003	5006	5007	5022	5037	5038	5040	5044	5050	5051
5060	5061	5093	5094	5168	5222	5247	5250	5275	5347	5351
5353	5355	5392	5400	5405	5432	5433	5466	5498	5520	5521
5554	5555	5560	5580	5601	5631	5632	5666	5671	5672	5678
5683	5800	5814	5900	5901	5902	5903	5904	5905	5906	5907
5908	5909	5910	5911	5920	5938	5984	5985	5986	5988	5989
6000	6001	6002	6050	6060	6070	6080	6082	6101	6106	6112
6161	6262	6274	6277	6333	6379	6405	6432	6443	6481	6502
6503	6504	6514	6542	6556	6660	6661	6667	6905	6988	7000
7001	7002	7021	7070	7071	7077	7080	7100	7144	7181	7210
7373	7443	7444	7474	7510	7547	7579	7580	7676	7700	7770
7777	7778	7780	7787	7800	7801	7860	7879	7902	7990	7999
8000	8001	8002	8003	8004	8006	8008	8009	8010	8012	8014
8020	8023	8028	8030	8080	8081	8082	8083	8084	8085	8086
8087	8088	8089	8090	8095	8098	8099	8100	8123	8127	8160
8161	8172	8180	8181	8182	8188	8193	8200	8205	8222	8265
8300	8303	8333	8400	8428	8443	8444	8445	8446	8447	8448
8449	8471	8488	8500	8503	8530	8531	8545	8649	8686	8787
8800	8812	8833	8834	8850	8871	8880	8883	8888	8889	8890
8899	8900	8901	8902	8903	8983	9000	9001	9002	9042	9060
9080	9081	9084	9090	9091	9092	9099	9100	9111	9152	9160

9200	9300	9380	9390	9391	9401	9418	9428	9440	9443	9444	9471
9495	9524	9527	9530	9593	9594	9595	9600	9809	9855	9997	9998
9999	10000	10001	10008	10050	10051	10080	10098	10162	10202	10203	10250
10255	10257	10259	10443	10616	10628	10800	11000	11099	11211	11234	11333
11434	11443	12174	12203	12221	12345	12379	12397	12401	13364	13400	13500
13778	13838	14330	15200	15671	15672	16102	16443	16992	16993	17185	17200
17472	17775	17776	17777	17778	17781	17782	17783	17784	17790	17791	17798
18245	18264	18789	18881	19000	19300	19530	19810	19888	20000	20010	20031
20034	20101	20111	20171	20222	20256	20293	21047	22222	23472	23791	23943
24442	25000	25025	25565	25672	26000	26122	27000	27017	27018	27019	27080
27888	28017	28222	28784	29418	30000	31001	31099	32400	32764	32843	32844
32845	32913	33060	34205	34443	34962	34963	34964	37718	37777	37890	37891
37892	38008	38010	38080	38102	38292	40007	40317	41025	41080	41523	41524
44334	44343	44818	45230	46823	46824	47001	47002	47290	48898	48899	49152
50000	50013	50021	50051	50070	50090	50121	50505	51443	52302	52311	53282
54321	54921	54922	54923	55553	55580	57772	61614	61616	62078	62514	65002
65535											

Prescan modes for large IP spaces

Sometimes you don't know the extent of your IP space, which subnets are in use, or how many assets you have, and that can make discovery scans hard to tune for efficiency and speed. When the space is also large, like a /16 with a few thousand IPs in use, a full discovery scan can take longer because it checks more than 500 TCP ports and 15 UDP ports on every address. In these cases, you may want to tune your scan settings to prefilter ranges and IP addresses before a full scan.

runZero has two prescan modes for faster scans: subnet sampling and host ping.

Subnet sampling

To speed up scans of large subnets, turn on the **"Only scan subnets with active hosts"** advanced scan option. A prescan then runs against the target space to find the subnets with an active host, using heuristics runZero has collected about which addresses are more likely to respond across subnets. That lets runZero cover a large space quickly by identifying the subnets in use before it starts full probes. Every subnet found to have active hosts is then fully scanned, unless you enable host pings.

Subnet sampling has two parameters. The sample rate sets what percentage of addresses in each subnet the prescan checks to decide whether to scan the subnet. The subnet size sets how many IP addresses make up each subnet. By default, the subnet size is 256 addresses, a /24 subnet, and the prescan checks 3% of the addresses in each subnet.

Host ping

Once you know which subnets are in use, you may want to limit the full scan to addresses that answer the most common ping methods, such as ICMP and some TCP and UDP ports. With the **"Limit scans to pingable hosts"** advanced scan option, only hosts that respond to a ping request are fully scanned.

The runZero Explorer uses multiple protocols for ping scans:

- Conventional ICMP ping, which sends an ICMP echo request and looks for an ICMP echo reply.
- TCP ping, which sends a TCP SYN packet to a series of common ports and checks whether the host answers with RST or TCP SYN/ACK.
- UDP ping, which sends a packet to port 65535 and checks for an ICMP port unreachable response.

You can adjust the ports used for TCP and UDP ping in the LAYER2 section of the Probes and SNMP tab when you set up a scan task.

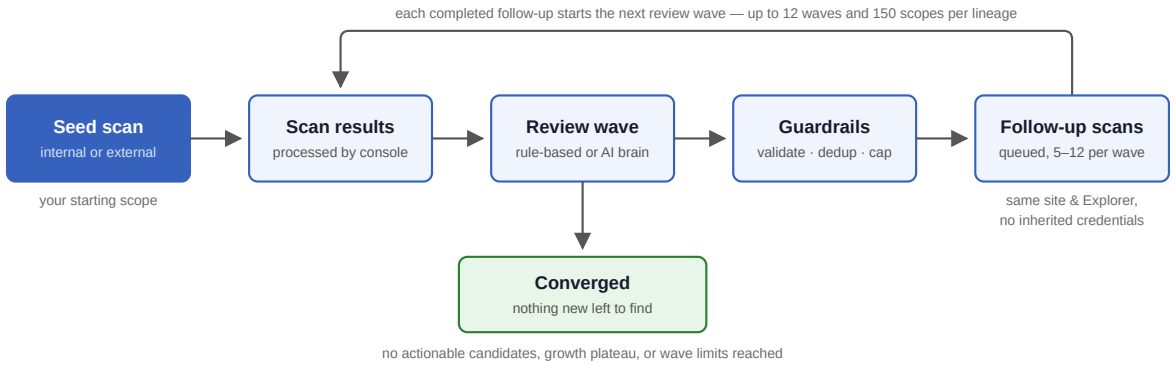
Enterprise firewalls often block ping, and hosts are often configured to ignore ping requests, so limiting scans to pingable hosts can miss assets entirely even though their IP addresses were probed. If your goal is faster scans, subnet sampling is usually the better option.

You can use subnet sampling and the pingable-host limit together, but we recommend it only as a last resort for reducing scan times.

Autonomous discovery

Autonomous discovery is a scan mode that reviews the results of every completed scan and automatically queues bounded follow-up scans to expand coverage. Rather than asking you to enumerate every subnet or subdomain up front, runZero starts from a small seed scope, reads the evidence each scan produces (routing tables, ARP caches, traceroute hops, TLS certificates, DNS answers), and decides what to scan next. The loop repeats until nothing new is left to find.

Autonomous discovery works out of the box on deterministic rules and needs no AI. Accounts with AI enabled can let their own AI provider drive the analysis instead; AI-assisted discovery describes both paths.



Requirements

- Your account needs the autonomous discovery entitlement, which comes with every runZero license tier, Community Edition included.
- Internal discovery needs at least one active Explorer that can reach the private networks you want to explore.
- External discovery targets public IP space and typically runs through a Hosted External Explorer.
- AI is not required. AI-assisted analysis runs only when the account has the AI entitlement and a working AI configuration.

How it works

1. You launch a scan with an autonomous discovery mode selected. This seed scan runs like any other scan.
2. After the console processes the results, a **review wave** analyzes the new evidence and proposes follow-up candidates: private subnets, related hostnames and domains, or deeper scans of confirmed-live networks.
3. runZero validates every candidate against the guardrails below, deduplicates it against everything the lineage has already scanned, and drops it if its confidence score is below 0.30.
4. Surviving candidates are queued as child scan tasks, capped at 5 to 12 follow-ups per wave depending on how confident the wave is (8 is typical). Adjacent `/24` candidates share a single task, and discovered hostnames land as one multi-target scan.
5. Each follow-up keeps the autonomous mode, so the loop repeats when it completes, up to 12 waves and 150 distinct scopes per lineage.

The loop **converges** (stops queueing new work) when a wave produces no actionable candidates, novelty drops off, asset growth or vendor diversity plateaus for consecutive waves, or the iteration and scope limits are reached. The task reports convergence as a status, not as an error.

Guardrails

- Follow-up scans never leave the lane they started in: internal discovery only proposes private address space, and external discovery only proposes names under your seed domains.
- Follow-ups always respect site exclusions, existing scope, and everything the lineage has already scanned.
- Overly wide targets are rejected: internal candidates are never broader than a `/16` (IPv4) or `/48` (IPv6).
- Follow-up scans do not inherit credentials from the seed scan. runZero strips credential IDs and sensitive scan options before it creates each child task.

Internal discovery

Internal mode grows coverage of private networks from topology evidence: SNMP routing tables, interface addresses and ARP caches, RIP routes, traceroute hops, CDP and LLDP management addresses, and default gateway attributes. It ignores public routes and default routes.

Internal discovery uses a depth progression rather than deep-scanning everything immediately:

- **discovery** runs a light probe of a newly inferred subnet to confirm live hosts.
- **sweep** runs a coarse liveness scan of a wider private block when clustered evidence justifies it.
- **deepen** runs a full default scan of a `/24` (or IPv6 `/48`) that discovery confirmed as live. A subnet is deepened only once, and a subnet that looks like a firewall answering on every address is never treated as live.

runZero also tracks coverage per scope and per dimension (liveness, ports, screenshots, and so on), so later waves can re-deepen scopes whose deep coverage has gone stale rather than only chasing new space.

External discovery

External mode starts from the public IPs and `domain:` targets in your seed scope and expands through names: TLS certificate subjects and SANs, forward DNS answers, parent domains, and observed hostname patterns such as region or environment series.

Expansion is strictly bounded by ownership. A candidate hostname or domain must share a registrable domain with your seed scope; an unrelated domain that appears in a shared certificate is dropped, and observed ASNs are never turned into IP sweeps. On the first wave, if public IPs are present, runZero may also queue a one-time passive Shodan import to enrich the seed assets before active expansion.

When scanning reveals infrastructure that cannot be enumerated over the network, such as cloud accounts or virtualization hosts, the review records an **integration hint** that suggests the relevant integration instead of queueing a scan.

Starting an autonomous scan

On the Scan page in the Data sources section of the navigation menu, the **External** and **Internal** buttons open the scan form pre-configured for that mode, with a recognizable task name and a recommended starting scope. You can also set up any standard scan by hand: in the scan form, set the **Autonomous discovery** control to **Off**, **Internal**, or **External**, and use the **Use recommended targets** button to fill the scope with the site's default scope (internal) or your organization's domain and public egress IP (external).

- Internal discovery works best with a **Continuous** schedule and a seed scope drawn from the Explorer's local networks. Internal scopes must contain private IP addresses

only, with no hostnames or public ranges.
- External discovery works best with a recurring schedule (the form defaults to every 5 minutes, since hosted scans do not support the Continuous frequency) and a seed scope of public IPs, ASNs, and `domain:` targets. External scopes must not contain private addresses.
- On Community Edition licenses, autonomous scans default to an hourly schedule.

Scan templates can set a default autonomous discovery mode, so recurring scans created from a template participate automatically.

Working with follow-up scans

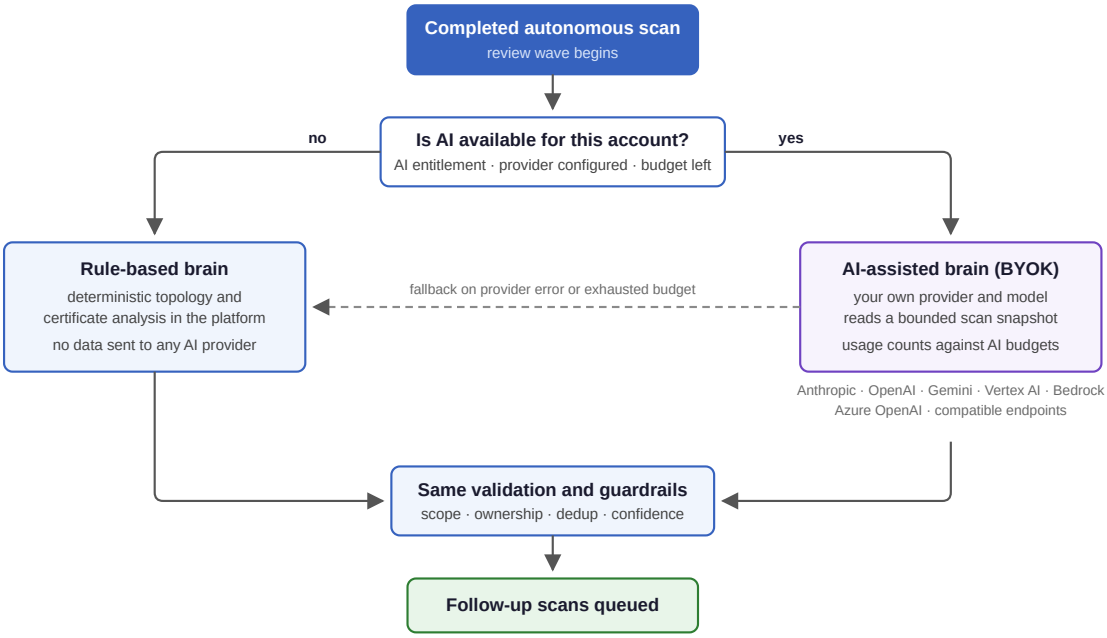
Queued follow-ups appear on the tasks page named `Autonomous follow-up: <target>`, tagged `autonomous=true`, with a description recording the wave number, confidence, and the evidence that motivated them. Each runs on the same site and Explorer (or hosted zone) as its parent.

Every autonomous task's detail page includes an **Autonomous discovery** card. It shows the loop status (mode, state, wave number, assets discovered, novelty, and vendor diversity) plus the latest review wave: which brain ran, every candidate it considered, its confidence, and the outcome (queued, skipped for low confidence, deferred by the wave cap, duplicate, or stop signal). When AI was used, the card also reports the wave's token usage.

To stop the loop, stop or dismiss any task in the lineage. This cancels the entire lineage: active siblings and descendants stop, and no further follow-ups are queued.

AI-assisted discovery

Autonomous discovery runs one of two analysis engines, and the task details card tells you which one reviewed each wave.



Without AI: the rule-based brain

By default, review waves use the **rule-based** brain: deterministic topology and certificate analysis that runs entirely inside the runZero platform. Nothing is sent to any AI provider, there is no per-wave cost, and results are fully reproducible. This is the complete, supported path: the rule-based brain implements all of the internal and external expansion logic described above.

With AI: bring your own key

runZero AI support is BYOK (bring your own key): you connect a model from your own AI provider account rather than runZero supplying one. When AI is available, review waves use the **AI-assisted** brain, and your configured model examines the same scan snapshot through a set of read-only tools and proposes candidates with reasoning. This can produce smarter expansion, such as recognizing naming conventions, prioritizing interesting segments, and stopping earlier when evidence is weak. To use it:

1. Your account must have the AI entitlement.
2. A superuser enables AI under Account settings > AI configuration by selecting or creating an AI provider credential. Supported providers include Anthropic, OpenAI, Google Gemini, Google Vertex AI, AWS Bedrock, Azure OpenAI, and OpenAI-compatible or Anthropic-compatible endpoints (OpenRouter, Ollama, internal proxies). **Verify & save** runs a live check against the provider before activating the configuration.
3. Each organization can override the account provider with its own credential, or opt out of AI entirely, from its AI settings.

The AI-assisted brain follows the same token budgets as other AI features. You can set daily input and output token caps per account and per organization, and each wave's usage is

recorded against the `autoscan` feature in the AI usage report on the AI configuration page, as well as on the task itself.

What to expect from AI-assisted waves:

- AI proposals are advisory. Every candidate passes exactly the same validation, scope, ownership, and deduplication checks as rule-based output before anything is queued.
- The AI-assisted brain sees only a compact summary of the completed scan's results and cannot browse your wider inventory. As with all BYOK features, your agreement with your provider governs the data sent to it.
- If AI is unavailable for any reason (missing entitlement, a deleted credential, an exhausted budget, or a provider error mid-wave), the wave falls back to the rule-based brain and the loop continues. Autonomous discovery never stalls waiting for AI.

Limits

- A lineage runs at most 12 review waves and 150 distinct scopes.
- Each wave queues at most 5 to 12 follow-up scans; the rest are deferred to the next wave.
- Candidates below a confidence score of 0.30 are recorded but never scanned.
- Internal candidates are capped at `/16` (IPv4) and `/48` (IPv6) widths; external candidates must stay under a seed registrable domain.

Troubleshooting

If autonomous discovery isn't behaving the way you expect, the questions and answers below may help.

Why didn't my autonomous scan queue any follow-up scans?

1. Check the **Autonomous discovery** card on the task details page. If the state is **Converged**, the review wave found no actionable evidence. That is the loop working as intended, not a failure.
2. Review the candidates table on the same card. The wave may have proposed candidates and then skipped them for low confidence, deduplicated them against earlier waves, or deferred them under the wave cap.
3. Verify the task was not stopped or dismissed while results were processing; a canceled lineage never queues follow-ups.

Why does the task show the rule-based brain when AI is enabled?

1. Confirm the account has the AI entitlement and that the AI configuration shows an active provider. A deleted credential disables AI-assisted waves rather than borrowing another provider.
2. Check whether the organization has opted out of AI or exhausted its daily token caps; budget-exhausted waves fall back to rules until the budget resets.
3. A provider error during a wave also falls back to rules for that wave; check the AI usage report for partial usage.

Why do follow-up scans run without my scan credentials?

This is by design. Inferred targets do not inherit trust from the seed scan, so runZero strips credentials and sensitive scan options from every follow-up. If you need deeper authenticated data, run a credentialed scan of the discovered scope manually.

Why did external discovery skip a hostname it found in a certificate?

External expansion follows only names whose registrable domain matches your seed scope, so a hostname on a shared certificate that belongs to another organization's domain is dropped on purpose. If you own that domain and want it explored, add it to a new seed scan's scope.

Initial network scans

Background

Once you have an Explorer installed, you can start network discovery. The end goal is scheduled scans you set and forget, but the first scans deserve a more structured approach.

Use your first scans to:

- Verify the Explorer is set up properly and has everything installed
- Validate Explorer connectivity to different parts of the network
- Learn how long scans of different sizes take, which helps with future scheduling

Your first few scans

Start small to confirm everything works as expected: a few /24 network blocks from each of the RFC 1918 ranges.

To set up the first scan:

1. Go to **Sites > New Site > Create** a new temporary site within the Organization
2. Go to **Tasks > Scan > Standard Scan** to create a scan task
3. Choose the new site you created in step 1
4. In the Discovery Scope, include a range of RFC1918 IP addresses plus a small network or two that you know is in use. A suggested RFC1918 range:

   ```
   10.0.0.0/24,10.0.255.0/24,10.64.0.0/24,10.64.255.0/24,10.128.0.0/24,10.128.255.0/24,
   10.192.0.0/24,10.192.255.0/24,10.255.0.0/24,10.255.255.0/24,192.168.0.0/24,192.168.64.0/24,
   192.168.128.0/24,192.168.192.0/24,192.168.255.0/24,172.16.0.0/24,172.23.0.0/24,172.31.0.0/24,
   <your networks here>
   ```

5. On the Advanced tab, enable the *Subnet sampling* option
6. Click Initialize Scan

When these scans complete, check the following:

- Check the `ipv4.traceroute` value for assets in each RFC1918 range to verify you aren't sending traffic to an edge router or firewall.
  - Unused private IPs should have stubbed-out routes so traffic doesn't go to the default gateway, which can create a loop. Traceroutes from the Explorer can also verify this.
- If your scan results show a long run of roughly sequential IPs with only ICMP or a few of the same ports open, that's probably a proxy or firewall. Check those IPs to see if any are real. To find assets with only ICMP enabled, use the inventory query `alive:t AND service_count:=1 AND service_count_icmp:=1`
  - You can add an allow rule for the Explorer IP to scan the devices on the other side properly
  - Or add a second Explorer on the other side of the proxy or firewall
- If you get reports or alerts about service outages, look for session-aware devices such as routers, firewalls, and proxies struggling with the session load. If you run into this, you have a couple of options.
  - The simplest fix is another Explorer on the other side of the device, running its own scans
  - Or segment your scans on the existing Explorer: run smaller, separate scan tasks for the network ranges on the other side of the device, with lower packets per second and *max group* sizes so fewer IPs are scanned at once
- Check how long each scan took to estimate how long larger scans would take
- Verify you see screenshots on ports that accept HTTP/HTTPS requests
  - If you don't see any, you likely need to install Chrome on the machine
- Check for MAC addresses
  - If you aren't seeing them, configure SNMP
  - If SNMP is configured, verify the community strings and check for unmanaged switches



Once your first scan has run successfully, you can delete the temporary site and set up a real scan.

Full RFC 1918 scans

After the initial test scans, expand scanning to cover every subnet with live assets. One way to find all of them is a full RFC 1918 scan.

The **Full RFC 1918 discovery** scan option discovers assets across these private address ranges in a single task:

- 10.0.0.0/8 or 10.0.0.0-10.255.255.255
- 172.16.0.0/12 or 172.16.0.0-172.31.255.255
- 192.168.0.0/16 or 192.168.0.0-192.168.255.255

> ⚠
> The **Full RFC 1918 discovery** scan option is recommended only for small networks with limited complexity, and only in a single site configuration.

In a large, complex network, a single scan of the entire RFC 1918 private address space can take days, if not weeks. For more on scanning the full RFC 1918 space, runZero recommends the Achieving RFC 1918 coverage playbook.

Once you've scanned your private address space, review Identifying gaps in scanning for the built-in reports that help you understand your network coverage.

Identifying gaps in scanning

Background

After a full network discovery scan, you can start to understand your coverage and optimize it. This guide covers the built-in reports in runZero that show your gaps in network coverage.

RFC 1918 coverage

Start with the **RFC 1918 coverage** report. It shows which internal IPv4 subnets you have scanned, which likely contain assets, and which are still unknown.

The scan coverage maps show every address scanned within the `10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16` ranges, and the legend gives the percentage of each address space you have scanned. Click into any scanned subnet to open the subnet grid for deeper asset analysis.

Identify scanned and un-scanned areas with the coverage map: Red outlines mark un-scanned addresses that runZero knows about indirectly but hasn't scanned directly, for example when it finds a secondary IP address on a multi-homed device within a scanned subnet. The red boxes show the subnets most likely to be in use but un-scanned.

Scan missing subnets: From the coverage report, you can launch a scan for any missing subnets in a given RFC1918 block. Look for the binocular icon.

Scan missed subnets: The missing subnets appear in the scan scope with subnet ping enabled by default. Tune the scan configuration as needed for your environment.

Subnet utilization

The **Subnet utilization** report shows a view of your network similar to the RFC 1918 coverage report, with the emphasis on subnets that contain live assets. It lists each subnet in your site definitions with a count of live assets for that site and subnet and a utilization percentage. Live assets outside any site subnet are grouped into an inferred subnet based on the network mask size you select.

From this report, you can pivot to the asset inventory for a subnet or start a new scan of it. You can also export the results as a CSV, which helps with more complex data analysis and with scheduling recurring scans.

Network Map

The Network Map is the fastest way to spot scanning gaps. It overlays Layer-2 switch topology, Layer-3 routing, traceroute paths, and bridges in a single interactive view, and renders **unmapped MACs** and **unscanned traceroute hops** by default, so every gap shows up as a node without an asset behind it. Search and filter directly inside the map (`multi_homed:t`, `category:OT`, etc.), then click any unmapped node to see which Explorer or scan reached it and which subnet it sits in.

Switch topology

The Switch Topology report shows layer-2 link information extracted from SNMP-enabled switches. Use it to find unmapped assets and investigate why they aren't showing up in your scans.

Configuration for this report: The Switch Topology report maps switch ports to assets from data enumerated over SNMP. Where SNMP v1 or v2 with the default public or private communities is in use, this enumeration happens automatically. You can also provide non-default communities for SNMP v1/v2/v3 in the scan configuration. Click a node in the report to expand it and show its connections.

Finding unmapped MACs: This topology view helps you see how a given asset or switch is connected, and it also gives you a data point about risk: the number of unmapped assets. An unmapped asset is a MAC address connected to a switch but not found in an ARP cache or through any of runZero's other techniques for remote MAC address discovery.

Re-scan to properly map MAC addresses: Where a runZero agent is connected to each network segment, unmapped MACs may point to VLANs or network segments missing from the scan scope. Where runZero scans assets multiple hops away, the unmapped asset count can estimate how well the remote segment is being identified.

Unmapped MACs

For a condensed view of all unmapped MACs, use the Unmapped MACs report. It lists every unmapped MAC in your organization, organized by switch and then by port, and you can use this data the same way as the unmapped MACs data from the Switch Topology report.

Managing scan templates

A scan template is a predefined set of scan options and settings. If you use a scan configuration often, save it as a scan template, and the next time you create a scan, choose the template instead of configuring the settings by hand. Every update you make to the template applies automatically to new and recurring scans based on it, and to any queued

scans that were set up from it but haven't started yet. Templates save you time and make a misconfigured scan less likely.

When you create a scan from a template, you can't edit the fields the template sets. You can still configure the site, scan name, discovery scope, Explorer, and scan schedule, because the template doesn't define those.

Each time a scan runs with values from a template, runZero saves the scan task with a copy of those parameters, so the task lists the values it used even if the template changes after the scan completes.

Creating a scan template

You can create a scan template in a few ways:

- From **Tasks > Task library**
- From an existing scan task
- From an existing scan template

Creating a scan template from scratch

1. Go to **Tasks > Task library** to open the task library.
2. From the **Task library**, click **Add template**.
3. In the scan configuration form, enter a name for the template. You'll use this name to find the template and assign it to a new scan.
4. Configure the scan as you normally would, using these tabs:
    - **Standard** holds the name, description, and scan rate for the template.
    - **Advanced** covers excluded hosts, included ports, tags, host rates, group size, UDP probe max attempts, max TTL for all scan probe packets, ToS, screenshots, and subnet sampling.
    - **Probes and SNMP** enables SNMP and additional probes and takes their credentials. With the SNMP probe enabled, the scan uses the port and credentials you provide to find anything with SNMP enabled.
    - **Credentials** enables any preconfigured credentials you want the scan to use; see Scanning with credentials. The scan uses a credential when its defined CIDR scope matches the scan target.
5. Save the template.

To use the template, open the *Templates* tab when you configure a scan or create a Template scan from the scan menu on the inventory or tasks pages.

Creating a scan template from a scan task

1. Go to **Tasks** to open the Tasks overview page.
2. Find the scan task you want to base the template on and click its name to view the task details.
3. From the task details, click the **Copy** dropdown menu and choose **Copy for new template**.
4. In the scan configuration form, enter a name for the template. You'll use this name to find the template and assign it to a new scan.

5. Configure the scan as you normally would, using these tabs:
    - **Standard** holds the name, description, and scan rate for the template.
    - **Advanced** covers excluded hosts, included ports, tags, host rates, group size, UDP probe max attempts, max TTL for all scan probe packets, ToS, screenshots, and subnet sampling.
    - **Probes and SNMP** enables SNMP and additional probes and takes their credentials. With the SNMP probe enabled, the scan uses the port and credentials you provide to find anything with SNMP enabled.
    - **Credentials** enables any preconfigured credentials you want the scan to use; see Scanning with credentials. The scan uses a credential when its defined CIDR scope matches the scan target.
6. Save the template.

To use the template, open the *Templates* tab when you configure a scan or create a Template scan from the scan menu on the inventory or tasks pages.

Creating a scan template based on a template

1. Go to **Tasks > Task library** to open the task library.
2. From the **Task library**, find the template you want to base the new one on and click its name to open it.
3. On the scan template configuration page, click **Copy**. runZero duplicates it as a new scan template.
4. In the scan configuration form, enter a name for the template. You'll use this name to find the template and assign it to a new scan.
5. Configure the scan as you normally would, using these tabs:
    - **Standard** holds the name, description, and scan rate for the template.
    - **Advanced** covers excluded hosts, included ports, tags, host rates, group size, UDP probe max attempts, max TTL for all scan probe packets, ToS, screenshots, and subnet sampling.
    - **Probes and SNMP** enables SNMP and additional probes and takes their credentials. With the SNMP probe enabled, the scan uses the port and credentials you provide to find anything with SNMP enabled.
    - **Credentials** enables any preconfigured credentials you want the scan to use; see Scanning with credentials. The scan uses a credential when its defined CIDR scope matches the scan target.
6. Save the template.

To use the template, open the *Templates* tab when you configure a scan or create a Template scan from the scan menu on the inventory or tasks pages.

Applying a scan template to a scan

When you configure a scan, the **Templates** tab lists every template available for it. Pick one instead of configuring the scan by hand.

You can also open the **Task library** page and choose **Create a scan based on this template** from the template's actions.

Editing a scan template

Every update to a scan template affects every scan that uses it. After you change a template, the next scan picks up the new settings automatically, and so does any scan queued to run.

Editing a scan that uses a scan template

To change the settings of a scan that uses a template, edit the template; the change reaches every scan that uses it. Otherwise, delete the scan and create a new scan configuration with your changes.

When you create a scan from a template, you can't edit the fields the template sets. You can still configure the standard fields, such as the site, scan name, discovery scope, Explorer, and scan schedule, because the template doesn't define those.

Deleting a scan template

Deleting a template removes it from runZero permanently, along with all of its configuration. To delete one, go to the Task library, find the template, and click **Delete**. A dialog asks you to confirm, because the deletion can't be undone.

Scanning with credentials

The Credentials page is a single place to store any secure credentials runZero needs, including:

- SNMPv3 credentials
- Access secrets for cloud services like AWS and Azure
- API keys for services such as Censys and Miradore

runZero stores credentials encrypted in its database. Credentials that runZero Explorers use, such as SNMP passwords, are also encrypted in transit to the Explorers. For security reasons, the secret part of any credential can't be viewed once entered.

When you add a credential, you can make it a global credential available to all organizations, or allow access only to specific organizations. The **Allow all** and **Disallow all** buttons apply the same setting across every organization at once, and you can also toggle access for each organization on its own.

You can edit most credential fields after saving. Some fields, like URLs, can't be edited after saving for security reasons. Sensitive fields, such as passwords or access keys, are hidden but can be overwritten.

Credential settings

The fields and options for a credential depend on its type.

VMware and SNMP credentials, which the runZero Explorer uses, accept a CIDR allow list that limits which scanned IP addresses the credential is used with. That keeps your SNMP or VMware credentials from going to every scanned host on the network and limits them to specific IP addresses or ranges.

Credential verification

You can verify a credential when you create or edit it to confirm it authenticates. Choose **Verify & save** to run the verification before saving. If verification fails, runZero shows an error message and offers **Save anyway**, which saves the most recent verification status.

Credential management

Managing credentials takes administrator-level permissions. Users with **Administrator** as their default role can fully manage all credentials. Users with per-organization permissions can't access global credentials and can manage credentials only in organizations where they hold administrator permissions. An organization administrator can't delete a shared credential.

Scanning with SNMP

SNMP is an open standard network protocol for collecting information about devices on a network. runZero scans with all three main versions of the protocol. With the right community strings or SNMP v3 credentials, a scan collects details such as serial numbers, device models, ARP caches, and switch port tables. If an asset is missing SNMP data, skip ahead to Checking SNMP v3 credentials and SNMP v1/v2 authentication.

SNMP versions 1 and 2

SNMP v1 and SNMP v2 both send community strings across the network in plain text. Neither version encrypts its data packets, and because SNMP runs over UDP, there's no way to add TLS for a secure connection.

SNMP version 1 was designed in the 1980s as an interim protocol, meant to be replaced by ISO CMIP. It was built to run on any network common at the time, not only TCP/IP, so security was left to the host network. The protocol defined a community string for grouping assets in any way you liked, but didn't specify how access should be granted.

SNMP version 2 tried to introduce a security model based on "parties," but it wasn't widely adopted, so a revised standard was issued as SNMP v2c. SNMP v2c dropped party-based security and went back to plain community strings. The original SNMP v2 specification is considered obsolete.

SNMP version 3

SNMP v3 fixes the security problems of SNMP v2 by building password-based authentication and encryption (called privacy protection) into the standard. The catch is that the standard allows several algorithms for both authentication and encryption, and you have to know which ones your network devices use. Unlike SSH, for example, the protocol has no negotiation of encryption methods.

SNMP v3 devices operate in one of three modes:

- noAuthNoPriv: no authentication and no privacy protection (no encryption).
- authNoPriv: authentication is required, but there's still no privacy protection.
- authPriv: both authentication and privacy protection are required.

runZero's SNMP support

runZero scans with SNMP v1, SNMP v2 (the common `v2c` variant), and SNMP v3. A scan can use only v1/v2, only v3, or both.

SNMP scanning is on by default. You can turn it off or customize it on the SNMP tab when you set up a scan or a scan template.

SNMP v1 & v2 scanning

You can set up community strings for SNMP v1/v2 scanning in two ways. The first is to enter them as a comma-separated list on the SNMP tab. By default, runZero supplies the community strings `public` and `private`, which are common defaults on network-enabled hardware such as printers and NAS servers. If you remove those defaults, runZero does not probe with them.

The second is to store them in the credentials feature as the credential type *SNMP v2 Communities*. Communities technically shouldn't be used as credentials, because they cross the network in plain text, but in practice many networks use them that way. Storing your community strings as credentials lets you use the CIDR allow list feature to control which parts of the network receive them, which reduces the risk of a rogue device capturing them.

If you remove all community strings from the SNMP tab and store no SNMP v2 Communities as credentials, runZero performs no SNMP v1/v2 scanning.

SNMP v3 scanning

Most devices only give up potentially sensitive information, such as serial numbers and software versions, in response to an *authenticated* SNMP v3 query. To perform authenticated SNMP v3 scanning, set up an *SNMP v3 Credential* record in runZero's credentials feature.

The authentication protocol sets the hashing algorithm that processes the authentication passphrase: in other words, how the runZero Explorer logs in to the remote device.

The most common default authentication algorithm on devices is `sha` (HMAC-SHA-96), which the SNMP v3 standards require. Newer devices may support more secure variants such as SHA-256, and runZero supports up to SHA-512. runZero also supports HMAC-MD5-96, as the standards require, but avoid it on your network because the MD5 algorithm is known to be insecure.

The privacy protocol determines how data to and from the remote device is encrypted against eavesdropping. The privacy passphrase seeds the encryption.

The most common privacy algorithm, required by the original SNMP v3 standards, is CBC-DES-128, selected as `des` in the runZero user interface. Later RFCs added AES-128, represented as `aes`. runZero also supports more secure variants that newer devices may use, such as AES256.

As described above, devices may require only authentication, both authentication and privacy, or neither. Some device configurations have room for only a single passphrase. This usually means the device uses the same passphrase for both privacy and authentication when it runs in authPriv mode.

As with SNMP v2 community strings, you can set a CIDR allow list to control which IP addresses receive SNMP v3 credentials. This is especially recommended if your network doesn't require privacy passphrases.

Beyond credentialed scanning, runZero also gathers information from the SNMP v3 pre-authentication handshake.

To perform SNMP v3 scanning across Cisco switches, modify the ACLs to include the following rule:

- Newer IOS versions: `snmp-server group YourGroupName v3 auth context vlan- match prefix`
- Older IOS versions: `snmp-server group YourGroupName v3 auth context vlan-1` (repeated for every VLAN)

Choosing which versions of SNMP are scanned

To scan with SNMP v3 only, remove the community strings from the SNMP tab when you set up a scan or template, and disable any SNMP v2 community strings stored as credentials on the Credentials tab.

To turn off authenticated SNMP v3 scanning and scan with SNMP v1/v2 only, disable any SNMP v3 credentials on the Credentials tab and specify community strings, either on the SNMP tab or in a credential.

To turn off all SNMP scanning, including uncredentialed SNMP v3, switch off the toggle on the SNMP tab labeled *Use the SNMP protocol for discovery*.

How scans are performed

runZero probes each device with every configured SNMP version and every community string, because devices often return different levels of data for SNMP v3 than for SNMP v2, and may return different information for different community strings.

SNMP runs over UDP, which doesn't guarantee packet order, so there's no way to control the order in which a remote device receives the community strings.

SNMP v3 authentication errors don't count as scan failures, because networks commonly use different sets of SNMP v3 credentials on different segments, and misconfigured devices with incorrect passphrases are common too.

Checking SNMP v3 credentials and SNMP v1/v2 authentication

If an asset you expected to have SNMP data has none, a common cause is a network firewall, which often blocks SNMP traffic entirely. If you can, deploy an Explorer on the far side of the firewall, on the network segment you want to scan with SNMP.

The second most common cause is incorrect SNMP v3 credentials. The asset's attributes tell you what happened:

- If runZero attempted SNMP v3 authentication, the asset records an *SNMP engine ID* as `snmp.engineID.raw`. The engine ID is usually a long string of hex digits and is used to calculate access keys.
- If SNMP v3 authentication failed, the attribute `snmp.failedAuth` reports the error. A request timeout likely means authentication succeeded but the encrypted data channel could not be set up, so you may have the wrong privacy password or privacy algorithm.
- If SNMP authentication succeeded (v1/v2 or v3), the `snmp.credentials` attribute holds the UUID of each credentials record that succeeded.
- If SNMP v1/v2 authentication with a community string succeeded, the asset has `snmp.secretCommunities` or `snmp.defaultCommunities` attributes, depending on the type of community string.
- If SNMP v1/v2 authentication succeeded with community strings entered directly into the task request rather than stored as credentials, the `snmp.secretCommunities` and/or `snmp.defaultCommunities` attributes are still present, but `snmp.credentials` is a zero UUID.

Debugging credential issues

Trying to debug SNMP v3 credential problems by trial and error, one scan at a time, is frustrating. Use the `snmpwalk` utility instead. It's part of the net-snmp open source project, packaged for most Linux distributions and available for macOS through Homebrew. Windows has unofficial builds, or you can install it in the Windows Subsystem for Linux (WSL).

`snmpwalk` connects to an IP address and dumps all the information it can retrieve. This example connects in `authPriv` mode with both authentication and privacy passphrases:

```
snmpwalk -v3 -l authPriv -a SHA -A "authentication passphrase"
  -x AES -X "privacy passphrase" -u username 10.0.1.25
```

(The text above is wrapped onto two lines, but it's one command.) The parameters are:

- `-v3`: switch to SNMP v3 mode.
- `-l authPriv`: run in authPriv mode, with both authentication and privacy required.
- `-a SHA`: authentication algorithm is SHA (SHA-128).
- `-A`: specify the authentication passphrase as the next argument.

- `-x AES`: privacy (encryption) algorithm is AES (AES-128).
- `-X`: specify the privacy passphrase as the next argument.
- `-u username`: specify the login username.
- `10.0.1.25`: the IP address to probe.

With the right passphrases and algorithms, you get data back, probably a lot of it. Otherwise you get an error message that may point to the cause.

To check SNMP v2 community strings, use a command like this:

```
snmpwalk -v2c -c commstring 10.0.1.25
```

The `-v2c` argument selects SNMP v2 (SNMP v2c) mode, and `-c` specifies the community string. With the wrong community string, `snmpwalk` runs for a while and then prints a timeout error.

Reviewing discovered SNMP services

When a scan finds SNMP services, runZero tracks their protocol versions at the asset level: you see `snmp` along with `snmp1`, `snmp2`, and `snmp3`, depending on which versions responded. That keeps asset inventory queries such as `protocol:snmp2` fast.

SNMP services also record how they authenticated and which protocols they used. If a common SNMP v2 community value worked (such as `public`, `private`, some vendor defaults, and common values like `password`, `cisco`, and `community`), the service details list the communities that responded in cleartext under `snmp.defaultCommunities`, and the asset details list them under `snmp.v2DefaultCommunities`. If the SNMP v2 community is not a common value, the service details report `snmp.secretCommunities` with a value of `true`, and the asset details carry an attribute called `snmp.v2SecretCommunities` with a value of `true`. If SNMP v3 was used, runZero populates `snmp.v3Usernames` at the asset level. Finally, an asset attribute called `snmp.auth` records whether `v2DefaultCommunity`, `v2SecretCommunity`, or `v3Username` authenticated to the asset.

Improving OS fingerprinting using Net-SNMP's `extend` feature

On some Linux distributions, the data Net-SNMP provides isn't enough to fingerprint the OS correctly. Distributions based on Red Hat Enterprise Linux, such as AlmaLinux OS, CentOS, Oracle Linux, and Rocky Linux, all return exactly the same values for a given point release. Net-SNMP's `extend` mechanism, which runs a defined command and returns its output over SNMP, can improve the result. Each target asset must run Net-SNMP as its SNMP software, and you must modify its SNMP configuration:

1. Modify `/etc/snmp/snmpd.conf` and
    1. Add the line:

        ```
        extend os-release /bin/cat /etc/os-release
        ```

    2. Grant the SNMP community string or user that runZero uses the right to collect the data by adding a line like:

        ```
        view systemonly included .1.3.6.1.4.1.8072.1.3.2
        ```

2. Restart the SNMP service:

```
sudo systemctl restart snmpd
```

3. Verify that the community string or account the runZero task will use can access the OID.

```
snmpwalk -v2c -c commstring 10.0.1.25 .1.3.6.1.4.1.8072.1.3.2.3.1.2.10.111.115.45.114.101.108.101.97.115.101
```

Object identifiers (OIDs)

This is the complete list of OIDs runZero may query on an asset. Some apply only to specific vendors.

System attributes

- `1.3.6.1.2.1.1.*.0`

IP addresses

- `.1.3.6.1.2.1.4.20.1.3.*`

MAC addresses

- `.1.3.6.1.2.1.2.2.1.6.*`

ARP caches

- `.1.3.6.1.2.1.4.35.1.4.*`
- `.1.3.6.1.2.1.3.1.1.2.*`
- `.1.3.6.1.2.1.4.22.1.2.*`

Routes

- `.1.3.6.1.2.1.4.24.7.1.7.*`

Port (CAM) tables

- `.1.3.6.1.2.1.17.7.1.2.2.1.2.*`
- `.1.3.6.1.2.1.17.4.3.1.2.*`

VLANs

- `.1.3.6.1.2.1.17.7.1.4.3.1.1.*`
- `.1.3.6.1.4.1.9.9.46.1.3.1.1.4.1.*`

Interface names

- `.1.3.6.1.2.1.17.1.4.1.2.*`
- `.1.3.6.1.2.1.31.1.1.1.1.*`

Serial numbers

- `.1.3.6.1.2.1.47.1.1.1.1.11.*`

Device models

- `.1.3.6.1.2.1.47.1.1.1.1.13.*`

Hostnames

- `.1.3.6.1.4.1.9.2.1.3.*`

Model names from Juniper switches

- `.1.3.6.1.4.1.2636.3.1.2.0.*`

Serial numbers from A10 devices

- `.1.3.6.1.4.1.22610.2.4.1.6.2`

OS release information from Net-SNMP's extend functionality (see above)

- `.1.3.6.1.4.1.8072.1.3.2.3.1.2.10.111.115.45.114.101.108.101.97.115.101`



For Cisco Catalyst & Nexus switches, the Explorer requires per-VLAN access to the ARP/FDB tables, through either community indexing (v2) or contexts (v3). runZero automatically enumerates the ARP cache and port table for each discovered VLAN. Without this access, the switch may be missing its snmp.macs.ports and snmp.macs.vlans attributes and may not appear on the Switch Topology report.

runZero gathers information from SNMP v3 systems even when no credentials are available. This information includes the engineID and the values of the following OIDs:

- `.1.3.6.1.6.3.15.1.1.1.0`: `usmStatsUnsupportedSecLevels`
- `.1.3.6.1.6.3.15.1.1.2.0`: `usmStatsNotInTimeWindows`
- `.1.3.6.1.6.3.15.1.1.3.0`: `usmStatsUnknownUserNames`
- `.1.3.6.1.6.3.15.1.1.4.0`: `usmStatsUnknownEngineIDs`
- `.1.3.6.1.6.3.15.1.1.5.0`: `usmStatsWrongDigests`
- `.1.3.6.1.6.3.15.1.1.6.0`: `usmStatsDecryptionErrors`

Using custom fingerprints

If you run a self-hosted instance or the standalone scanner, you can write your own fingerprints. Custom fingerprints add coverage for one-of-a-kind or custom assets and services, such as device prototypes or proprietary applications and services. They can also override similar built-in runZero fingerprints when you give them a same-or-higher certainty value. Understanding fingerprints explains how runZero builds its own.



To use custom fingerprints with the runZero standalone scanner, set `RUNZERO_EXTERNAL_FINGERPRINTS` as an environment variable when you launch the scanner.

Create new fingerprints

Custom fingerprints follow the structure and format of the open-source Recog fingerprint database. Write your fingerprint XML entries in files named and formatted like those in Recog. When an asset or service matches both a built-in runZero fingerprint and a custom fingerprint of the same kind, the fingerprint with the higher "certainty" value(s) wins (for example `hw.certainty`, `os.certainty`, `service.certainty`). If the certainty values tie, the custom fingerprint wins.

Add new fingerprints to your self-hosted runZero instance

Custom fingerprints must live inside the runZero installation directory (/opt/runzero by default) so the self-hosted instance can read them. To set them up:

- Create a new directory within the runZero install directory (for example `mkdir /opt/runzero/myfingerprints`)
- Update your `/etc/runzero/config` file with the new directory location (see below)
- Copy your custom fingerprint XML files into the new directory (for example `cp *.xml /opt/runzero/myfingerprints`)

To point `/etc/runzero/config` at your custom fingerprints directory, add this line to the end of the file with your favorite editor, replacing `myfingerprints` with the name of the directory you created:

```
RUNZERO_EXTERNAL_FINGERPRINTS=myfingerprints
```

You only need to create the directory and update `/etc/runzero/config` once. After that, add, remove, or modify custom fingerprints in the directory as needed, then restart the runZero application to reload them.

Restart the runZero service

Restart the self-hosted instance whenever you add, remove, or update custom fingerprints. Watch the runZero log file during the restart to confirm that the fingerprints loaded or to catch an error. A successful load logs a message like this:

```
@cee:{"level":"info","msg":"loaded (5) external fingerprints","time":"2022-09-12T19:51:49Z"}
```

If runZero hits a problem loading or processing the custom fingerprints, or reaches a 10 second timeout, it logs a warning and keeps running without any custom fingerprints.

Verify your fingerprints

After you add the custom fingerprints and restart the instance, check that they behave as expected in one of two ways:

- Run a scan task against a relevant asset or service, or
- Import an existing scan data file for the relevant asset or service

When the scan task or the import finishes, open the asset or service in the inventory and confirm that your custom fingerprints took effect.

Passive sampling

runZero Explorers can identify assets, services, and vulnerabilities by passively monitoring network traffic. Passive sampling can be an alternative data source where active scanning is prohibited, and it can help you find unknown IP ranges. A passive sampling task is limited to a single CPU core and temporarily skips packets when it reaches that limit. If an active scan runs on an Explorer with a passive sampling task, the passive task is interrupted and restarts once the active scan completes.

Configuring passive sampling

Configure passive sampling from the Explorer details screen: choose *Deploy* from the left navigator and click an Explorer. The network interfaces available for passive sampling appear on the left. Once enabled, the sampling task processes any traffic received on these interfaces, including plain layer-2 frames from SPAN and TAP ports, traffic encapsulated with 802.1q (VLAN or QinQ), and layer-3 traffic encapsulated with GRE or VXLAN. A full SPAN capture gives the best results, but any traffic, including broadcast, adds useful enrichment.

The passive sampling task automatically enables promiscuous mode on the selected interfaces. Some virtual infrastructure, such as VMware ESXi, may require promiscuous mode on the virtual switch before it forwards traffic to the interface.

The *Discovery scope* sets which IP ranges qualify for asset, service, and vulnerability creation. runZero ignores traffic with no source or destination inside an allowed range. You can also list *Excluded hosts* for IP ranges you want to skip.

The *Site* option sets where assets are created and updated. If the same IPs exist in other sites, you can end up with duplicate asset records.

You can set *Asset tags* for systems identified through passive traffic sampling. Tags can help you review the new assets and learn more about them with active scans.

Protocol support

aaa **4**ai **5**auth **4**automotive **2**backplane **20**backup **1**building-automation **5**camera **1**clear **56**database **18**deep **24**directory **2**discovery **57**email **2**encrypted **6**file **9**gaming **4**integration **1**iot **36**legacy **10**light **79**media **2**messaging **11**mgmt **70**mobile-core **12**monitoring **8**multicast **10**naming **4**network **1**ot **56**passive **14**printing **9**proxy **2**remote-access **12**routing **2**security **6**serial **1**siem **1**storage **8**time **2**tls **44**voip **13**vpn **7**web **22**

Filter by protocol, port, transpo

TCP **227** UDP **105** TLS **1** SCTP **8**

317 of 317 protocols

`acop` — Atlas Copco Open Protocol

TCP

Atlas Copco Open Protocol is a tightening-controller protocol used by Power Focus and Power MACS controllers on manufacturing assembly lines. Negotiates the protocol revision and queries controller identity, returning the supplier code, controller name, controller serial and software version, tool software version, and cell and channel identifiers.

Port: 4545

ot

`acpp`

TCP

`activemq` — Apache ActiveMQ OpenWire

TCP

Apache ActiveMQ OpenWire is the native binary wire protocol used by JMS clients to talk to ActiveMQ message brokers. Performs an OpenWire handshake and returns the broker version, host JVM and OS, and negotiated wire-format options.

Ports: 8161, 61616, 61617

messaging

`adb` — Android Debug Bridge

TCP

Android Debug Bridge is a developer protocol used to debug, install, and control Android devices over the network (typically on rooted phones, TVs, and IoT devices). Probes for an open ADB endpoint and returns the device's access state and ADB banner.

Ports: 5037, 5555

mgmtremote-access

`ads` — Beckhoff ADS/AMS

TCPUDP

Beckhoff ADS/AMS is the Automation Device Specification / Automation Message Specification protocol used to communicate with TwinCAT runtimes on Beckhoff PLCs and industrial PCs. Issues a ReadDeviceInfo request and returns the device name, AMS NetID, and TwinCAT runtime version.

Port: 48898

ot

`advantech-adam` — Advantech ADAM/EKI Config

TCPUDP

Advantech ADAM/EKI is the proprietary device-configuration service (5058/tcp and 5058/udp) served by Advantech EKI serial device servers and ADAM I/O modules. runZero sends the read-only 0x4102 network-info request (magic 4D 41 44 41, broadcast target,

empty token) and parses the reply for the device MAC, model, firmware version, and IP configuration. The read opcode is unauthenticated and has no side effects.
Port: 5058
discoverylightot
`airplay` — Apple AirPlay
TCP
AirPlay is Apple's wireless streaming protocol used by iOS, macOS, and Apple TV devices to mirror displays and stream audio/video. Detected from the _airplay._tcp and _raop._tcp mDNS records and HTTP banners advertised by AirPlay receivers.
discoveryiot
`ajp` — Apache JServ Protocol (AJP)
TCP
Apache JServ Protocol (AJP) is a binary protocol used by reverse-proxy front-ends (Apache httpd mod_proxy_ajp, nginx, IIS) to bridge HTTP requests to a back-end Tomcat or JBoss application server, typically on TCP/8009. runZero attributes services as AJP from external integration data (Shodan) and from banner / port hints; no active AJP probe is sent.
clearweb
`amqp`
TCP
AMQP is the Advanced Message Queuing Protocol used by RabbitMQ and other brokers for message-oriented middleware. Exchanges the protocol-header preamble and returns the negotiated AMQP version, broker product and version, runtime platform, cluster name, advertised SASL mechanisms, and TLS requirement.
Ports: 5671, 5672
messagingtls
`ard` — Apple Remote Desktop
UDP
Apple Remote Desktop is a discovery protocol used by macOS systems to advertise themselves to Apple Remote Desktop administrators on UDP/3283. runZero sends the ARD discovery request, validates the response type, and returns the advertised hostname and Apple machine model from the reply.
Port: 3283
discoverylightremote-access
`arp` — Address Resolution Protocol
TCP
Address Resolution Protocol is a Layer-2 protocol used to resolve IPv4 addresses to Ethernet MAC addresses on the local broadcast segment. runZero sends ARP requests for each target on the local network to discover live hosts, captures the MAC address from each reply for asset attribution and OUI-based vendor identification, and passively records ARP traffic observed during the scan.
discoverymulticastpassive
`atg` — Automatic Tank Gauge
TCP
Automatic Tank Gauge is a Veeder-Root serial protocol tunneled over TCP (commonly port 10001) used by ATG consoles such as the TLS-3xx and TLS-4xx series found in fuel-station forecourts. Issues the I20100 in-tank inventory and I90200 system-revision commands and returns the station name, console software and module revisions, tank count, and per-tank product names.
Port: 10001
ot

`backupexec`
TCP
`bacnet` — BACnet/IP
UDP
BACnet/IP is a building-automation protocol used for HVAC, lighting, access control, and other building systems. Issues Who-Is and ReadProperty requests and returns the device instance, vendor, model, firmware, and a summary of objects and routing devices behind the gateway.
Ports: 46808, 47808, 47809, 47810, 47811, 47812, 47813, 47814, 47815, 47816, 47817, 47818, 47819, 47820, 47821, 47822, 47823, 47824, 48808
backplanebuilding-automationdeepot
`bedrock` — Minecraft Bedrock
UDP
Minecraft Bedrock is the game-server query and discovery protocol used by Minecraft Bedrock Edition clients. Sends an Unconnected Ping and returns the server uptime, GUID, and the raw advertisement payload (MOTD and version fields).
Port: 19132
gaminglight
`bgp` — Border Gateway Protocol
TCP
Border Gateway Protocol is the inter-domain routing protocol used between autonomous systems on the Internet and inside large networks. Attempts an OPEN exchange and returns the BGP version, advertised AS number, BGP identifier, and supported capabilities.
Port: 179
routing
`bitdefender-app` — Bitdefender Endpoint
TCP
Bitdefender Endpoint is a proprietary check-in channel exposed by Bitdefender, NETGEAR Armor, and ThreatTrack mobile-security and VPN agents on Android and iOS endpoints. runZero passively matches the agent's JSON app_id payload from observed banners and reports the Bitdefender vendor along with the application identifier and software version of each detected product.
Port: 7519
lightmgmtsecurity
`bjnp` — Canon BJNP
UDP
Canon BJNP is a vendor-specific printing and scanning protocol used by Canon's network drivers. Sends a BJNP discovery probe and returns the device type (printer or scanner), MAC address, and IPv4/IPv6 address advertised in the reply.
Ports: 8611, 8612
discoveryprinting
`brother` — Brother Network Discovery
UDP
Brother Network Discovery is a UDP-based broadcast protocol used by Brother printers and multi-function devices to announce themselves and respond to driver/management discovery probes. runZero parses the response to recover the printer model, firmware revision, and serial number.
discoveryiot
`brother-scanner` — Brother Network Discovery
TCP

Brother Network Discovery is the scanner protocol used by Brother network devices. Identifies the service from its +OK 200 / -NG 401 reply and tags the asset as a Brother scanner.
Ports: 54921, 54922, 54923
discoverylightprinting
`bsap-ip` — Emerson BSAP/IP
UDP
Emerson BSAP/IP is the Bristol Standard Asynchronous Protocol over IP used by Emerson ControlWave and Bristol RTUs in oil & gas SCADA. Queries node identification and, when enabled, walks the BSAP local-address hierarchy to enumerate child RTUs, returning RTU identity, firmware, and topology summaries.
Ports: 1234, 1235
backplanedeepot
`c12.22` — ANSI C12.22
TCP
ANSI C12.22 is the metering-network transport used by electric-utility advanced-metering-infrastructure head-ends and relays. Issues an EPSEM Identification request and returns the device serial number, ED class, and C12.22 standard version.
Port: 1153
backplanedeepot
`c37118` — IEEE C37.118 Synchrophasor
TCP
IEEE C37.118 Synchrophasor is the streaming protocol used by Phasor Measurement Units and Phasor Data Concentrators on the electric grid. Requests CFG-2 and header frames and returns PMU/PDC identification, station and channel naming, and reporting rate.
Port: 4712
backplanedeepot
`cacti` — Cacti Network Monitoring Web UI
TCP
Cacti Network Monitoring Web UI is the open-source RRDtool-based network graphing and monitoring console deployed by network and IT operations teams. The HTTP extractor matches the Cacti login page title, parses the embedded versionInfo string, and returns the cacti.version attribute and Cacti software identification.
lighttlsweb
`cassandra` — Apache Cassandra CQL
TCP
Apache Cassandra CQL is the native client protocol used to query the Cassandra wide-column store. Performs a STARTUP/OPTIONS exchange and returns the cluster name, CQL version, and Cassandra release.
Ports: 9042, 9160
databasetls
`cdp` — Cisco Discovery Protocol
TCPUDP
Cisco Discovery Protocol is a Layer-2 announcement protocol used by Cisco devices to advertise themselves to neighbors. Passively decodes CDP frames and returns the device ID, software version, platform, capabilities, native VLAN, and management addresses advertised by the neighbor.
discoverymgmtpassive
`ceph`
TCP

Ceph is a cluster-internal messenger protocol used by Ceph distributed storage daemons (mon, mgr, osd, mds) to communicate. runZero identifies Ceph services from the "ceph v" banner returned on connection and tags the asset as a Ceph storage node.
storage
`chargen` — Character Generator
TCP
Character Generator is a legacy diagnostic service (RFC 864) that emits a stream of printable characters. Records the response banner to confirm the service.
legacylight
`checkmk` — Checkmk Agent
TCP
Checkmk Agent is the host-side metric exporter used by the Checkmk server to collect host metrics. Reads the agent banner and returns the agent version, build date, host operating system, architecture, and configured hostname.
Port: 6556
mgmtmonitoring
`chromecast` — Google Chromecast
TCP
Chromecast is the Google Cast device discovery and control protocol used by Chromecast, Google TV, Google Nest Hub, and Cast-enabled speakers. Detected via the _googlecast._tcp mDNS record and the device's HTTP /setup/eureka_info endpoint, which exposes the device name, model, build, and timezone.
discoveryiot
`cip` — Common Industrial Protocol
TCPUDP
Common Industrial Protocol is an industrial automation protocol used by Rockwell/Allen-Bradley and other vendors for PLC and I/O communication. Issues the List Identity, List Services, and List Interfaces commands to return vendor, product code, revision, serial, product name, and device-type information for the controller. Depending on the configured options and safety gates, it may also issue Get Attributes All requests and to collect additional information for any backplane modules.
Port: 44818
backplanedeepot
`cisco-phone` — Cisco IP Phone Web Interface
TCP
Cisco IP Phone Web Interface is the embedded HTTP server exposed by Cisco SPA, 7900-series, 8800-series, and Unified IP Phones. runZero attributes services as cisco-phone from Recog banner matches and uses the response to recover the model, firmware version, MAC address, and call-manager configuration.
clearvoip
`ciscosmi` — Cisco Smart Install
TCP
Cisco Smart Install is a zero-touch deployment protocol commonly abused for unauthenticated configuration access. Sends the Smart Install probe, tags the asset as Cisco IOS, and records the raw response payload.
Port: 4786
mgmt
`citrix` — Citrix ICA Browser
UDP

Citrix ICA Browser is a UDP/1604 service used by Citrix XenApp and Virtual Apps and Desktops clients to locate published applications and server farms. runZero sends an ICA Browser request and parses the reply to return the server-farm name and the list of advertised published applications.
Port: 1604
discoveryremote-access
`cldap` — Connectionless LDAP
UDP
Connectionless LDAP is a UDP-based directory protocol used to query directory servers without setting up a TCP connection. Issues a rootDSE search and returns the LDAP attributes parsed from the reply (vendor name and version, supported controls, extensions, capabilities, and SASL mechanisms).
Port: 389
directorydiscovery
`click` — Click Modular Router
TCP
Click Modular Router is the control socket exposed by hosts running the Click software router. Identifies the service from the Click::ControlSocket banner and tags the asset accordingly.
Port: 7734
lightmgmt
`coap`
UDP
CoAP is the Constrained Application Protocol (RFC 7252) used by constrained devices and IoT deployments for resource-oriented messaging. Issues a GET on /.well-known/core and returns the CoAP version, message type, response code, options, content format, and the resource list or payload from the reply.
Ports: 5683, 5684
iot
`cockpit` — Cockpit Linux Web Console
TCP
Cockpit Linux Web Console is the web-based Linux server-management console shipped with Red Hat, Fedora, CentOS Stream, and Debian, typically served over TLS on TCP/9090. The HTTP extractor matches the embedded environment JSON in the login page and returns the host name, OS pretty name, and OS variant identifiers.
lighttlsweb
`codesys` — CODESYS V3 Runtime
TCP
CODESYS V3 Runtime is the IEC 61131-3 controller runtime used by many OEMs (WAGO, Beckhoff, Schneider, Eaton, ...). Issues the runtime identification request and returns the runtime vendor, product, version, and target identification.
Ports: 1200, 1217, 2455, 11740
ot
`codesys2` — CODESYS V2 Runtime
TCP
CODESYS V2 Runtime is the older 3S CODESYS V2 controller runtime used by industrial controllers from many OEMs. Performs the V2 login probe and returns the runtime version and target identification.
Ports: 1200, 2455
ot
`cognex` — Cognex In-Sight

TCP
Cognex In-Sight is an operator-access service exposed by Cognex In-Sight industrial machine-vision cameras for configuration and monitoring on factory networks. runZero identifies In-Sight cameras from the proprietary banner returned on connection and tags the asset as a Cognex vision system.
lightot
`common-socket-connection` — Raritan Common Socket Connection
TCP
Raritan Common Socket Connection is a proprietary management transport used by Raritan KVM-over-IP switches, Dominion serial consoles, and rack PDUs, typically on TCP/5000. runZero identifies CSC services from the Raritan banner returned on connection and tags the asset as a Raritan management device.
clearlightmgmt
`companion-link` — Apple CompanionLink
TCP
Apple CompanionLink is a discovery and pairing service used by Apple devices (iOS, macOS, tvOS, HomePod) to negotiate AirPlay, HomeKit, and Continuity sessions with paired peers, advertised via _companion-link._tcp on Bonjour. runZero attributes the service from the mDNS / port hint and reports the asset as Apple CompanionLink.
iotlight
`comtrol` — Comtrol Device Discovery
TCP
Comtrol Device Discovery is the broadcast protocol used by Comtrol/Pepperl+Fuchs RocketLinx switches and DeviceMaster serial servers to advertise their presence on the network. runZero parses the response to recover the model name, hardware/firmware revision, MAC address, IP configuration, and serial number.
discoveryot
`confluence` — Atlassian Confluence
TCP
Atlassian Confluence is a wiki and team-collaboration server available in Server, Data Center, and Cloud editions. The HTTP probe fetches Confluence login, dashboard, and version endpoints and returns the product name, edition, and build number.
lighttlsweb
`consul` — HashiCorp Consul
TCP
Consul is HashiCorp's service-mesh and key/value store. Detected on the Consul HTTP API (TCP/8500), where the /v1/status/leader and /v1/agent/self endpoints disclose the datacenter, node name, build version, and Raft leader of the cluster.
Port: 8500
clearmgmt
`couchdb` — Apache CouchDB
TCP
CouchDB is the Apache document-oriented database fronted by an HTTP/JSON API. Detected from the welcome document at /, which discloses the CouchDB version, vendor name, UUID, and -- on misconfigured deployments -- whether the admin party is enabled.
Port: 5984
clearmgmt
`crestron` — Crestron Discovery
UDP

Crestron Discovery is a vendor protocol used to locate Crestron control processors and AV equipment. Sends the discovery probe and returns the device hostname, model, and firmware version.
Port: 41794
building-automationdiscoverylight
`crimsonv3` — Red Lion Crimson 3
TCP
Red Lion Crimson 3 is a configuration and runtime protocol used to read data from Red Lion Graphite, DA-series, and other industrial HMIs and gateways. Reads the manufacturer (register 0x012b) and model (register 0x012a) registers and returns those strings.
Port: 789
ot
`crowd` — Atlassian Crowd
TCP
Atlassian Crowd is a centralized SSO and identity-management product. runZero attributes services as Crowd from Recog matches against the application's HTTP banners and login pages, recovering the product version and build information.
clearmgmt
`cspv4` — Allen-Bradley CSPv4 / PCCC
TCP
Allen-Bradley CSPv4 / PCCC is a legacy controller protocol used by SLC 5/05 and MicroLogix PLCs for register access. Issues a PCCC identify request and returns the controller family, processor type, and series/revision string.
Port: 2222
backplanedeepot
`cups` — Common Unix Printing System
TCP
CUPS is the Common Unix Printing System administrative web interface. runZero matches the CUPS HTTP banner and the IPP server header, returning the cupsd version and host operating system.
Port: 631
cleariot
`dahua-dhip` — Dahua DHIP
UDP
Dahua DHIP is a proprietary discovery and management protocol used by Dahua and OEM-rebranded IP cameras and NVRs (Amcrest, Lorex, and similar). Sends the single-byte DHIP discovery probe and returns the device serial, machine name, vendor, firmware version, MAC, and IPv4/IPv6 configuration.
Port: 37810
discoveryiotlight
`daytime`
TCPUDP
Daytime is a legacy diagnostic service (RFC 867) that returns the current date and time. Records the response banner, parses the timestamp, and infers an OS hint from the format.
Port: 13
legacylight
`db2` — IBM Db2
TCPUDP
IBM Db2 is a relational database protocol used by Db2 LUW and Db2 for z/OS clients. Performs a DRDA EXCSAT/ACCSEC exchange and returns the server product identifier,

version, and platform.
Port: 523
database
`dcerpc` — DCE/RPC Endpoint Mapper
TCP
DCE/RPC Endpoint Mapper is the RPC locator used to enumerate RPC services on Windows hosts. Queries the endpoint mapper and returns a summary of registered RPC interfaces, their UUIDs, versions, and bindings.
Ports: 135, 593
mgmt
`dhcp`
UDP
DHCP is the Dynamic Host Configuration Protocol used to lease IPv4 addresses and network configuration. Sends a DHCPDISCOVER and returns the offered server identifier, lease parameters, and any vendor-class identification revealed by the response.
Ports: 67, 68
discovery
`diameter` — Diameter (TCP)
TCP
Diameter (TCP) is an authentication, authorization, and accounting protocol over TCP; the successor to RADIUS, widely used in mobile-core networks. Sends a Capabilities-Exchange-Request and returns the origin host, realm, vendor, product name, and supported applications.
Port: 3868
aaamobile-coretls
`diametersctp` — Diameter (SCTP)
SCTPTCP
Diameter (SCTP) is the Diameter (RFC 6733) AAA protocol carried over SCTP, used between mobile-core elements (Diameter Edge Agent, DRA, HSS, PCRF, MME). Sends a Capabilities-Exchange-Request and returns the origin host, realm, vendor, product name, and supported applications.
Port: 3868
aaamobile-coretls
`digi` — Digi ADDP
UDP
Digi ADDP is the Advanced Device Discovery Protocol used by Digi International serial servers, cellular gateways (TransPort, IX, EX), and embedded modules to advertise themselves on the local network. runZero sends DIGI, DVKT, and DGDP discovery requests on UDP/2362 and parses the TLV reply to return the device MAC, IP, model, firmware version, and hardware revision.
Port: 2362
discoveryiotlight
`digi-realport` — Digi RealPort
TCP
Digi RealPort is a serial-over-TCP service exposed by Digi serial server appliances. Sends a RealPort hello and parses the product name and exposed serial-port count from the handshake reply.
Ports: 771, 1027
iotlightserial
`dnp3`
TCP

DNP3 is Distributed Network Protocol 3 (IEEE 1815) used in electric, water, and oil & gas SCADA between control centers (masters) and RTUs/IEDs (outstations). Performs an unsolicited link-layer test and an Object Group 0 read and returns the outstation address, vendor, model, firmware, and device-attributes summary.
Port: 20000
backplanedeepot
`dns`
TCPUDP
DNS is the Domain Name System used to resolve hostnames to addresses and other resource records. Issues version.bind, hostname.bind, and recursion-test queries and returns the resolver software identification, recursion availability, and observed CHAOS-class metadata.
Port: 53
naming
`docker` — Docker Engine API
TCP
Docker Engine API is the HTTP control plane exposed by dockerd. runZero queries /version and /info on unauthenticated daemons (typically TCP/2375 or 2376) to recover the engine version, API version, kernel version, operating system, architecture, and container runtime.
Ports: 2375, 2376
clearmgmt
`doip` — Diagnostics over IP
TCPUDP
Diagnostics over IP is an automotive diagnostics protocol used to reach in-vehicle ECUs (UDS/KWP) over Ethernet. Issues a Vehicle Identification Request and (when enabled) Entity Status, returning VIN, EID/GID, logical addresses, and reachable ECU summaries behind the gateway.
Port: 13400
automotivebackplanedeepot
`dotnet-remoting` — .NET Remoting
TCP
.NET Remoting is a Microsoft RPC framework used by legacy .NET Framework applications for cross-process and cross-host RPC. Identifies the service from the .NET Remoting binary-protocol prefix in the connection banner and saves the raw banner.
Port: 9090
mgmt
`drbd`
TCP
DRBD is the Distributed Replicated Block Device protocol used to replicate block devices between Linux nodes for high availability. Identifies the service from the connection-error banner observed on TCP/8787 and saves the raw banner.
Port: 8787
lightstorage
`drobo-nasd` — Drobo NASd
TCP
Drobo NASd is the management daemon used by Drobo Dashboard to administer Drobo storage appliances. Identifies the daemon from the DRINASD banner returned on TCP/5000 and saves the raw banner.
Port: 5000
lightmgmtstorage
`dtls`

UDP
DTLS is Datagram Transport Layer Security (RFC 6347/9147), the UDP/SCTP variant of TLS, used by WebRTC, CoAP, OpenVPN, EAP-TTLS, and other datagram services. Performs a DTLS ClientHello and returns the negotiated version, cipher suite, and any presented certificate metadata.
Ports: 443, 3391, 4433, 5246, 5349, 5684, 12346, 12366, 12386, 12406, 12426
encrypted
`echo`
TCPUDP
Echo is a legacy diagnostic service (RFC 862) that echoes received bytes. Records the response to confirm the service and to detect amplification-capable hosts.
Port: 7
legacylight
`eero-ebid` — eero EBID
UDP
eero EBID is a proprietary discovery protocol used by Amazon eero mesh Wi-Fi access points to advertise the extender beacon identifier between mesh nodes on the local segment. runZero attributes the service from the EBID hint and applies eero-specific fingerprinting to identify the asset as an eero mesh extender.
discoveryiotpassive
`eerogw` — eero Gateway
UDP
eero Gateway is a proprietary discovery protocol used by Amazon eero mesh gateways to advertise themselves to companion mobile applications on the local network. runZero attributes the service from the eero gateway hint and applies eero-specific fingerprinting to identify the asset as an eero gateway node.
discoveryiotpassive
`elasticsearch`
TCP
Elasticsearch is the Elastic search/analytics engine. runZero queries the root HTTP endpoint to recover the cluster name, node name, Elasticsearch version, Lucene version, and build hash.
Port: 9200
clearmgmt
`epm` — DCE/RPC Endpoint Mapper Service
TCP
EPM is the surface name used by runZero for services attributed to the Microsoft DCE/RPC Endpoint Mapper following Recog/banner-based fingerprinting. The lower-level wire protocol is decoded as dcerpc; this label captures hosts where only fingerprint evidence (banners, version strings) was available.
Port: 135
clearmgmt
`epmd` — Erlang Port Mapper Daemon
TCP
EPMD is the Erlang Port Mapper Daemon used by Erlang and Elixir distributed nodes (including RabbitMQ and CouchDB) to advertise registered node names and the dynamic ports they listen on. runZero issues the NAMES_REQ to enumerate registered nodes and their listening ports.
Port: 4369
clearmgmt

`epo` — Trellix/McAfee ePolicy Orchestrator (ePO)
TCP
Trellix/McAfee ePolicy Orchestrator (ePO) is the central management console for Trellix (formerly McAfee) endpoint-security agents. The HTTP probe fetches the ePO login page and returns the product name, build number, and version metadata exposed in the page markup.
lightsecuritytlsweb
`erlangdp` — Erlang Distribution
TCP
Erlang Distribution is the inter-node messaging protocol used between Erlang and Elixir nodes. Queries EPMD for registered node names and (when nodes are discovered) performs a distribution handshake on the first node, returning the EPMD names list, node name and hostname, distribution version, supported flags, and handshake status.
Port: 4369
messaging
`erldp` — Erlang Distribution Protocol
TCP
Erlang Distribution Protocol is the distribution protocol used between Erlang and Elixir VM nodes for inter-node messaging (also commonly referred to as ErlDP). runZero matches the Erlang distribution handshake on the wire and reports the protocol alongside the active erlangdp scanner so port-scan results are tagged consistently with EPMD-discovered nodes.
clearmessaging
`etcd` — etcd v3 API
TCP
etcd is the distributed key-value store used by Kubernetes and other CoreOS-derived projects. Detected via the v3 HTTP/gRPC API (typically TCP/2379), where /version reports the etcd-server and etcd-cluster versions.
Port: 2379
clearmgmt
`etcd2` — etcd v2 API
TCP
etcd2 is the legacy v2 HTTP API exposed by older etcd deployments. Detected via /v2/stats/self and /version, which expose the cluster name, member ID, and etcd version on unauthenticated installations.
Ports: 2379, 4001
clearmgmt
`ethercat`
UDP
EtherCAT (Ethernet for Control Automation Technology, IEC 61158) fieldbus used for high-speed motion control and distributed I/O on machine-control and CNC segments. Queries master and slave registers and returns the master vendor identification and a summary of slaves discovered on the segment.
Port: 34980
backplanedeepot
`fgfm` — FortiGate to FortiManager
TCP
FGFM is the proprietary TLS-wrapped management protocol used by Fortinet FortiGate firewalls to register with and receive configuration from a FortiManager. Detected by the FGFM TLS server certificate and banner; runZero records the device serial number and model where exposed.
Port: 541

clearmgmt
`finger`
TCP
Finger is a legacy user-information service (RFC 1288) historically exposed on TCP/79, today seen mostly on Cisco IOS devices and embedded printers. runZero reads the Finger banner returned on connection and extracts the printer model (HP JetDirect-style) or Cisco IOS identification, and reports the service for legacy-protocol exposure tracking.
legacy
`fins` — Omron FINS
UDP
Omron FINS is the Factory Interface Network Service used by Omron CJ, CS, NJ, and NX PLCs and related automation devices on factory floors. runZero records FINS-derived asset attributes (controller model, firmware) emitted by the active omronfins scanner and uses the FINS protocol identifier when categorizing assets and assigning OT asset functions.
Port: 9600
ot
`firebird` — Firebird SQL
TCP
Firebird SQL is the relational database wire protocol used by the Firebird open-source database engine. Performs the Firebird connection handshake and returns the server architecture, protocol version, and Firebird release.
Port: 3050
database
`focas` — Fanuc FOCAS
TCP
Fanuc FOCAS is an Ethernet protocol used to monitor and control Fanuc CNC machine tools and robots (Open CNC API Specification, FOCAS2/Ethernet). Issues the system-info call and returns the CNC series, version, machine number, and (when enabled) per-path machining data.
Port: 8193
backplanedeepot
`fortigate-to-fortimanager`
TCP
FortiGate-to-FortiManager is the Fortinet FGFM management protocol used by FortiManager to manage FortiGate firewalls. Inspects the FGFM TLS handshake and returns the FortiGate model, firmware, and serial number embedded in the certificate.
Port: 541
mgmtsecurity
`fox` — Tridium Niagara Fox
TCP
Tridium Niagara Fox is the building-automation control protocol used by Niagara Framework JACE controllers and supervisors. Sends the Fox hello and returns the Fox version, station name, host ID, host name, OS name and version, JVM name and version, brand identifier, and authentication agent.
Ports: 1911, 4911
building-automationot
`ftp`
TCP
FTP is the standardized file-transfer protocol (RFC 959). Reads the FTP greeting and issues SYST/HELP/AUTH probes, returning the server software, system type, supported features,

and TLS-availability indicators.
Ports: 21, 2121, 9090
clearfiletls
`ganglia`
TCP
Ganglia is a distributed monitoring system commonly deployed on HPC clusters and Linux server farms. runZero identifies Ganglia services from the GANGLIA_XML document returned by gmond/gmetad and captures the banner so cluster identification and host metrics are available for inventory.
monitoring
`gesrtp` — GE SRTP
TCP
GE SRTP is the Service Request Transport Protocol used to communicate with GE/Emerson PACSystems, Series 90, and RX3i/RX7i PLCs. Issues a controller-identification request and returns the model, firmware, sweep state, and slot configuration.
Port: 18245
ot
`giop` — GIOP / CORBA IIOP
TCP
GIOP / CORBA IIOP is the OMG General Inter-ORB Protocol (the wire format under CORBA IIOP). Identifies the service from the GIOP magic in the connection banner.
Port: 535
mgmt
`git` — Git Smart Protocol
TCP
Git Smart Protocol is the native transport used by git:// servers for clones, fetches, and pushes. Sends an upload-pack advertisement request and returns the advertised refs summary, server capabilities, and HEAD reference.
Port: 9418
file
`git-http` — Git Smart HTTP Service
TCP
Git Smart HTTP Service is the Git smart-HTTP transport (git-upload-pack and git-receive-pack endpoints) used by GitLab, Gitea, Bitbucket, cgit, and bare git-http servers for clones, fetches, and pushes. The HTTP extractor matches the _gitlab_session cookie and parses the manifest body, returning the git-http protocol attribution and the GitLab manifest hash.
lighttlsweb
`googlewifi` — Google Wifi / Nest Wifi
TCP
Google Wifi is the local management API exposed by Google Wifi and Nest Wifi mesh access points. Detected via mDNS (_googlecast._tcp) and the local HTTP setup endpoints, which reveal the device's hardware model, build version, and mesh role.
discoveryiot
`gpsd`
TCP
GPSD is the GPS daemon JSON-over-TCP protocol used to share location and timing data from connected GNSS receivers. Sends a ?WATCH request and identifies the service from the GPSD banner returned in the response.
Port: 2947
iot

`gtpc` — GTP-C
UDP
GTP-C is the GPRS Tunneling Protocol control plane that carries signaling between mobile-core nodes (SGSN/GGSN, MME/SGW/PGW). Sends an Echo Request and returns the GTP version, restart counter, and supported features.
Ports: 2123, 2152
mobile-core
`gtpprime` — GTP'
UDP
GTP' is the GTP charging variant used to ship CDRs from mobile network elements to a Charging Gateway. Sends an Echo Request and returns the GTP' version and node identification.
Port: 3386
mobile-core
`gtpu` — GTP-U
UDP
GTP-U is the GPRS Tunneling Protocol user plane that encapsulates subscriber traffic between mobile-core nodes and base stations. Sends an Echo Request and returns the GTP-U version and observed extension-header support.
Port: 2152
mobile-core
`gvcp` — GigE Vision Control
UDP
GigE Vision Control is the AIA GVCP protocol used to discover, configure, and trigger industrial machine-vision cameras over Ethernet. Sends a Discovery_Cmd and returns the camera manufacturer, model, serial, firmware, MAC, and supported GVCP version.
Port: 3956
discoveryot
`gvsp` — GigE Vision Streaming
UDP
GigE Vision Streaming is the AIA GVSP protocol used to transport image and chunk data from machine-vision cameras to host applications. Passively classifies stream packets and returns the streaming state, packet format, and block identifier.
Port: 20202
otpassive
`hartip` — HART-IP
TCP
HART-IP is an industrial protocol used to tunnel HART process-instrument traffic over TCP/UDP through gateways and multiplexers. Issues HART command 0 and returns the gateway identification, and (when enabled) walks sub-device indices via Cmd 84 to enumerate connected field instruments.
Port: 5094
backplanedeepot
`hicp` — HMS HICP/SHICP
UDP
HMS HICP/SHICP is the HMS Industrial Networks discovery protocol used to discover and configure Anybus and Netbiter industrial gateways. Sends the HICP discovery probe and returns the device hostname, MAC, IP configuration, and firmware revision.
Port: 3250
discoveryot

`hiddiscoveryd` — HID DiscoveryD
UDP
HID DiscoveryD is the discovery service used to locate HID VertX/Edge access-control panels and readers. Sends the discovery probe and returns the device model, firmware, and primary network configuration.
Port: 4070
building-automationdiscoveryiotlight
`hikvision` — Hikvision IP Camera/NVR Web
TCP
Hikvision IP Camera/NVR Web is the HTTP management interface for Hikvision (and OEM) IP cameras and recorders. Identifies the product family from the WWW-Authenticate realm and pins the firmware version using either the embedded ?version= query strings on the login page assets, or the Last-Modified header on /doc/page/login.asp combined with a known build-date table.
cameraiotlightweb
`hostmeta` — Web Host Metadata
TCP
Web Host Metadata is the document defined by RFC 6415 and exposed at /.well-known/host-meta or host-meta.json, commonly used by federated-identity, WebFinger, and ActivityPub deployments. runZero sets the hostmeta protocol on the asset when the active HTTP probe successfully retrieves the document and surfaces the discovered links for inventory.
lightweb
`hsms` — SEMI HSMS / SECS-GEM
TCP
SEMI HSMS / SECS-GEM is the High-Speed SECS Message Services transport that carries SECS-II/GEM messages for semiconductor fab equipment (SEMI E37). Performs the HSMS Select handshake and an S1F1 Are-You-There, returning the equipment model identifier, software revision, and (when enabled) Equipment Constant subsystem summary.
Port: 5000
backplanedeepot
`http`
TCP
HTTP is the Hypertext Transfer Protocol used by the World Wide Web and most modern APIs. Issues HEAD/GET probes and runs HTTP-specific extractors, returning server software, response codes and headers, page titles and generators, favicons, and any application fingerprints recognized by extractor rules.
Ports: 80, 3000, 4567, 5000, 5985, 8000, 8001, 8080, 8081, 8082, 8200, 8443, 8888, 9001, 9080, 9090, 9100
tlsweb
`http2` — HTTP/2
TCP
HTTP/2 is a binary, multiplexed framing protocol for HTTP (RFC 7540), negotiated via TLS ALPN or the HTTP/2 cleartext upgrade. runZero negotiates HTTP/2 when offered and feeds the response into the standard HTTP analyzer so headers, server identification, and TLS attributes are captured.
tlsweb
`https`
TCP
HTTPS is HTTP carried over TLS, the dominant transport for web applications, REST APIs, and management UIs. Implemented in runZero as the standard HTTP scanner over a TLS

connection; the protocol is reported as https (rather than http) whenever the connection negotiates TLS or the port hint is flagged as TLS-only, and the full TLS handshake metadata (certificate, ciphers, fingerprints) is recorded alongside the HTTP response.
encryptedtlsweb
`iax2`
UDP
IAX2 is the Inter-Asterisk eXchange version 2 VoIP signaling and media protocol used between Asterisk PBXes. Sends a POKE and returns the responder's IAX2 version and PBX identification.
Port: 4569
voip
`icmp` — ICMP Echo (Ping)
TCP
ICMP Echo (Ping) is the IPv4/IPv6 ICMP Echo Request/Reply exchange (RFC 792, RFC 4443) issued by the host-discovery probe to confirm host liveness, capture round-trip times, and observe TTL/Hop-Limit and IP TOS fields.
discoverylight
`ics-trace` — ICS Passive Trace
TCP
ICS Passive Trace is a synthetic protocol used internally to record evidence from passive analysis of ICS/OT traffic when active OT probing is disabled. runZero attributes the asset under ics-trace and emits the protocol identifiers and attributes observed in the passive trace.
otpassive
`ident`
TCP
Ident is the Identification Protocol (RFC 1413), a legacy user-identity lookup service. Sends an ident query against the connecting socket and returns the operating-system field and any user-identity string disclosed by the response.
Port: 113
legacylight
`idrac` — Dell iDRAC
TCP
iDRAC is the Dell Integrated Dell Remote Access Controller out-of-band management interface. runZero attributes services as iDRAC from the SSH/HTTPS banners, redfish endpoints, and TLS certificates issued for the controller, recovering the firmware version and service tag.
clearmgmt
`iec104` — IEC 60870-5-104
TCP
IEC 60870-5-104 is a SCADA telecontrol protocol used between control centers and RTUs/substation gateways, primarily in electric power and rail. Sends TESTFR (and, when enabled, STARTDT/General Interrogation) and returns the common ASDU address, originator address, and any device-identity ASDUs received.
Port: 2404
deepot
`iec61850-goose` — IEC 61850 GOOSE
TCP
IEC 61850 GOOSE is the Generic Object Oriented Substation Event Layer-2 multicast protocol (EtherType 0x88B8) used by substation IEDs for peer-to-peer trip and status signaling.

runZero passively decodes GOOSE frames, attributes the publisher MAC as an IEC 61850 IED, and records the GoCB reference and dataset.
multicastotpassive
`iec61850-mms` — IEC 61850 MMS
TCP
IEC 61850 MMS is the Manufacturing Message Specification mapping used by substation Intelligent Electronic Devices for monitoring, control, and reporting. Opens an MMS session and (when enabled) issues Identify, returning the IED vendor, model, firmware, and logical-device summary.
Port: 102
backplanedeepot
`iec61850-sv` — IEC 61850 Sampled Values
TCP
IEC 61850 Sampled Values is the Layer-2 multicast protocol (EtherType 0x88BA) used by substation merging units to publish synchronized current and voltage measurements to protection and control IEDs. runZero passively decodes SV frames, attributes the publisher MAC as a merging unit, and records the SvCB reference and dataset.
multicastotpassive
`igel` — IGEL Discovery
UDP
IGEL Discovery is the discovery protocol used by IGEL OS thin clients to advertise themselves to the IGEL Universal Management Suite (UMS), deployed in VDI and remote-desktop environments. Sends the IGEL discovery probe and returns the endpoint hostname, hardware model, IGEL OS version, MAC, and the UMS-server registration state.
Port: 30005
discoverylightmgmt
`igel-discovery` — IGEL UMS Discovery
UDP
IGEL UMS Discovery is the UDP broadcast used by IGEL OS thin clients to locate their Universal Management Suite server. Distinct from the igel management protocol decoded over TCP, this entry captures discovery-only sightings on UDP/30005.
Port: 30005
discoverymgmt
`iis` — Microsoft Internet Information Services (IIS)
TCP
Microsoft Internet Information Services (IIS) is the HTTP server bundled with Windows Server, commonly hosting ASP.NET, Outlook Web Access, and SharePoint. The HTTP fingerprinter inspects Server and X-Powered-By headers and default landing pages and returns the IIS version and ASP.NET version hints.
lighttlsweb
`ike`
UDP
IKE is the Internet Key Exchange protocol used to negotiate IPsec security associations. Sends IKEv1/IKEv2 SA proposals and vendor-ID probes and returns the negotiated proposal summary, supported transforms, and any vendor-ID strings that disclose the gateway implementation.
Ports: 500, 4500
vpn
`ikev2`
UDP

IKEv2 is the Internet Key Exchange version 2 protocol (RFC 7296) used by IPsec VPN gateways on UDP/500 and UDP/4500. runZero sends an IKE_SA_INIT request and parses the responder SPI, accepted transform set, and any vendor-ID payloads that disclose the gateway implementation.
Ports: 500, 4500
vpn
`infinispan` — Infinispan Hot Rod
TCP
Infinispan Hot Rod is a binary client/server protocol used by Red Hat Data Grid and JBoss-family caches. Performs the Hot Rod ping and returns the server version, topology identifier, and supported protocol version.
Port: 11222
database
`influxdb`
TCP
InfluxDB is the time-series database from InfluxData. runZero queries /ping and /health on the HTTP API to recover the server build, X-Influxdb-Version header, and on permissive deployments the list of available databases.
Port: 8086
clearmgmt
`intermapper`
TCP
InterMapper is the probe interface exposed by Fortra (formerly HelpSystems) InterMapper network-monitoring agents installed on monitored servers and appliances to report status to a central InterMapper server. Reads the InterMapper service banner and returns the product name and agent version.
Port: 8181
lightmonitoring
`ipmi`
UDP
IPMI is the Intelligent Platform Management Interface used for out-of-band server management on BMCs (iLO, iDRAC, IMM). Performs an IPMI 2.0 RMCP+ Get Channel Authentication Capabilities exchange and, when credentials are configured, returns supported cipher suites, authentication algorithms, and the BMC vendor/firmware identification.
Port: 623
authmgmt
`ipp-browse` — IPP Browse
UDP
IPP Browse is the legacy CUPS browse protocol used by macOS and Linux print servers to advertise IPP print queues to the local segment via UDP/631 broadcasts. runZero passively decodes browse packets and active responses and returns the advertised queue URI, printer name, and CUPS server identification, attributing the asset as a print server.
Port: 631
discoverylightmulticastprinting
`ippbrowse` — IPP Browse
UDP
IPP Browse is the legacy CUPS browse protocol used by Unix print servers to advertise IPP print queues via UDP/631 broadcasts. Passively decodes browse packets and returns the advertised queue URI, printer name, and CUPS server identification.
Port: 631

discoverylightmulticastprinting
`irc`
TCP
IRC is the Internet Relay Chat protocol, a text-based group messaging protocol. Reads the IRC server greeting and runs a NICK/USER probe, returning the server software, version, and 005-numeric capability summary.
Ports: 6667, 6668, 6669, 6697, 7000, 7001
clearmessagingtls
`iscsi`
TCP
iSCSI is the Internet Small Computer Systems Interface protocol used to expose block storage over IP. Sends a SendTargets request and returns the iSCSI Target Name list, target portals, and authentication-method summary.
Port: 3260
storage
`iua` — IUA (SCTP)
SCTP
IUA (SCTP) is the ISDN User Adaptation Layer (RFC 4233) carried over SCTP, used to backhaul ISDN signaling in SIGTRAN networks. Verifies the SCTP association and IUA payload protocol identifier and returns endpoint identification.
Port: 9900
voip
`jabber`
TCP
Jabber is the legacy product name for XMPP, still used by Cisco Jabber, ejabberd, Openfire, and similar chat/presence servers. runZero identifies Jabber services from the <stream:stream> or jabber.org-namespaced response returned on connection and tags the asset alongside the active xmpp scanner results.
messagingtls
`java-object` — Java Object Serialization
TCP
Java Object Serialization is the binary stream format produced by java.io.ObjectOutputStream, often indicating exposed RMI, JMX, or JBoss endpoints when seen on the wire. Inspects the magic header and returns the serialization-protocol version and any class-name hints disclosed in the stream.
lightmgmt
`java-rmi` — Java RMI
TCP
Java RMI is the Remote Method Invocation protocol used by Java applications. Performs the JRMP handshake and a registry list, returning the RMI-registry version and the names and stub classes of bound objects.
Port: 1099
mgmt
`jdbc-hsqldb` — HyperSQL JDBC
TCP
HyperSQL JDBC is the JDBC database server protocol used by the HyperSQL (HSQLDB) engine. Identifies the service from the 'HSQLDB JDBC Network Listener' banner returned on connection.
Port: 9001
database

`jdwp` — Java Debug Wire Protocol
TCP
Java Debug Wire Protocol is the unauthenticated JVM debugging transport used by IDEs and debuggers to control a JVM. Performs the JDWP handshake and Version command, returning the JDK version, JVM vendor, and process description.
Ports: 3999, 5000, 5005, 8000, 8453, 8787, 8788, 9001, 18000
mgmt
`jetdirect` — HP JetDirect
TCP
HP JetDirect is the raw printing protocol on TCP/9100 (PJL banner port). Sends a PJL INFO ID/STATUS probe and returns the printer make/model, firmware, page count, and PJL device-attribute summary.
Ports: 9100, 9101, 9102
lightprinting
`jira` — Atlassian Jira
TCP
Atlassian Jira is an issue-tracking and project-management server available in Server, Data Center, and Cloud editions. The HTTP probe fetches Jira login, dashboard, and REST endpoints and returns the product name, edition, and build number.
Ports: 80, 443, 8080
lighttlsweb
`jms` — JMS / JMX-RMI Port Mapper
TCP
JMS / JMX-RMI Port Mapper is the OpenMQ/GlassFish imqbroker port-mapper service that publishes the names, transports, and ports of bound JMS and JMX-RMI endpoints (default TCP/7676). Queries the port mapper and returns the broker version and the bound endpoint names, transports, and ports.
Port: 7676
messagingmgmt
`kafka` — Apache Kafka
TCP
Apache Kafka is a distributed event-streaming wire protocol. Sends an ApiVersions request and returns the broker identifier, supported API versions, and (when an unauthenticated MetadataRequest is permitted) cluster and topic-name summaries.
Ports: 9092, 9093, 9094
messagingtls
`kasa` — TP-Link Kasa
TCP
TP-Link Kasa is the smart-home control protocol used by Kasa plugs, bulbs, and switches. Sends the obfuscated SYS_INFO query and returns the device alias, model, firmware, hardware version, and MAC.
Port: 9999
iotlight
`kerberos`
TCPUDP
Kerberos is the network authentication protocol used by Active Directory and many enterprise services. Sends an AS-REQ for a benign principal and returns the realm, KDC error code, and supported encryption types.
Ports: 88, 464, 749, 750
aaa

`knxnet` — KNXnet/IP
UDP
KNXnet/IP is the IP-tunneling encapsulation used to tunnel and route KNX building-automation telegrams (lighting, HVAC, shading). Sends a SEARCH_REQUEST and returns the device serial, MAC, KNX individual address, supported services, and friendly name.
Port: 3671
building-automationot
`l2t` — L2TP (UDP 1701)
UDP
L2TP (UDP 1701) is the Layer 2 Tunneling Protocol on UDP/1701, used by VPN concentrators and remote-access gateways (often paired with IPsec) to tunnel PPP sessions. Sends an L2TP SCCRQ and returns the host name, vendor name, and firmware revision AVPs disclosed by the responder.
Ports: 1701, 2228
vpn
`l2tp`
UDP
L2TP is the Layer 2 Tunneling Protocol used to carry PPP sessions, commonly with IPsec for VPN. Sends an SCCRQ and returns the host name, vendor name, and firmware revision AVPs.
Port: 1701
vpn
`landesk` — Ivanti / LANDesk Agent
TCP
Ivanti / LANDesk Agent is the endpoint-management agent for Ivanti (formerly LANDesk). Reads the agent banner and returns the agent version and bound management server.
Port: 9595
lightmgmt
`langflow`
TCP
Langflow is an open-source visual builder for LLM-powered applications. runZero attributes services as Langflow from Recog matches against the application's HTTP responses and OpenAPI document, recovering the application version.
aiclearweb
`lantronix` — Lantronix Discovery
UDP
Lantronix Discovery is the discovery protocol used to locate Lantronix serial-to-Ethernet device servers. Sends the discovery probe and returns the device model, firmware, MAC, and configured serial-port settings.
Port: 30718
discoveryiotlight
`ldap`
TCP
LDAP is the Lightweight Directory Access Protocol (RFC 4511) used by Active Directory, OpenLDAP, 389 Directory Server, and other directory services for authentication and identity lookups. Queries the rootDSE and returns the supported LDAP versions, naming contexts, supported controls, and any forest or domain identifiers exposed.
Ports: 389, 636, 3268, 3269
authdirectorytls
`lexmark` — Lexmark Discovery
TCP

Lexmark Discovery is the printer/MFP network discovery protocol used by Lexmark devices. Sends the discovery probe and returns the device model, serial, firmware, and printer/MFP capabilities.
Port: 10000
discoverylightprinting
`librechat`
TCP
LibreChat is an open-source self-hosted LLM chat front-end. runZero attributes services as LibreChat from Recog matches against the web UI banners and configuration endpoints.
Ports: 443, 3080
aiclearweb
`lldp` — Link Layer Discovery Protocol
TCP
Link Layer Discovery Protocol is the IEEE 802.1AB Layer-2 discovery protocol used by switches, routers, IP phones, and hypervisors to advertise themselves to neighbors. runZero passively decodes LLDP frames seen during the scan and returns the chassis ID, port ID, system name, system description, capabilities, and management addresses of each neighbor.
discoverymgmtmulticastpassive
`llmnr`
UDP
LLMNR is Link-Local Multicast Name Resolution (RFC 4795) used by Windows hosts to resolve single-label names on the local link when DNS is unavailable. Sends an LLMNR query and returns the responding hostname and the IP version of the answering host.
Port: 5355
lightmulticastnaming
`lockdownd` — Apple lockdownd
TCP
Apple lockdownd pairing service exposed on TCP/62078 by iPhone, iPad, and iPod touch devices and used by iTunes, Finder, Apple Configurator, and MDM tooling. Reads the lockdownd query response and returns the device class, product type, iOS or iPadOS version, serial number, and unique device identifier (UDID).
Port: 62078
lightmgmt
`lpd`
TCP
LPD is the standardized BSD line-printer protocol (RFC 1179). Sends a Receive-Job command and returns the raw printer banner from the daemon for downstream make and model fingerprinting.
Port: 515
lightprinting
`lsv2` — Heidenhain LSV/2
TCP
Heidenhain LSV/2 is a control protocol used by Heidenhain TNC CNC controls (TNC 640, iTNC 530, TNC 320). Queries control identification and returns the NC software type, version, and (when enabled) NC software-options bitmask.
Ports: 8000, 8001, 8002, 8003, 8004, 19000
deepot
`lwm2m` — OMA LwM2M
UDP

OMA LwM2M is the OMA Lightweight M2M device-management protocol layered on CoAP and used to manage constrained IoT endpoints, sensors, and cellular modules. Sends a CoAP GET for /.well-known/core and returns whether the LwM2M registration directory and bootstrap-server resources are advertised along with a server-implementation hint.
Ports: 5683, 5783
iotmgmt
`m2pa` — M2PA (SCTP)
SCTP
M2PA (SCTP) is the MTP2 Peer Adaptation Layer (RFC 4165) over SCTP used in SIGTRAN to carry SS7 MTP2 between signaling gateways. Establishes an SCTP association, verifies the M2PA payload protocol identifier, and returns the M2PA message class, message type, link state, and any error code or info string in the reply.
Port: 3565
mobile-corevoip
`m2ua` — M2UA (SCTP)
SCTP
M2UA (SCTP) is the MTP2 User Adaptation Layer (RFC 3331) over SCTP used in SIGTRAN deployments. Establishes an SCTP association, verifies the M2UA payload protocol identifier, and returns the M2UA message class, message type, and any error code or info string in the reply.
Port: 2904
mobile-corevoip
`m3ua` — M3UA (SCTP)
SCTP
M3UA (SCTP) is the MTP3 User Adaptation Layer (RFC 4666) over SCTP, the most common SS7-over-IP transport. Establishes an SCTP association, verifies the M3UA payload protocol identifier, and returns the M3UA message class, message type, and any error code or info string in the reply.
Port: 2905
mobile-corevoip
`managesieve`
TCP
ManageSieve is a protocol used by mail clients to manage Sieve mail-filter scripts on the server. Reads the capabilities response and returns the implementation name, version, supported SASL mechanisms, and STARTTLS availability.
Ports: 2000, 4190
emailtls
`matter`
UDP
Matter is the Connectivity Standards Alliance smart-home application protocol used by Matter-certified devices over Wi-Fi and Thread. runZero identifies Matter devices from _matter._tcp mDNS records, parses the VP and DT TXT fields, and resolves them against the bundled Matter vendor table to return the vendor name, product name, and device type.
discoveryiotpassive
`mbus-tcp` — M-Bus over TCP
TCP
M-Bus over TCP is the EN 13757 Meter-Bus protocol tunneled over TCP, used by gateways aggregating utility meters (heat, water, gas, electric). Sends REQ_UD2 to the gateway and (when enabled) walks primary addresses 1-250 to enumerate connected meters, returning meter manufacturer, identification, version, and medium.

Ports: 8888, 8889
backplanedeepot
`mcp` — Model Context Protocol
TCP
Model Context Protocol (MCP) is Anthropic's JSON-RPC protocol used by AI agents to connect to external tools and data sources. runZero attributes services as MCP from Recog matches against the server's HTTP/SSE handshake and the initialize response, which exposes the server name, version, and advertised capabilities.
aiclear
`mdns`
UDP
mDNS is Multicast DNS used by Bonjour, Avahi, and other zero-configuration networking stacks. Sends a service-enumeration query and returns the advertised service types, instance names, ports, hostnames, and TXT-record metadata.
Port: 5353
discoverylightmulticastnaming
`megaco` — Megaco / H.248
UDP
Megaco / H.248 is a media gateway control protocol (RFC 3525) used between softswitches and media gateways in carrier and enterprise VoIP networks. Sends a ServiceChange probe and returns the media-gateway identifier (MID) and the negotiated H.248 protocol version.
Ports: 2944, 2945
voip
`melsecq` — Mitsubishi MELSEC-Q
TCP
Mitsubishi MELSEC-Q is a protocol used to communicate with Mitsubishi MELSEC PLCs. Issues a CPU model-name read using SLMP 3E (and, when enabled, 4E) and returns the CPU model name and CPU type code along with the matching MELSEC product CPE.
Ports: 5006, 5007
backplanedeepot
`memcache` — Memcache (text)
TCPUDP
Memcache (text) is the ASCII text wire protocol exposed by the Memcached distributed in-memory key-value cache and compatible servers. Issues a text VERSION/STATS request and returns the daemon version, current connections, and items and bytes held in cache.
Port: 11211
cleardatabasetls
`meshcop` — Thread Mesh Commissioning
UDP
Thread Mesh Commissioning is the Mesh Commissioning Protocol used by Thread border routers (Apple HomePod, Google Nest Hub, eero, Nordic OTBR) to advertise commissioning endpoints to companion mobile apps. runZero attributes the asset as a Thread border router from the meshcop hint and surfaces the advertised network name, extended PAN ID, and vendor identification.
discoveryiotpassive
`mgcp`
UDP
MGCP is the Media Gateway Control Protocol (RFC 3435) used between call agents and media gateways in VoIP networks. Sends an AUEP probe and returns the response code along with any endpoint identifiers and packages disclosed in the reply.

Ports: 2427, 2727
voip
`mikrotik-bandwidth` — MikroTik Bandwidth Test Server
TCP
MikroTik Bandwidth Test Server is the proprietary throughput-testing service exposed by MikroTik RouterOS devices. runZero detects the listener via its banner; presence indicates the device is reachable for line-rate tests, which can be abused for amplification.
Port: 2000
clearmgmt
`mikrotikwinbox` — MikroTik Winbox
TCP
MikroTik Winbox is the protocol used by the Winbox utility to administer RouterOS devices. Sends the index-request and returns the RouterOS architecture, version, and bootloader identification.
Ports: 2000, 8291, 8728
mgmt
`milvus` — Milvus Vector Database
TCP
Milvus is an open-source vector database used by retrieval-augmented LLM applications. runZero attributes services as Milvus from Recog matches against the gRPC and HTTP endpoints and the management UI banners.
Port: 19530
aiclearmgmt
`minecraft` — Minecraft Java
TCP
Minecraft Java is the server query/list-ping protocol used by Minecraft Java Edition. Sends a status-request and returns the server MOTD, version, protocol number, current and maximum player counts, and any sample player names disclosed.
Port: 25565
gaming
`mms` — ISO 9506 MMS (IEC 61850)
TCP
ISO 9506 MMS (IEC 61850) is the Manufacturing Message Specification over RFC 1006/COTP, the application protocol used by IEC 61850 substation Intelligent Electronic Devices (IEDs). The probe issues an A-ASSOCIATE plus MMS Initiate-Request and parses the Initiate-Response for vendor identity, negotiated MMS version, and supported services.
Port: 102
backplanedeepot
`modbus` — Modbus/TCP
TCP
Modbus/TCP is a protocol used to read and write registers on PLCs, RTUs, drives, and meters. Issues a Read Device Identification (function 43/MEI 14) and returns the vendor, product code, revision, vendor URL, product name, and (when configured) extended identification objects.
Port: 502
backplanedeepot
`mongodb` — MongoDB Wire Protocol
TCP
MongoDB Wire Protocol is the document-database wire protocol used by MongoDB drivers. Sends an unauthenticated isMaster/hello and returns the server version, build environment,

replica-set role, and observed authentication requirements.
Ports: 27017, 27018, 27019, 28017
databasetls
`mountd` — NFS mountd
TCPUDP
NFS mountd is the companion daemon to NFS that authorizes mount requests and enumerates exports, registered through rpcbind on Unix file servers and NAS appliances. runZero locates mountd through rpcbind, sends an EXPORT (procedure 5) call for each advertised version, and returns the exported directory list and per-export host or netgroup access lists.
filestorage
`moxa` — Moxa Discovery
UDP
Moxa Discovery is the proprietary UDP service (4800/udp) used by the Moxa configuration utilities to locate NPort, OnCell, and MGate devices. Sends an identify request and returns the device MAC address and vendor, confirming a Moxa device.
Port: 4800
discoveryot
`mqtt`
TCP
MQTT is a lightweight publish/subscribe messaging protocol used by IoT devices and brokers. Sends a CONNECT and returns the broker's CONNACK response code, supported MQTT version, and any properties advertised by the broker.
Ports: 1883, 8883
iotmessagingtls
`mssql` — Microsoft SQL Server (TDS)
TCPUDP
Microsoft SQL Server (TDS) is the Tabular Data Stream protocol used by Microsoft SQL Server and Sybase ASE database engines for client-server query traffic. Sends a TDS PRELOGIN and returns the server version, encryption requirement, and named-instance identification.
Ports: 1433, 1434
databasetls
`mssql-replica` — Microsoft SQL Server Replica
TCP
Microsoft SQL Server Replica is the Always-On availability-group and database-mirroring endpoint that exchanges replica traffic on TCP/5022 separately from the user TDS endpoint on TCP/1433. runZero hints TCP/5022 as mssql-replica during the MSSQL probe and tags the asset as a SQL Server replica endpoint when the standard TDS handshake is not offered.
cleardatabaselight
`mtconnect`
TCP
MTConnect is a protocol used by CNC machine tools, robots, and additive-manufacturing systems to publish device state over an HTTP/XML REST API. Issues a GET /probe and returns the agent version, instance ID, sender host, and per-device manufacturer, model, serial number, and UUID.
Ports: 5000, 7878
discoveryot
`munin`
TCP

Munin is the protocol used by the Munin master to poll plugins on hosts. Reads the node banner and returns the node hostname, Munin version, and configured plugin list summary.
Port: 4949
monitoring
`mysql` — MySQL / MariaDB
TCP
MySQL / MariaDB is the binary client-server wire protocol used by MySQL, MariaDB, and Percona Server database engines. Reads the server-greeting packet and (when credentials are configured) authenticates, returning the server version, capability flags, supported authentication plugins, and TLS availability.
Ports: 3306, 33060
databasetls
`mysqlx` — MySQL X Protocol
TCP
MySQL X Protocol is a Protocol-Buffer-based wire protocol exposed by MySQL Server (default TCP/33060) for MySQL Shell and X DevAPI document-store and CRUD clients. Sends a CapabilitiesGet message and returns the supported X-Protocol capabilities and TLS requirements.
Port: 33060
database
`natpmp` — NAT-PMP
UDP
NAT-PMP is a protocol used by clients to request port forwards from a NAT gateway (RFC 6886). Sends a public-address request and returns the gateway's external IPv4 address, response code, and seconds-since-epoch counter.
Port: 5351
lightmgmt
`nats`
TCP
NATS is a lightweight publish/subscribe and request/reply messaging system. Reads the NATS INFO message and returns the server identifier, version, host, port, and authorization/TLS requirements.
Ports: 4222, 6222, 8222
clearmessagingtls
`ndmp`
TCP
NDMP is the Network Data Management Protocol used by enterprise backup software to coordinate backups of NAS devices. Opens a CONNECT_OPEN session and returns the NDMP protocol version reported by the server along with the connection status and any reason text.
Port: 10000
backupstorage
`neo4j` — Neo4j Bolt
TCP
Neo4j Bolt is the graph-database wire protocol used by Neo4j drivers and clients. Performs the Bolt handshake and returns the negotiated Bolt version and server release.
Ports: 7473, 7474, 7687
databasetls
`netbios` — NetBIOS Session Service
TCP

NetBIOS Session Service is the TCP/139 transport used by the legacy NetBIOS-over-TCP framing of SMB. runZero records the session-service banner alongside the NetBIOS name and workstation/server resource records reported by the host. The application-layer SMB protocol is decoded separately.
Port: 137
clearmgmt
`netbios-dgm` — NetBIOS Datagram Service
UDP
NetBIOS Datagram Service is the UDP/138 broadcast/datagram side of NetBIOS-over-TCP. runZero observes datagrams to recover the NetBIOS computer and workgroup names announced by Windows hosts and SMB servers.
Port: 138
discovery
`netbios-ns` — NetBIOS Name Service
UDP
NetBIOS Name Service is a protocol used for legacy Windows name registration and resolution. Sends a NetBIOS node-status query and returns the registered names, node type, and any associated MAC address.
Ports: 137, 138
naming
`netis`
UDP
Netis is an identifier for Netis and Netcore SOHO routers, including the well-known UDP/53413 administrative backdoor exposed by historical firmware. runZero attributes the service from the netis-specific hint and tags the asset as a Netis router so the historical backdoor exposure is surfaced in inventory and reports.
iotmgmt
`netop-remote-control` — Netop Remote Control
TCP
Netop Remote Control is a commercial remote-administration product from Netop (formerly Danware), commonly deployed in classroom, kiosk, and retail environments. runZero attributes the asset from the Netop host banner observed on the standard service port and tags the service as remote-access for inventory and exposure reporting.
clearlightremote-access
`nfs`
TCPUDP
NFS is the Network File System used to share files across Unix-like systems (Sun RPC program 100003). Issues a NFS NULL ping and a MOUNT EXPORT call (via portmap) and returns the supported NFS versions and the list of exported filesystems and allowed clients.
Port: 2049
filestorage
`nrpe` — Nagios NRPE
TCP
Nagios NRPE is the protocol used by Nagios/Icinga to run checks on remote hosts. Sends a _NRPE_CHECK probe and returns the NRPE protocol version and any version banner disclosed.
Port: 5666
monitoringtls
`ntp`
UDP

NTP is the Network Time Protocol used to synchronize clocks across networks (RFC 5905). Sends mode-3 client and mode-6 control queries and returns the stratum, reference identifier, refid clock source, and (when enabled) implementation/version strings disclosed by mode-6 readvar.
Port: 123
time
`omronfins` — Omron FINS
UDP
Omron FINS is the Factory Interface Network Service protocol used to communicate with Omron CJ/CS/NJ/NX PLCs and related automation devices. Issues a Controller Data Read (0501) and returns the controller model and firmware version along with the FINS handshake banner.
Port: 9600
ot
`opcda` — OPC Classic (OPC DA)
TCP
OPC Classic (OPC DA) is the OLE for Process Control data-access standard layered over Microsoft DCOM, widely deployed in legacy SCADA, HMI, and historian gateways. runZero detects OPC DA servers when the DCERPC scanner observes the OPCEnum interface UUID advertised by the endpoint mapper.
deepot
`opcua` — OPC UA
TCP
OPC UA is a vendor-neutral industrial information-model and data-access standard. Performs a GetEndpoints request and returns the application URI, product URI, server-certificate metadata, and per-endpoint security policies and identity tokens.
Ports: 4840, 4843, 48050
backplanedeepot
`openvpn`
UDP
OpenVPN is the tunnel control-channel protocol used by OpenVPN Community Edition and OpenVPN Access Server VPN gateways. Sends a P_CONTROL_HARD_RESET_CLIENT_V2 packet and returns the local and remote OpenVPN session identifiers from the server's hard-reset reply.
Port: 1194
encryptedvpn
`oracle` — Oracle TNS
TCP
Oracle TNS is the Transparent Network Substrate listener protocol used by Oracle Database servers. Sends a TNS Connect carrying a VERSION command and returns the listener version, instance name, and supported services.
Ports: 1521, 1522, 1525, 2483, 2484
database
`oracledb` — Oracle Database (TNS)
TCP
Oracle Database (TNS) is a lightweight Oracle Net listener probe used to identify Oracle Database servers without negotiating a session. Sends a minimal TNS Connect, parses the Refuse/Accept/Resend reply, and returns the packet type, VSNNUM, error code, and disclosed listener version.
Ports: 1521, 1522, 1525

database
`orion` — SolarWinds Orion Platform
TCP
SolarWinds Orion Platform is a Windows-based network-management suite that bundles NPM, NCM, NTA, IPAM, SAM, UDT, and related modules. The HTTP extractor matches the Orion footer in the web console, parses the platform release and component list, and returns orion.version, orion.components, and SolarWinds Orion software identification.
lighttlsweb
`panasonictv` — Panasonic TV
TCP
Panasonic TV is the network control service exposed by Panasonic Viera and similar consumer smart TVs for remote-control companion apps. runZero identifies these televisions from the proprietary control-protocol banner returned on connection, applies Panasonic TV fingerprinting, and attributes the asset as a Panasonic television.
iotlight
`panxmlapi` — Palo Alto Networks PAN-OS XML API
TCP
Palo Alto Networks PAN-OS XML API is the authenticated management API (system info, ARP/MAC/neighbor caches, interfaces) issued against Palo Alto Networks firewalls and Panorama. Used by the runZero scanner with a user-supplied API key to enumerate adjacent assets and device facts.
authdeeptls
`pca` — Symantec pcAnywhere
UDP
Symantec pcAnywhere is a remote-access protocol. Sends the pcAnywhere status probe and returns the host name, status, and capability flags disclosed in the response.
Port: 5632
lightremote-access
`pcworx` — Phoenix Contact PCWorx
TCP
Phoenix Contact PCWorx is a runtime protocol used to program and interact with ILC-series and other Phoenix Contact controllers. Queries controller identification and returns the PLC type, model number, and firmware version, date, and time.
Port: 1962
ot
`pega` — Pega Platform
TCP
Pega Platform is a low-code business process management and CRM application server from Pegasystems used for enterprise workflow automation. The HTTP extractor matches Pega in the page title or body, parses the version span, and returns the pega.version attribute and Pegasystems Pega software identification.
lighttlsweb
`pfcp`
UDP
PFCP is the Packet Forwarding Control Protocol used in 5G/LTE mobile cores between control-plane and user-plane functions. Sends a PFCP Heartbeat Request and returns the recovery time stamp and supported feature flags.
Port: 8805
mobile-core
`pop3`

TCP
POP3 is the Post Office Protocol version 3 used by mail clients to retrieve messages from a server. Reads the POP3 greeting and runs CAPA, returning the server software banner, supported capabilities, and STARTTLS availability.
Ports: 110, 995
emailtls
`postgresql`
TCP
PostgreSQL is the frontend/backend wire protocol used by PostgreSQL database servers and compatible engines such as Amazon RDS/Aurora and CockroachDB. Performs an SSLRequest followed by a StartupMessage and returns the server version, supported authentication mechanisms, advertised server parameters, and TLS availability.
Ports: 5432, 5433, 6432
databasetls
`powerlink` — Ethernet POWERLINK
UDP
Ethernet POWERLINK is a real-time industrial Ethernet protocol from the EPSG, used for deterministic motion control and I/O between managing nodes and controlled nodes on machine tools, packaging lines, and robotics cells. Passively decodes POWERLINK frames and returns the node identifier, vendor identifier, product code, and revision.
otpassive
`pptp`
TCP
PPTP is the Microsoft Point-to-Point Tunneling Protocol legacy VPN. Sends a Start-Control-Connection-Request and returns the protocol version, vendor, firmware revision, and host name.
Port: 1723
vpn
`printerid` — Printer Identification
TCP
Printer Identification is a vendor-specific service exposed on TCP/9200 by HP, Lexmark, and other network printers and MFPs. runZero parses the model string returned by the device, records it as printerid.model, and uses the value as a synthetic fingerprint to identify the printer make and model during asset categorization.
lightprinting
`proconos` — Phoenix Contact ProConOS
TCP
Phoenix Contact ProConOS is the PLC runtime protocol from KW-Software/Phoenix Contact, used by ILC, RFC, and OEM-rebadged controllers running the ProConOS or ProConOS eCLR runtime. Issues a runtime identification query and returns the ladder-logic runtime version, PLC type, project name, boot project, and project source-code identifier.
Port: 20547
ot
`profinet`
UDP
PROFINET is an industrial Ethernet protocol used for cyclic and acyclic real-time communication with PROFINET I/O devices and PLCs. Performs a Read Identification request and returns the device vendor, order number, serial, software/hardware revision, and (when enabled) discovered slot/subslot module list.
Ports: 34962, 34963, 34964

backplanedeepot
`profinet-dcp` — PROFINET DCP
UDP
PROFINET DCP is the Discovery and basic Configuration Protocol used at Layer 2 to identify and assign names/IPs to PROFINET stations. Passively decodes DCP Identify announcements and returns the station name, vendor and device identifiers, and device role.
discoveryotpassive
`prosoft` — ProSoft Discovery Service
TCP
ProSoft Discovery Service is the UDP discovery protocol used by ProSoft Technology industrial gateways and radios. runZero parses the response to recover the device model, firmware revision, MAC address, and IP configuration.
Port: 1718
discoveryot
`psdisco` — PlayStation Discovery
UDP
PlayStation Discovery is the Sony PlayStation 4/5 console-discovery service (HTTP/1.1-style SRCH messages on UDP/987 and UDP/9302) used by Remote Play and second-screen companion apps. Sends a SRCH probe and decodes the response, returning the host id, host name, host type, system version, discovery-protocol version, and running-app title metadata.
Ports: 987, 9302
discoverygamingiotlight
`qdrant` — Qdrant Vector Database
TCP
Qdrant is an open-source vector search database used by retrieval-augmented LLM applications. runZero attributes services as Qdrant from Recog matches against the HTTP API root and the /metrics endpoint, which expose the server version.
Ports: 6333, 6334
aiclearmgmt
`qotd` — Quote of the Day
TCPUDP
Quote of the Day is the RFC 865 diagnostic service historically exposed by Unix inetd hosts and frequently abused for UDP amplification. Passively decodes QOTD replies in TCP and UDP captures and returns the protocol tag along with the truncated quote text or banner observed in the response.
Port: 17
legacylight
`qualys` — Qualys Cloud Agent / Scanner Appliance
TCP
Qualys Cloud Agent / Scanner Appliance is the web management interface exposed by Qualys vulnerability-management components, including the on-premises Scanner Appliance and Cloud Agent UI. The HTTP fingerprinter inspects these pages and returns the product name, appliance role, and any version identifiers disclosed.
lightsecuritytlsweb
`quic`
UDP
QUIC is the IETF transport (RFC 9000), an encrypted UDP-based transport that carries HTTP/3 and is increasingly used by CDNs, web servers, and SaaS endpoints. runZero attributes the service as QUIC when the long-header Initial packet is observed on a probed UDP port and tags the asset for web / TLS exposure tracking.

encryptedtlsweb
`radius`
UDP
RADIUS is an AAA protocol used in Wi-Fi, VPN, and network-access control. Sends an Access-Request with an invalid principal and returns the response code, any Reply-Message and NAS-Identifier, and the list of attributes present in the reply.
Ports: 1645, 1646, 1812, 1813
aaa
`raritan-csc` — Raritan CommandCenter
TCP
Raritan CommandCenter is the Common Socket Connection (CSC) management protocol used by Raritan CommandCenter Secure Gateway appliances and adjacent Raritan KVM/serial console managers. Passively detects the <CSC/> banner emitted on connection and tags the asset as a Raritan CommandCenter device.
lightmgmt
`rdp`
TCP
RDP is the Microsoft Remote Desktop Protocol used by Windows Remote Desktop Services. Performs an X.224 Connection-Request and returns the supported security protocols, NLA requirement, and (when TLS is offered) certificate-derived hostname/version metadata.
Ports: 3389, 3390
remote-accesstls
`redfish`
TCP
`redis`
TCP
Redis is an in-memory data-structure store and message broker. Issues PING/INFO/AUTH probes and returns the Redis version, role, mode, and authentication or protected-mode requirements.
Ports: 6379, 16379, 26379
databasetls
`rexec`
TCP
rexec is the BSD Remote Execution protocol (legacy, transmits credentials in cleartext). Detects an rexec listener and returns the responsiveness and any host-identification banner observed.
Port: 512
clearlegacylightremote-access
`riak` — Riak Protocol Buffers
TCP
Riak is a distributed NoSQL key/value store from Basho. This entry covers the Protocol Buffers transport on TCP/8087 used by the native Riak clients; the HTTP API is reported separately as riak-http.
Port: 8098
clearmgmt
`riak-http` — Riak HTTP API
TCP
Riak HTTP API is the REST interface exposed by Basho Riak nodes (typically TCP/8098). runZero queries /stats and /riak/ to recover the node name, ring size, and Riak version.
Port: 8098

clearmgmt
`rip`
UDP
RIP is a distance-vector IGP. Sends a RIP Request and returns the RIP version and any advertised routes disclosed by the responder.
Port: 520
routing
`rlogin`
TCP
rlogin is the legacy BSD remote-login protocol (RFC 1282) superseded by SSH. runZero records the banner and any login prompt returned by the server, and flags the service as a clear-text credential exposure.
Port: 513
clearmgmt
`roomalert` — AVTECH Room Alert
TCP
AVTECH Room Alert is an environmental monitoring appliance. Reads the device banner and returns the model, OS version, MAC address, and IP address.
Port: 9999
iotlightmonitoring
`rpcbind` — ONC RPC / rpcbind
TCPUDP
ONC RPC / rpcbind is the portmap service used to discover Sun RPC programs (NFS, NIS, ...). Queries the portmap dump and returns the registered program list with versions, protocols, and ports.
Port: 111
discoverymgmt
`rsync` — rsync (SSH-tunneled)
TCP
rsync is the rsync daemon wire protocol exposed on TCP/873 by file mirrors, backup servers, and software-distribution archives. Reads the server greeting, performs the version handshake, and returns the protocol version, raw banner, and module list when the server permits enumeration.
Port: 873
file
`rsyncd` — rsync daemon
TCP
rsync daemon is the standalone rsync service (rsync://) listening on TCP/873. Reads the daemon greeting and lists modules, returning the rsync version, available module names, and module comments.
Port: 873
clearfilelight
`rtmp`
TCP
RTMP is the Adobe Real-Time Messaging Protocol used to stream audio, video, and data between Flash players and media servers. Performs the RTMP handshake and returns the protocol version byte echoed by the server.
Port: 1935
media
`rtps` — OMG RTPS / DDS

UDP
OMG RTPS / DDS is the Real-Time Publish-Subscribe wire protocol from the Object Management Group; underlies DDS in robotics, ROS 2, and industrial IoT. Sends an SPDP participant announcement and returns the participant GUID, vendor identifier and name, and protocol version.
Ports: 7400, 7401, 7410, 7411
discoverymessagingot
`rtsp`
TCP
RTSP is a streaming-control protocol used by IP cameras, NVRs, and media servers to control streams. Issues an OPTIONS request and returns the server software, supported methods, and any session-description metadata disclosed by DESCRIBE.
Ports: 554, 8554
iotmediatls
`s7comm` — Siemens S7Comm
TCP
Siemens S7Comm is the protocol used to program and exchange data with SIMATIC S7-300, S7-400, S7-1200, and S7-1500 PLCs. Issues SZL identification reads and returns the module name, plant identification, copyright, serial, module type, hardware/firmware version, and (when enabled) backplane rack/slot module list.
Port: 102
backplanedeepot
`sadp` — Hikvision SADP
UDP
Hikvision SADP is the Search Active Devices Protocol used to discover Hikvision and OEM IP cameras and NVRs. Sends a SADP Inquiry and returns the device serial, model, firmware, MAC, IP configuration, and activation state.
Port: 37020
discoveryiotlightmulticast
`sccp` — Cisco SCCP / Skinny
TCP
Cisco SCCP / Skinny is a call-control protocol used by Cisco IP phones registering with CallManager/Unified Communications Manager. Sends a Register message and returns the call-manager response and station identification.
Ports: 2000, 2443
voip
`sctptun` — SCTP Tunnel
UDP
SCTP Tunnel is runZero's identifier for SCTP-over-UDP encapsulation (RFC 6951) used to traverse middleboxes that block native SCTP. It is reported when an SCTP INIT is observed inside a UDP/9899 datagram.
Port: 9899
clearmgmt
`securemote` — Check Point SecuRemote
TCP
Check Point SecuRemote is a Check Point VPN topology discovery service. Sends the topology query and returns the gateway hostname and server identifier disclosed in the response.
Port: 264
securityvpn

`sentinel` — Redis Sentinel
TCP
Redis Sentinel is the high-availability supervisor for Redis. runZero issues SENTINEL ping and SENTINEL master to recover the sentinel version, monitored master name, and quorum configuration on unauthenticated instances.
Port: 26379
clearmgmt
`sercos-iii` — SERCOS III
UDP
SERCOS III is a real-time industrial Ethernet protocol used for drives, servos, and I/O in machine tools and packaging machinery. Passively decodes SERCOS III frames and returns the slave count, cycle time, and vendor and device codes.
otpassive
`servicetag` — Sun Service Tag
TCPUDP
Sun Service Tag is a discovery service used to inventory Sun/Oracle hardware and software. Sends the discovery probe and returns the registered product instance, instance URN, and version.
Port: 6481
discoverymgmt
`sgsap` — SGsAP (SCTP)
SCTP
SGsAP (SCTP) is the SGs interface (3GPP TS 29.118) over SCTP between MME and MSC for SMS over SGs and CSFB. Verifies the SCTP association and SGsAP payload protocol identifier and returns endpoint identification.
Port: 29118
mobile-core
`sip`
TCPUDP
SIP is a signaling protocol used to establish voice, video, and messaging sessions. Sends an OPTIONS request and returns the response code, server/user-agent strings, allowed methods, and any contact and via metadata disclosed.
Ports: 5060, 5061
tlsvoip
`sixnet-udr` — Sixnet UDR
TCPUDP
Sixnet UDR (Universal Data Register) is the station control protocol served by sxether_client on 1594/tcp and 1594/udp on Red Lion / Sixnet SixTRAK and VersaTRAK RTUs. runZero identifies it passively - a reply that leads with the 0x29 sync byte and carries a valid CCITT CRC - without ever sending a UDR frame, because the channel is unauthenticated by default and can carry destructive file and I/O-register writes. A reachable listener is flagged as an exposure.
Port: 1594
discoverylightot
`slp`
TCPUDP
SLP is the Service Location Protocol used to discover services on a LAN (RFC 2608); commonly exposed by VMware ESXi and printers. Sends an attribute and service-type request and returns the SLP version, advertised services, and per-service attribute summary.
Port: 427

discovery
`smb` — SMB / CIFS
TCP
SMB / CIFS is the file-sharing and IPC protocol used by Windows and Samba. Negotiates SMB1/2/3 and reads tree/share metadata, returning the dialect, signing/encryption requirements, server OS, NetBIOS/computer name, domain, and (when permitted) the list of shares.
Ports: 139, 445
filestorage
`smb1` — SMBv1
TCP
SMBv1 is the legacy Server Message Block dialect (CIFS / NT LM 0.12) deprecated by Microsoft and disabled by default since Windows 10 1709 / Server 2019. runZero flags this dialect specifically because it is required by the EternalBlue and related exploits and should be disabled wherever possible.
Ports: 139, 445
clearmgmt
`smb2` — SMBv2
TCP
SMBv2 is the Server Message Block dialect family introduced in Windows Vista / Server 2008 (dialects 2.0.2 and 2.1). runZero records the negotiated dialect, signing requirements, server GUID, and operating-system version reported during NEGOTIATE_PROTOCOL and SESSION_SETUP.
Port: 445
clearmgmt
`smb3` — SMBv3
TCP
SMBv3 is the Server Message Block protocol family introduced in Windows 8 / Server 2012 (dialects 3.0, 3.0.2, 3.1.1). runZero records the negotiated dialect, signing and encryption capabilities, and any pre-auth integrity hash advertised by the server.
Port: 445
clearmgmt
`smtp`
TCP
`snmp`
UDP
SNMP is the Simple Network Management Protocol used to monitor and configure network devices and servers. Walks system.* and selected enterprise OIDs over SNMPv1/v2c (and SNMPv3 when configured) and returns sysDescr, sysObjectID, sysName, location, contact, and a vendor/device-type fingerprint derived from the response.
Ports: 161, 162, 10161, 10162
authmgmt
`socks` — SOCKS Proxy
TCP
SOCKS is a generic proxy protocol with two incompatible versions (SOCKS4 and SOCKS5). This entry captures sightings where only the SOCKS family was identified; the version-specific decoders socks4 and socks5 record the negotiated authentication methods and supported commands.
Port: 1080
clearmgmt

`socks4`
TCP
SOCKS4 is a proxy protocol (SOCKS version 4) used by client applications and proxy servers (Squid, Dante, SSH dynamic forwarding) to relay TCP connections through an intermediary. Sends a CONNECT request to a benign target and returns the proxy responsiveness and the SOCKS reply status code observed.
Ports: 1080, 1081
clearlightproxy
`socks5`
TCP
SOCKS5 is a proxy protocol (SOCKS version 5, RFC 1928) used by client applications, Tor (9050/9150), and proxy servers such as Dante, 3proxy, and Squid to relay TCP and UDP through an intermediary. Sends a method-selection request and returns the SOCKS version, supported authentication methods, and proxy reachability.
Ports: 1080, 1081, 9050, 9150
lightproxy
`solr` — Apache Solr
TCP
Apache Solr is the Lucene-based enterprise search platform. runZero attributes services as Solr from the X-Solr-Version response header, the admin-UI banners, and the /solr/admin/info/system JSON, which exposes the Solr and Lucene versions, JVM details, and host operating system.
Port: 8983
clearmgmt
`some-ip` — AUTOSAR SOME/IP
UDP
AUTOSAR SOME/IP is the Scalable service-Oriented MiddlewarE over IP used between automotive ECUs on in-vehicle Ethernet for service discovery and RPC. Sends a SOME/IP-SD FindService and returns the advertised service IDs and service count.
Ports: 30490, 30491
automotiveot
`sonarqube`
TCP
SonarQube is the static analysis and code-quality platform from SonarSource. runZero attributes services as SonarQube from the application's HTTP banners and the /api/system/status endpoint, recovering the server version and edition.
Port: 9000
clearmgmt
`sonicwall-sgms` — SonicWall GMS Agent
TCP
SonicWall GMS Agent is the Global Management System agent used to manage SonicWall firewalls. Identifies the agent from its TCP banner.
Port: 3023
lightmgmtsecurity
`splunk` — Splunk Enterprise / Universal Forwarder Web UI
TCP
Splunk Enterprise / Universal Forwarder Web UI is the HTTP-served management interface for Splunk's SIEM and log-collection platform. The HTTP extractor inspects splunkd and Splunk Web responses and returns the product edition and version.
lightsiemtlsweb

`spotify-connect` — Spotify Connect
TCP
Spotify Connect is the device-discovery and remote-control protocol used by Spotify clients to find playback endpoints (smart speakers, AV receivers, set-top boxes). Detected via the _spotify-connect._tcp mDNS record and the /zc HTTP endpoint exposed by the device.
discoveryiot
`ssdp`
UDP
SSDP is the Simple Service Discovery Protocol used by UPnP devices to advertise services. Sends an M-SEARCH and returns the advertised service types, USN, server string, and Location URLs of the responding devices.
Port: 1900
discoveryiotlightmulticast
`ssh`
TCP
SSH is the Secure Shell remote-access and tunneling protocol. Reads the SSH banner, runs a KEX-init exchange, and (when credentials are configured) authenticates, returning the server software string, supported KEX/host-key/cipher/MAC algorithms, host keys and fingerprints, and accepted authentication methods.
Ports: 22, 2222, 22222
encryptedremote-access
`steam` — Steam Server Discovery
UDP
Steam Server Discovery is the Valve Steam Remote Play / In-Home Streaming LAN broadcast protocol. Sends a CMsgRemoteClientBroadcastDiscovery and returns the hostname, client/instance/device IDs, client version, OS type, public IP, and Steam Deck / VR / Remote Play status.
Port: 27036
discoverygaminglight
`stun`
UDP
STUN is the Session Traversal Utilities for NAT protocol (RFC 5389) used for NAT discovery in WebRTC and VoIP. Sends a binding request and returns the SOFTWARE attribute and the observed XOR-MAPPED-ADDRESS reported by the server.
Ports: 3478, 3479, 5349, 5350
voip
`sua` — SUA (SCTP)
SCTP
SUA (SCTP) is the SCCP User Adaptation Layer (RFC 3868) over SCTP, used to carry SS7 SCCP signaling over IP. Verifies the SCTP association and SUA payload protocol identifier and returns endpoint identification.
Ports: 2904, 14001
mobile-corevoip
`subversion` — Apache Subversion
TCP
Subversion is the Apache version-control system. This entry covers detections produced by integration data and Recog banners; runZero's active svn decoder negotiates the svnserve protocol on TCP/3690 to recover the repository UUID, root URL, and supported capabilities.
Port: 3690
clearmgmt

`sunrpc` — Sun RPC Portmapper
UDP
Sun RPC Portmapper (also known as rpcbind) is the legacy ONC RPC service registry. This entry covers Recog-only matches against banners that did not return a structured rpcbind dump; the active rpcbind decoder enumerates registered programs and their dynamic ports.
Port: 111
clearmgmt
`svn` — Subversion
TCP
Subversion is the Apache Subversion svn:// version-control protocol. Reads the SVN greeting and returns the minimum and maximum supported protocol versions, supported capabilities, and offered authentication mechanisms.
Port: 3690
file
`sybase` — Sybase / SAP ASE (TDS 5.0)
TCP
Sybase / SAP ASE (TDS 5.0) is the Tabular Data Stream 5.0 wire protocol used by Sybase ASE and SAP Adaptive Server Enterprise. Sends a TDS prelogin probe and returns the server version reported in the prelogin response.
Port: 5000
database
`syslog`
TCPUDP
Syslog is a standard event-logging protocol used to forward log messages between hosts and collectors. Identifies syslog listeners over UDP/TCP and decodes RFC 3164/5424 messages to report the priority, facility, severity, version, hostname, and application name.
Ports: 514, 6514
monitoringtls
`tcpmux`
TCP
TCPMUX is the TCP Port Service Multiplexer (RFC 1078), a legacy diagnostic service. Queries the registered service list and returns the disclosed names.
legacylight
`teamviewer`
TCP
TeamViewer is the proprietary binary protocol used by the TeamViewer remote-access client to reach the TeamViewer cloud over TCP/5938. Sends a TeamViewer ping/hello probe and identifies the service from the response magic and command byte.
Port: 5938
lightremote-access
`telnet`
TCP
`tenable-agent-id` — Tenable Agent Identifier
TCP
Tenable Agent Identifier is the synthetic protocol runZero uses to track the unique agent UUID reported by Tenable Nessus and Tenable.io agents observed via integrations. It is not a network protocol and has no associated wire-level scan.
integration
`tftp`
UDP

TFTP is the Trivial File Transfer Protocol (RFC 1350) used for boot images, firmware, and config transfer. Sends a benign read request and returns the responsiveness and any error-code metadata that discloses the server implementation.
Port: 69
file
`thinprint`
TCP
ThinPrint is the Cortado virtual-printing protocol used by Citrix, VMware Horizon, and Microsoft RDS deployments to redirect print jobs from session hosts to client-side printers. Detected from the TPAutoConnect listener banner on TCP/4000.
Port: 4000
clearmgmt
`time`
TCPUDP
Time is the legacy 32-bit time-of-day service (RFC 868). Reads the response to confirm the service and detect amplification-capable hosts.
Port: 37
legacylighttime
`tls` — TLS / SSL
TCP
TLS / SSL is the Transport Layer Security encrypted-transport substrate used by HTTPS and most modern Internet protocols. Performs a TLS handshake and returns the negotiated version and cipher suite, supported versions and extensions, and the full server-certificate chain with subject, issuer, SANs, validity, and key metadata.
Ports: 443, 5986, 6443, 8443, 9443
encrypted
`turn`
UDP
TURN is the Traversal Using Relays around NAT protocol (RFC 5766) used as a media relay for WebRTC and VoIP. Sends an Allocate request and returns the SOFTWARE attribute, mapped and relayed addresses, allocation lifetime, requested transport, and any error code.
Ports: 3478, 3479, 5349, 5350
voip
`ubnt` — Ubiquiti Discovery
UDP
Ubiquiti Discovery is a device-discovery protocol used by UISP/UNMS and the Ubiquiti Discovery Tool. Sends the discovery probe and returns the device hostname, model, firmware, MAC, and IP configuration.
Port: 10001
discoverylightmgmt
`unitronics` — Unitronics PCOM
TCP
Unitronics PCOM is the proprietary protocol used to communicate with Unitronics Vision and Samba/UniStream PLC+HMI controllers. Queries the controller identification and returns the model and OS version.
Port: 20256
ot
`upnp` — Universal Plug and Play
TCP

UPnP is the device-control protocol layered on top of SSDP. runZero fetches the device-description XML referenced in the SSDP LOCATION header to recover the friendly name, manufacturer, model, serial number, UPnP UUID, and the list of advertised services.
cleardiscoveryiot
`uscan` — Apple Image Capture (uscan)
TCP
uscan is the AirScan/Mopria HTTP scanner protocol used by macOS Image Capture, iOS Notes, and Mopria-compatible clients to discover and drive network scanners. Detected via the _uscan._tcp mDNS record and the /eSCL/ScannerCapabilities XML returned by the device.
cleardiscoveryiot
`uscans` — Apple Image Capture over HTTPS (uscans)
TCP
uscans is the TLS-protected variant of the AirScan/Mopria scanner discovery protocol, advertised via the _uscans._tcp mDNS record. runZero fetches the HTTPS /eSCL/ScannerCapabilities document to recover the device make, model, firmware, and supported scan profiles.
discoveryiot
`vault` — HashiCorp Vault
TCP
Vault is HashiCorp's secrets-management platform. runZero queries /v1/sys/health and /v1/sys/seal-status on the HTTP API to recover the Vault version, cluster name, cluster ID, sealed/initialized state, and replication mode.
Port: 8200
clearmgmt
`vertiv_liebert`
TCP
`vmauthd` — VMware vmauthd
TCP
VMware vmauthd is the authentication daemon listening on VMware ESXi and Workstation hosts. Identifies the service from its 220 greeting banner.
Port: 902
lightmgmt
`vmware`
TCPTLS
`vnc` — VNC / RFB
TCP
VNC / RFB is the Virtual Network Computing remote-desktop protocol, also known as the Remote Frame Buffer protocol. Reads the protocol-version handshake and returns the RFB version, supported security types, and any disclosed vendor banner.
Ports: 5800, 5900, 5901, 5902, 5903
remote-accesstls
`vsdp` — Vivotek Search Discovery Protocol
UDP
VSDP is the UDP broadcast discovery protocol used by Vivotek IP cameras and video servers. runZero parses the response to recover the camera model, firmware version, MAC address, and IP configuration.
Port: 3702
discoveryiot
`vxlan` — Virtual eXtensible LAN
UDP

VXLAN is the L2-over-UDP tunneling protocol (RFC 7348) used by data-center overlay networks. runZero observes VXLAN encapsulation on UDP/4789 and records the VXLAN Network Identifier (VNI) of the inner Ethernet frame.
Port: 4789
clear
`waveu` — Wave/UE Discovery
UDP
WaveU is the UDP discovery service used by Crestron AirMedia and selected Wave-branded conferencing endpoints to advertise their presence to companion mobile and desktop apps. runZero parses the response to recover the model and firmware revision.
discoveryiot
`wbsm` — IBM Web-Based System Manager
TCP
IBM Web-Based System Manager (WSM/WebSM) is the remote-administration protocol used by the Java WSM client to manage AIX systems. Identifies the service from its banner.
Port: 9090
lightmgmt
`wdbrpc` — VxWorks WDB Agent
UDP
WDB RPC is the Wind River Workbench Debug Bridge agent built into many VxWorks images and exposed (often unintentionally) on UDP/17185. runZero issues a target-info query to recover the VxWorks version, BSP name, CPU type, and target name; presence of WDB indicates the host can be remotely controlled without authentication.
Port: 17185
clearmgmtot
`webmin`
UDP
Webmin is a web-based Unix administration suite that exposes a UDP discovery service on UDP/10000 alongside Usermin and Virtualmin. Sends the "webmin" query and returns the advertised Webmin server IP, port, and HTTP/HTTPS scheme used to reach the management UI.
Port: 10000
lightmgmtweb
`wiznet` — WIZnet Discovery
UDP
WIZnet Discovery is a device-discovery protocol used by WIZnet serial-to-Ethernet modules and embedded TCP/IP chips. Sends the discovery probe and returns the raw advertised configuration fields.
Ports: 5000, 50001
discoveryiotlight
`wsd` — Web Services Dynamic Discovery
UDP
Web Services Dynamic Discovery (WS-Discovery) is the SOAP-over-UDP multicast discovery protocol used by Windows printers, scanners, and ONVIF IP cameras. runZero issues a Probe to ff02::c / 239.255.255.250 and parses the ProbeMatch response to recover the device's endpoint reference, types, scopes, and metadata URLs.
Port: 3702
cleardiscoveryiot
`wsman` — WS-Management
TCP

WS-Management (WS-Man) is the SOAP-over-HTTP/HTTPS management protocol used by Windows Remote Management (WinRM), iDRAC, iLO, and IPMI baseboard controllers. runZero issues an Identify request to recover the product vendor, product version, and supported protocol versions.
Ports: 5985, 5986
clearmgmt
`x11` — X11 / X Window System
TCP
X11 / X Window System is the X Window System display protocol used by Unix graphical desktops; if exposed, it allows remote display capture and input injection. Performs an X11 connection-setup probe and returns the X.Org/XFree86 vendor string, protocol-major version, and the access state of the server.
Ports: 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6007, 6008, 6009, 6010, 6011, 6012, 6013, 6014, 6015
clearlightremote-access
`x2ap` — X2AP (SCTP)
SCTP
X2AP (SCTP) is the X2 Application Protocol (3GPP TS 36.423) over SCTP used between LTE eNodeBs. Verifies the SCTP association and X2AP payload protocol identifier and returns endpoint identification.
Port: 36422
mobile-core
`xdmcp`
UDP
XDMCP is the X Display Manager Control Protocol used by X Window System login managers (xdm, gdm, kdm). Sends an XDMCP query and returns the responding manager's hostname and supported authentication types.
Port: 177
legacylightremote-access
`zabbix` — Zabbix Agent
TCP
Zabbix Agent is the monitoring agent used by the Zabbix server to collect host metrics. Sends an agent.version request and returns the agent version, derived CPE, and whether remote commands are enabled.
Ports: 10050, 10051
monitoringtls
`zabbix-agent` — Zabbix Agent
TCP
Zabbix Agent is the host-side collector polled by a Zabbix server over TCP/10050. runZero requests the agent.version and agent.hostname items to recover the agent build and configured hostname from unauthenticated agents.
Port: 10050
clearmgmt
`zebra` — Zebra Discovery
UDP
Zebra Discovery is a network discovery protocol used by Zebra Technologies label and barcode printers. Sends the discovery probe and returns the printer hostname disclosed in the response.
Port: 6101
discoverylightprinting

`zookeeper` — Apache ZooKeeper
TCP
Apache ZooKeeper is a distributed coordination service whose wire protocol exposes four-letter administrative commands. Sends a four-letter (ruok/srvr/conf) command and returns the access state, mode (leader/follower/standalone), node count, and ZooKeeper version when the command is permitted.
Ports: 2181, 2888, 3888
databasemgmt
`zyxel` — Zyxel Device Web Management
TCP
Zyxel Device Web Management is the HTTP management interface exposed by Zyxel switches and routers (e.g. GS1200 series), serving a system_data.js endpoint that advertises model, firmware version, MAC, hostname, and IP configuration.
lightnetworktlsweb

Additional protocols

- `afp`
- `anydesk`
- `banner`
- `citrixica`
- `h323`
- `iec60870-5-104`
- `imap`
- `ipp`
- `ipsec`
- `pulsar`
- `smpp`
- `snpp`
- `spice`
- `sstp`
- `wireguard`
- `xmpp`

Default ports

runZero scans a broad default port set, and you can configure any TCP or UDP port range.

What TCP ports does runZero scan?

runZero scans the following TCP ports by default:

7	13	17	21	22	37	53	80	88	102	110
111	113	135	137	139	179	264	389	427	443	445
464	502	512	513	514	515	523	535	541	554	593
631	636	749	750	771	789	873	902	995	1027	1080
1081	1099	1153	1200	1217	1433	1434	1521	1522	1525	1594
1718	1723	1883	1911	1935	1962	2000	2049	2121	2181	2222
2375	2376	2379	2404	2443	2455	2483	2484	2888	2947	3000
3023	3050	3080	3260	3268	3269	3306	3389	3390	3690	3868
3888	3999	4000	4001	4190	4222	4369	4545	4567	4712	4786
4840	4843	4911	4949	5000	5005	5006	5007	5037	5058	5060
5061	5094	5432	5433	5555	5666	5671	5672	5800	5900	5901
5902	5903	5938	5984	5985	5986	6000	6001	6002	6003	6004
6005	6006	6007	6008	6009	6010	6011	6012	6013	6014	6015
6222	6333	6334	6379	6432	6443	6481	6514	6556	6667	6668
6669	6697	7000	7001	7473	7474	7519	7676	7687	7734	7878
8000	8001	8002	8003	8004	8080	8081	8082	8086	8098	8161
8181	8193	8200	8222	8291	8443	8453	8500	8554	8728	8787
8788	8883	8888	8889	8983	9000	9001	9042	9050	9080	9090
9092	9093	9094	9100	9101	9102	9150	9160	9200	9418	9443
9595	9999	10000	10001	10050	10051	11211	11222	11740	13400	16379
18000	18245	19000	19530	20000	20256	20547	22222	25565	26379	27017
27018	27019	28017	33060	44818	48050	48898	54921	54922	54923	61616
61617	62078									

What UDP ports does runZero scan?

runZero scans the following UDP ports by default:

7	13	17	37	53	67	68	69	88	111	123
137	138	161	162	177	389	427	443	464	500	514
520	523	623	631	749	750	987	1194	1234	1235	1433
1434	1594	1604	1645	1646	1701	1812	1813	1900	2049	2123
2152	2228	2362	2427	2727	2944	2945	3250	3283	3386	3391
3478	3479	3671	3702	3956	4070	4433	4500	4569	4789	4800
5000	5058	5060	5061	5246	5349	5350	5351	5353	5355	5632
5683	5684	5783	6101	6481	6514	7400	7401	7410	7411	8611
8612	8805	9302	9600	9899	10000	10001	10161	10162	11211	12346
12366	12386	12406	12426	13400	17185	19132	20202	27036	30005	30490
30491	30718	34962	34963	34964	34980	37020	37810	41794	44818	46808

47808 47809 47810 47811 47812 47813 47814 47815 47816 47817 47818 47819
47820 47821 47822 47823 47824 48808 48898 50001

Scanning IoT and OT

Can I safely scan my IoT and OT environments?

Yes. IoT and OT equipment is often sensitive to high packet rates or malformed traffic, and past experiences with aggressive scanners have led many teams to put a “don’t scan” rule in place for these networks. runZero was built to operate safely in these environments, so most organizations can include their IoT and OT assets in their active inventory.

runZero discovers assets with a lightweight active scan engine called the Explorer. You can deploy the Explorer almost anywhere on your network, with no SPAN or TAP ports to configure and no agents to install on individual devices. Because discovery runs actively from a single point on the network, you don’t have to modify the environment you want to inventory.

What makes the Explorer safe for sensitive environments?

The Explorer was engineered with fragile OT and IoT systems in mind. It is not based on `nmap`, `masscan`, or any other open source or commercial scanner. It sends only well-formed, standards-compliant traffic: never a malformed packet, and never an attempt to exploit a vulnerability.

Safety controls built into every scan include:

- Scan rates are conservative. The defaults are `1000` packets per second across the entire scan and `40` packets per second per host. Both values are adjustable, and traffic is balanced evenly across all targets in scope.
- You can exclude critical assets, subnets, and individual TCP/UDP ports globally or per scan.
- Every UDP service probe, including each OT/IoT-specific probe, can be enabled or disabled individually.
- The Explorer paces connections: it balances SYN and ACK traffic and watches for port exhaustion on both the scanner and the target.
- A configurable max group size limits how many targets are probed concurrently, which directly controls the connection load on stateful middle boxes such as firewalls, proxies, and small routers.
- Only TCP and UDP ports that yield actionable fingerprinting data are probed, rather than all 65535. The list is fully adjustable for specialized equipment on non-standard ports (see Ports scanned by runZero).
- Probe behavior is tuned per protocol for known-fragile services. For example, the Explorer collects a banner from `9100/tcp` on a printer but never sends active probe data that could result in printed “garbage.”

For a step-by-step rollout plan, see the Scanning OT networks playbook.

Where should I deploy the Explorer?

Two patterns are common:

- Co-locate the Explorer with your existing vulnerability scanners. Those hosts typically already have allow-lists, network reachability, and session-table sizing that account for scan traffic, which simplifies deployment.
- Deploy dedicated Explorers at remote and OT sites. An Explorer on the same network segment as the assets it scans gives the best layer-2 visibility, removes middle-box concerns, and lets you tune scans independently per location.

For isolated OT networks with no internet access, deploy a self-hosted runZero console or use the offline scanner so Explorers never need to leave the OT zone.

OT/ICS vendors runZero fingerprints

runZero recognizes many OT, ICS, and IoT vendors out of the box, including:

- ABB
- Allen-Bradley / Rockwell Automation
- AutomationDirect
- Beckhoff
- BARIX
- Cisco
- CODESYS
- Control Solutions
- Control Techniques
- Danfoss
- Delta Electronics
- Digi
- Emerson
- Festo
- GE
- GENEREX
- GLC Controls
- Hilscher
- Hirschmann
- HMS Networks
- Honeywell
- ifm
- Johnson Controls
- Lantronix
- Linor Koda
- Mitsubishi Electric
- Moxa
- Murrelektronik
- Omron
- Phoenix Contact

- Pilz
- Pressac
- ProSoft Technology
- Red Lion
- Rittal
- Saia-Burgess
- Schneider Electric
- SICK
- Siemens
- Turck
- WAGO
- Weidmüller
- Westermo
- Yokogawa

IoT and OT protocols

runZero ships dedicated probes for the most common industrial, building-automation, and field-bus protocols. Each probe is standards-compliant and asks only for the information needed to fingerprint a device. You can toggle every probe individually in your scan templates.

Probe	Protocol	Typical port(s)	Notes
`BACNET`	BACnet/IP	`47808/udp`	HVAC, lighting, building management. Enumerates routed devices and BBMD/FDT entries as sub-assets.
`ETHERNETIP`	EtherNet/IP (CIP)	`44818/tcp+udp`, `2222/udp`	Rockwell / Allen-Bradley and other CIP devices. Enumerates backplane modules as sub-assets.
`MODBUS`	Modbus/TCP	`502/tcp`	Tune `modbus-identification-level` to match device capabilities. Enumerates connected unit IDs as sub-assets.
`S7COMM`	Siemens S7Comm	`102/tcp`	Set `s7comm-request-extended-information` to `true` for richer data. Enumerates rack/slot modules as sub-assets.
`KNXNET`	KNXnet/IP	`3671/udp`	Building automation; enumerates KNX devices behind the IP interface.
`DNP3`	DNP3	`20000/tcp+udp`	Some outstations only talk to one master at a time; test before broad rollout.
`PROFINET`	PROFINET	`34962-34964/udp`	Cyclic and acyclic real-time control; enumerates slot/subslot modules.
`ETHERCAT`	EtherCAT	`34980/udp`	Enumerates slaves on the EtherCAT segment.
`FINS`	Omron FINS	`9600/tcp+udp`	Omron PLCs and compatible controllers; enumerates connected units.

MELSECQ	Mitsubishi MELSEC-Q	5006-5007/tcp	Returns CPU model and type; enumerates backplane CPUs.
HARTIP	HART-IP	5094/tcp	Multiplexers in front of HART field instruments; walks Cmd 84 sub-device indices.
IEC104	IEC 60870-5-104	2404/tcp	SCADA telecontrol; enumerates outstation addresses.
MMS	IEC 61850 MMS	102/tcp	Substation IEDs; enumerates logical devices.
ADS	Beckhoff ADS	48898/tcp	Beckhoff TwinCAT runtime; enumerates ADS sub-devices.
C37118	IEEE C37.118 Synchrophasor	4712/tcp	PMUs reported by Phasor Data Concentrators.
ATG	Veeder-Root Automatic Tank Gauge	10001/tcp	Fuel-tank monitoring at retail / fleet sites.
BSAP-IP	Emerson BSAP/IP	1234-1235/udp	Emerson ControlWave / Bristol RTUs in oil & gas SCADA.
C12.22	ANSI C12.22	1153/tcp	Electric-utility AMI head-ends and relays.
CSPV4	Allen-Bradley CSPv4 / PCCC	2222/tcp	Legacy SLC 5/05 and MicroLogix PLCs.
DOIP	Diagnostics over IP	13400/tcp+udp	In-vehicle ECUs over Ethernet; enumerates reachable ECUs.
FOCAS	Fanuc FOCAS	8193/tcp	Fanuc CNC machine tools and robots.
HSMS	SEMI HSMS / SECS-GEM	5000/tcp	Semiconductor fab equipment (SEMI E37).
MBUS-TCP	M-Bus over TCP	8888-8889/tcp	Tunneled EN 13757 utility-meter aggregators.
OPCUA	OPC UA	4840/tcp, 4843/tcp	Vendor-neutral industrial information model and data access.

Fifteen of these probes (BACnet, CIP, Modbus, KNXnet, S7Comm, DNP3, PROFINET, EtherCAT, Omron FINS, MELSEC-Q, HART-IP, IEC 60870-5-104, IEC 61850 MMS, Beckhoff ADS, IEEE C37.118) also enumerate downstream devices behind the gateway and report each as its own sub-asset linked back to the gateway. See Protocol gateways for how that works and how to query the resulting sub-assets.

The protocol catalog has the complete list.

Start with the OT-specific probes disabled, validate a limited scan against a representative subset of devices, then enable additional protocols as your team gains confidence. The Scanning OT networks playbook lays out this phased rollout.

Platform integrations

Inbound integrations

Enriching runZero results with data from other tools

runZero integrates with several sources of asset data so you can enrich your asset inventory and find assets and subnets that aren't effectively managed or protected. Through product APIs and export/import functionality, runZero can pull data from many IT and security tools to extend visibility across your organization's network.

Supported integrations

Cloud and virtualization

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure
- Prisma
- VMware
- Wiz

Endpoint protection

- CrowdStrike Falcon
- Microsoft 365 Defender
- Microsoft Intune
- Miradore MDM
- SentinelOne
- Tanium API Gateway

Endpoint management

- Microsoft Endpoint Configuration Manager (MECM)

Asset and identity management

- Google Workspace
- Microsoft Active Directory
- Microsoft Entra ID (formerly Azure AD)

Vulnerabilities and risk

- Censys Search & Data
- Censys Platform
- Dragos
- Qualys VMDR
- Rapid7
  - InsightVM
  - Nexpose
- Shodan
- Tenable
  - Tenable Vulnerability Management
  - Tenable Nessus Professional
  - Tenable Security Center
  - Tenable Nessus (file import)

Network management

- Cisco Meraki Dashboard
- Tailscale

Custom integrations

If runZero has no integration for the tool you want to draw data from, Platform and Community users can add asset data from custom sources with the custom integrations feature. The primary path is custom integration scripts, which run on your Explorers or on the console and can pull from REST APIs or directly over SSH, SMB, WinRM, WMI, and SQL. A library of over a hundred ready-to-use scripts is available in the runzero-custom-integrations repository and from the example picker in the console's script editor. Where AI is enabled, the console can generate a script from a plain-language description. You can also import custom asset data through the API or the runZero Python SDK.

Each custom integration is available either to every organization in the account or only to the organizations granted access to it, and administrators can edit only the integrations that belong to organizations they administer. See organization access for details.

Scan probes or connector tasks

Most integrations can run either as a scan probe or as a connector task.

Scan probes run as part of a scan task, so one task can scan your environment and sync integrations at the same time. To run an integration as a scan probe:

1. Configure a scan task from the *Scan* menu in your inventory or tasks page.
2. Activate the integration under the *Probes* tab.
3. Activate the correct credential under the *Credentials* tab.

4. Configure, activate, or deactivate other scan task options as preferred.

Connector tasks run independently of scan tasks, so you can schedule integration syncs and asset scans separately. They run from the runZero cloud by default, or from an Explorer in your organization if you prefer. To run an integration as a connector task:

1. Configure a connector task from the Integrations page or the *Integrate* menu in your inventory or tasks page.
2. Select an Explorer from the *Explorer* menu (optional).
3. Configure, activate, or deactivate other connector task options as preferred.

Importing integration data

Some integrations can be used by importing data from the other platform into runZero. For example, runZero ingests `.nessus` files from Tenable Nessus and `.xml` files from Rapid7 Nexpose without connecting to their APIs.

Automatic asset merge

runZero maps each integration host to an asset like this:

- If the host matches an existing runZero asset, runZero updates the asset-level attributes and adds the integration-specific attributes.
- If the host matches no existing runZero asset, runZero creates a new asset in the site you chose when you set up the integration task.

runZero matches integration data to existing assets by the following, in priority order:

1. MAC address
2. IP address (3-day window)
3. Hostname

You can also merge integration assets into runZero assets by hand with the *Merge* button on the Asset Inventory page.

Removing an integration data source

When you remove an integration as a data source, runZero removes its attributes from your assets. Because some asset attribute fields are merged, attributes populated by both runZero scans and the integration may be deleted too. Rescanning the affected assets restores them.

Excluding integration attributes

You can configure your account and organizations to skip collecting certain integration attributes, for example personally identifiable information (PII) such as usernames or email addresses. Some attributes drive runZero's merging and fingerprinting, though, and excluding them may make your inventory less accurate.

Set excluded attributes in your Account settings under *Personally Identifiable Information (PII) collection* to apply them across all organizations, or in the settings of an individual Organization to override the account-level settings.

You can use the default list of attributes or define your own. Attributes take the format `@source.resourceType.attributeName`. For example, `@crowdstrike.dev.firstLoginUser` excludes the attribute "firstLoginUser" from CrowdStrike devices.

Only integrations that import PII support attribute exclusions. At this time those are:

- CrowdStrike
- SentinelOne
- Tanium
- Google Workspace
- Intune
- Miradore

The Directory Users and Directory Groups inventories don't support attribute exclusions. The AzureAD, LDAP, and Google Workspace integrations import directory users and groups by default, but you can turn that off in the task configuration.

Source names and IDs

This table maps each source name to the source ID you use when querying assets and vulnerabilities.

ID	Name	Description
-1	`custom`	Custom
1	`runzero`	runZero
2	`miradore`	Miradore
3	`aws`	AWS
4	`crowdstrike`	CrowdStrike
5	`azure`	Azure
6	`censys`	Censys
7	`vmware`	VMware
8	`gcp`	GCP
9	`sentinelone`	SentinelOne
10	`tenable`	Tenable
11	`nessus`	Nessus
12	`rapid7`	Rapid7

13	`insightvm`	InsightVM
14	`qualys`	Qualys
15	`shodan`	Shodan
16	`azuread`	AzureAD
17	`ldap`	LDAP
18	`ms365defender`	MS365Defender
19	`intune`	Intune
20	`googleworkspace`	GoogleWorkspace
21	`sample`	Sample
22	`tenablesecuritycenter`	TenableSecurityCenter
23	`packet`	Packet
24	`wiz`	Wiz
25	`meraki`	Meraki
26	`mecm`	MECM
27	`tanium`	Tanium
28	`simulator`	Simulator
29	`netbox`	NetBox
30	`cip`	CIP
31	`pan`	Palo Alto Networks
32	`prisma`	Prisma
34	`dragos`	Dragos
35	`bacnet`	BACnet
36	`modbus`	MODBUS
37	`knxnet`	KNXnet
38	`insightvmcloud`	InsightVMCloud
39	`s7comm`	S7comm
40	`dnp3`	DNP3
41	`profinet`	PROFINET
42	`ethercat`	EtherCAT
43	`omronfins`	OmronFINS
44	`melsecq`	MELSEC-Q

45	`hartip`	HART-IP
46	`c37118`	C37.118
47	`ads`	ADS
48	`opcua`	OPC-UA
49	`s7`	S7
50	`iec104`	IEC60870-5-104
51	`mms`	IEC61850-MMS
52	`censysplatform`	CensysPlatform
10001	`tailscale`	Tailscale
10002	`ninjaone`	NinjaOne

Amazon Web Services

runZero integrates with Amazon Web Services (AWS) to give you better visibility across your cloud environment. The integration imports data from each applicable API to add detail to your asset inventory:

- AWS EC2 API
- AWS RDS API
- AWS ELBv1 API
- AWS ELBv2 API
- AWS Lambda API
- AWS FSx API
- AWS Inspector2 API

With AWS synced, you can see at a glance how many EC2 instances, elastic load balancers, relational database services, FSx file systems, and VPCs you have running, along with their region, account, and more.

The integration imports all running EC2 instances, RDS instances, Lambda functions, FSx file systems, VPCs, and active application, network, gateway, or classic load balancers. You can connect it to a single AWS account or to all accounts in your organization, and it imports data across multiple regions.

Getting started

runZero supports these AWS resource types:

- EC2 instances
- Elastic load balancers
- RDS instances
- Lambda instances
- FSx file systems

- VPCs

To set up the AWS integration:

1. Configure AWS to allow API access for runZero.
2. Add the AWS credential to runZero, including the access key and secret key.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the AWS integration to sync your data with runZero.

Requirements

Before you set up the AWS integration:

- Make sure you have access to the AWS console.
- Use AWS Organizations if you want to connect to multiple accounts.
    - With AWS Organizations and a compatible credential, such as the IAM Role Credential, runZero enumerates the member accounts to sync all of their data.

Step 1: Create an AWS credential

runZero can authenticate to AWS with any of these credential types:

- IAM Role Credential (recommended): runZero authenticates to AWS with an IAM role that you create in your AWS account. No long-term access keys are needed.
    - IAM Role credentials are not supported for self-hosted instances, GovCloud-based accounts, or Explorer tasks. Use IAM Key credentials in those cases.
- IAM Role Credential - Manual Account Selection (alternative): runZero authenticates to AWS the same way as the IAM Role Credential, but with account IDs you specify by hand rather than AWS Organization subaccounts. Recommended only for advanced use cases.
- IAM Key Credential (deprecated): runZero authenticates to AWS with an IAM user and long-term access keys.

Option A: IAM role credential

To give runZero authenticated access to AWS through an IAM role, you need an **External ID**. An external ID is a unique ID that runZero provides for your IAM role. You use it when configuring the IAM role so that only your runZero account can access the role.

For more on external IDs in cross-account access, see AWS' documentation on the confused deputy problem.

Step 1a.1: Create a runZero credential and external ID

To generate an external ID, create an AWS IAM Role credential in the runZero console:

1. Go to the Credentials page and click **Add Credential**.

2. Choose **AWS IAM Role** from the list of credential types.

3. Name the credential, for example `AWS IAM Role`.

4. In the AWS IAM role ARN field, enter the full ARN of the IAM role you will create in the next step. An IAM role ARN has the format `arn:aws:iam::123456789012:role/runzero-sync`, where `123456789012` is your AWS account ID and `runzero-sync` is the name of the role you will create.

   If you don't know your AWS account ID yet, use `arn:aws:iam::XXXXXXXXXXXX:role/runzero-sync` as a placeholder and update it later.

5. Click **Save** to save the credential and generate an external ID.

   Verify & Save won't work yet, because the IAM role doesn't exist in AWS. You'll test the credential in Step 1a.3.

6. Copy the external ID. You'll need it in the next step.

Step 1a.2: Create AWS IAM role(s)

Follow the directions below to create an IAM role in every AWS account you want to sync with runZero. If you use AWS Organizations, reuse the same role name and external ID from your credential in each account.

The preferred way to create the IAM role is with CloudFormation StackSets:

1. Create a `runzero-cloudformation-stackset.json` file with the following content. Replace `<externalID>` with the external ID generated in the runZero console.

```
"Resources": {
  "IamrunZeroAccessRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "RoleName": "runzero-sync",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::336004405641:root"
          },
          "Action": "sts:AssumeRole",
          "Condition": {
            "StringEquals": {
              "sts:ExternalId": "<externalID>"
            }
          }
        }]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess",
        "arn:aws:iam::aws:policy/AmazonRDSReadOnlyAccess",
```

```
"arn:aws:iam::aws:policy/AmazonFSxReadOnlyAccess",
        "arn:aws:iam::aws:policy/AWSOrganizationsReadOnlyAccess",
        "arn:aws:iam::aws:policy/AmazonInspector2ReadOnlyAccess"
      ]
    }
  }
}
```

2. Go to **CloudFormation > StackSets** and click **Create StackSet**.
3. Select **Template is ready** and **Upload a template file**. Upload the template you just created.
4. Enter a name for the StackSet, for example runZero-IAM-Role. Click next.
5. (Optional) Set tags. Click next.
6. Set deployment options. Click next.
7. Review and create the StackSet.
8. The StackSet creates an IAM role with the format `arn:aws:iam::<accountID>:role/runzero-sync`. Copy the full ARN of the role. You'll need it in Step 1a.3.

To create the IAM role(s) manually instead:

1. Sign in to the AWS console.
2. Go to **Identity and Access Management (IAM) > Roles** to create a role that gives runZero access.
3. Click the **Create role** button.
4. On the "Select trusted entity" step:
    1. Select **AWS account** in the **Trusted entity type** panel.
    2. Select **Another AWS account** in the **An AWS account** panel.
    3. Enter `336004405641` for the account ID. This is the account runZero uses to access your AWS account.
    4. Check the **Require external ID...** checkbox and enter the external ID generated in Step 1a.1.
    5. Click the **Next** button.
5. On the "Add permissions" step, select the following AWS managed policies and click the **Next** button. These grant runZero read-only access to the inventory in your AWS account.
    - `AmazonEC2ReadOnlyAccess` to sync EC2, ELB, and VPC resources
    - `AmazonRDSReadOnlyAccess` to sync RDS instances
    - `AWSLambda_ReadOnlyAccess` to sync Lambda functions
    - `AmazonFSxReadOnlyAccess` to sync FSx file systems
    - `AWSOrganizationsReadOnlyAccess` to sync AWS Organization account information
    - `AmazonInspector2ReadOnlyAccess` to import AWS Inspector vulnerability findings for EC2 instances and Lambda functions
6. On the "Name, review, and create" step, enter a role name, such as `runzero-sync`, and a description. Click the **Create role** button.
7. Copy the full ARN of the new role. It has the format `arn:aws:iam::<accountID>:role/<roleName>`. You'll need it in Step 1a.3.

Additional notes:

- A StackSet often can't configure the payer account by default. In that case, the StackSet can configure all member accounts, and you may need to configure the payer account manually.

- If you intend to sync AWS regions that are not enabled by default, change the region compatibility of the global STS endpoint. In the AWS console, go to IAM > Account Settings > Security Token Service (STS) and change the Global Endpoint to be compatible with "All AWS Regions."

Step 1a.3: Update the runZero credential

1. Return to the runZero console, go to the Credentials page, and edit the credential you created in Step 1a.1.
2. Update the **AWS IAM role ARN** field with the full ARN of the role you created in Step 1a.2.
3. Click **Verify & Save** to save and test the credential.

The credential is ready. Continue to Step 2 to set up and activate the connection that brings in data from AWS.

Option B: IAM role credential with manual account selection

Choose this option only when the default IAM Role Credential can't target the subset of AWS accounts you need. It adds operational overhead, so weigh it carefully.

Step 1b.1: Create a runZero credential and external ID

This step is almost the same as step 1a.1.

To generate an external ID, create an AWS IAM Role credential in the runZero console:

1. Go to the Credentials page and click **Add Credential**.
2. Choose **AWS IAM Role - Manual Account Selection** from the list of credential types.
3. Name the credential, for example `AWS IAM Role - Manual`.
4. In the AWS IAM Role field, enter the name of the role you are using for all accounts.
5. Enter a newline-separated list of the account IDs to import asset information from. An AWS account ID is a 12-digit number, so an entry looks like `123456789012`.
6. Click **Save** to save the credential and generate an external ID.

   Verify & Save won't work yet, because the IAM role doesn't exist in AWS. You'll test the credential in Step 1b.3.
7. Copy the external ID. You'll need it in the next step.

Step 1b.2: Create AWS IAM role(s)

Follow the directions in step 1a.2: the setup is the same. Repeat them in each account you intend to import asset data from, because this credential type doesn't use AWS Organizations the way the IAM Role Credential does.

Step 1b.3: Update the runZero credential

1. Return to the runZero console, go to the Credentials page, and edit the credential you created in Step 1b.1.
2. Click **Verify & Save** to save and test the credential.

The credential is ready. Continue to Step 2 to set up and activate the connection that brings in data from AWS.

Option C: IAM key credential

If you created an IAM role in Option A above, skip this section and proceed to Step 2. This option authenticates runZero to AWS with an IAM user and long-term access keys, which we don't recommend. Use an IAM role instead.

Step 1c.1: Create an AWS IAM user

1. Sign in to the AWS console. If you use AWS Organizations and intend to sync multiple accounts, sign in to the AWS Organizations management account, sometimes called the "payer" or "root" account.
2. Go to **Identity and Access Management (IAM) > Users** and select or create a user that gives runZero API access.
3. Click **Add permissions > Attach existing policies directly**. Search for and attach the policies for the services you want to sync:
    - `AmazonEC2ReadOnlyAccess` to sync EC2, ELB, and VPC resources
    - `AmazonRDSReadOnlyAccess` to sync RDS instances
    - `AWSLambda_ReadOnlyAccess` to sync Lambda functions
    - `AmazonFSxReadOnlyAccess` to sync FSx file systems
    - `AWSOrganizationsReadOnlyAccess` to sync AWS Organization account information
    - `AmazonInspector2ReadOnlyAccess` to import AWS Inspector vulnerability findings for EC2 instances and Lambda functions
4. Add an inline policy that allows assuming runZero roles in sub-accounts (only required with AWS Organizations):

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::*:role/<rolename>"
    }]
}
```

5. From the user summary screen, open the **Security credentials** tab and click **Create access key**.
6. Save the **Access key ID** and **Secret access key**. You'll need them in Step 1c.3.
7. If you intend to sync AWS regions that are not enabled by default, change the region compatibility of the global STS endpoint. In the AWS console, go to IAM > Account Settings > Security Token Service (STS) and change the Global Endpoint to be compatible with "All AWS Regions."

Step 1c.2: Add an IAM role to sub-accounts

If you use AWS Organizations and want runZero to sync all accounts in your organization, create an IAM role in each sub-account that lets the IAM user from Step 1c.1 assume it. If you aren't using AWS Organizations, skip this step and proceed to Step 1c.3.

In every AWS sub-account:

1. Go to **Identity and Access Management (IAM) > Roles** and click **Create role**.
2. Choose **Another AWS Account** for the type of trusted entity.
3. For Account ID, enter the ID of your organization's management account.
4. Click **Next: Permissions**.
5. Attach the `AmazonEC2ReadOnlyAccess` policy if you want to sync EC2, ELB, and VPC resources.
6. Attach the `AmazonRDSReadOnlyAccess` policy if you want to sync RDS instances.
7. Attach the `AWSLambda_ReadOnlyAccess` policy if you want to sync Lambda functions.
8. Attach the `AmazonFSxReadOnlyAccess` policy if you want to sync FSx file systems.
9. Attach the `AmazonInspector2ReadOnlyAccess` policy if you want to import AWS Inspector vulnerability findings for EC2 instances and Lambda functions.
10. Click **Next: Tags** and optionally add tags.
11. Click **Next: Review** and name the role. (The role must have the same name in every account.)
12. Click **Create role**.

Step 1c.3: Add the AWS credential to runZero

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Name the credential, for example `AWS IAM User`.
3. Choose **AWS Access & Secret** from the list of credential types.
4. Fill in the credential:
    - In **AWS access key**, enter the access key ID from Step 1c.1.
    - In **AWS secret access key**, enter the secret access key from Step 1c.1.
    - In **AWS role**, enter the assumed role used to connect to other accounts in your organization. This is the role name you used in Step 1c.2.
    - Select the region(s) that you want to sync.
5. Save the credential.

Step 2: Choose how to configure the AWS integration

You can run the AWS integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 3: Set up and activate the AWS integration to sync data

After you add your AWS credential, set up a connector task or scan probe to sync your data.

Step 3a: Configure the AWS integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates new AWS-only assets.

1. Activate a connection to AWS. All available third-party connections are reachable from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, make sure the credential has access to your current organization.
3. Name the task, for example `AWS sync`.
4. Schedule the sync to run once or on a recurring schedule. It starts on the date and time you set.
5. Under **Task configuration**:
    - Choose the site to add your assets to. All newly discovered assets are stored in this site. You can also choose **Automatically create a new site per VPC** or **Automatically create a new site per account**, and runZero creates the sites for newly discovered assets or accounts.
    - Choose whether to **automatically delete stale AWS assets**. With this option checked, runZero deletes AWS assets it saw in an earlier sync but did not find in the most recent one.
    - Choose whether to include AWS assets that are not currently running. With this option checked, runZero imports asset data for AWS assets that are not in a running state.
6. Under **Service options**, select the services to sync data from. You must choose at least one.
7. To import AWS Inspector vulnerability findings for your EC2 instances and Lambda functions, switch the **Import AWS vulnerabilities** toggle to *Yes*. The credential's IAM role or user needs the `AmazonInspector2ReadOnlyAccess` managed policy (or the equivalent `inspector2:ListFindings` permission) attached in each account you sync. See Step 1: Create an AWS credential above. runZero imports findings only for resources that AWS Inspector has already scanned and matched to an existing EC2 instance or Lambda function.
8. To exclude from the import any assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes assets that runZero has not scanned.
9. Activate the connection when you are done. The sync runs on the schedule you set, and the Scheduled tasks page shows when the next sync will occur.

Step 3b: Configure the AWS integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters as needed.
3. On the Probes and SNMP tab, choose which additional probes to include, set the AWS toggle to *Yes*, and change any default options if needed.
4. On the Credentials tab, set the AWS toggle for the credential you want to use to *Yes*.
5. Click **Initialize scan** to save the scan task and run it immediately or at the scheduled time.

Step 4: View AWS assets

After a successful sync, go to your inventory to view your AWS assets. They show an AWS icon in the **Source** column.

To filter for AWS assets, try these queries:

- View all AWS assets:

  ```
  source:aws
  ```

- View all AWS EC2 instances:

  ```
  source:aws AND has:"@aws.ec2.instanceID"
  ```

- View all AWS Elastic Load Balancers:

  ```
  source:aws AND (has:"@aws.elb.loadBalancerArn" OR has:"@aws.elb.loadBalancerName")
  ```

- View all AWS VPCs:

  ```
  source:aws AND has:"@aws.vpc.vpcID"
  ```

Click into an asset to see its attributes as returned by the AWS APIs.

Troubleshooting

If the integration isn't working, the questions and answers below may help.

Why is the Amazon Web Services integration unable to connect?

1. Check whether the AWS integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the AWS integration can't connect, check the task log for errors. Common errors include:
   - 500: server error, unable to connect to the endpoint
   - 404: hitting an unknown endpoint on the server
   - 403: not authorized, likely a credential issue

Why aren't AWS Inspector vulnerabilities showing up on my EC2 or Lambda assets?

1. Confirm the **Import AWS vulnerabilities** option is enabled on the connector task or scan probe. This is separate from selecting the EC2 or Lambda service options.
2. Confirm the IAM role or user behind the credential has the `AmazonInspector2ReadOnlyAccess` managed policy (or a custom policy granting `inspector2:ListFindings`) attached in the specific AWS account and region being synced. Every member account needs this permission, not just the payer/root account. If it is missing, the task log shows an `AccessDeniedException` similar to:

   ```
   AccessDeniedException: User: arn:aws:sts::<account-id>:assumed-role/<role-name>/runZeroAWSIntegrationSession is
   authorized to perform: inspector2:ListFindings on resource: arn:aws:inspector2:<region>:<account-id>:/findings/l
   because no identity-based policy allows the inspector2:ListFindings action
   ```

3. Confirm that AWS Inspector scanning (EC2 deep inspection and/or Lambda standard scanning) is activated for the account and region, and that AWS Inspector has scanned the resource. runZero imports only findings that AWS Inspector has already generated; it does not trigger new scans.
4. For Lambda functions, confirm the function was invoked or updated within the last 90 days, uses the `$LATEST` version, and is not excluded from scanning by tag or a customer-managed KMS key.

Enriching scans with EC2

During a discovery scan, runZero automatically enriches scanned assets with data from the AWS EC2 API when it's available. Assets pick up internal IP addresses, external IP addresses, hostnames, MAC addresses, and tags, along with other EC2-specific attributes such as the account ID and instance type.

runZero needs no extra configuration for this enrichment, but you may need to modify the permissions on the instance's IAM role.

Find Explorers with EC2 enrichment capabilities

EC2 enrichment works when the runZero Explorer runs on an instance with permission to describe your EC2 instance list. You can grant that through an IAM role attached to the instance or by configuring AWS credentials for the root user account.

To see which Explorers have this capability, view your registered Explorers. An Explorer with the cloud icon can enumerate EC2 instances.

Scans from these Explorers automatically merge EC2 instance fields into the asset for any in-scope target that matches the instance list.

Add permissions to describe instances

To enable EC2 instance data enrichment, add the EC2 `ec2:DescribeInstances` permission to the instance role configured for your instance. The Amazon docs explain how to create and

update policies.

Your policy will look like this:

```
{
   "Version": "2012-10-17",
   "Statement": [{
     "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
         "ec2:DescribeInstances",
      ],
      "Resource": "*"
   }
   ]
}
```

In the IAM UI, go to **Roles > Permissions > Attach policies** and search for the EC2 service. Under the actions, select `DescribeInstances`, which is listed under List.

Instead of the IAM instance role, you can also configure credentials on the instance by running `aws configure` as root.

After you save your policy, restart your Explorer. The easiest way is to force an update from the Explorer menu.

If your configuration uses one region but the instance is in another, the Explorer uses the instance's region for all API requests.

Attributes runZero gets from the EC2 API

When runZero determines that an IP address is also an EC2 instance, it enriches the existing asset data with EC2 metadata, including more hostnames (based on AWS asset tags), MAC addresses, and internal and external IPs.

Attributes runZero can get from the EC2 API metadata include:

- aws.accountID
- aws.architecture
- aws.availabilityZone
- aws.hypervisor
- aws.imageID
- aws.instanceID
- aws.instanceType
- aws.ipv4
- aws.ipv6
- aws.keyName
- aws.launchTimeTS
- aws.macs
- aws.privateDNS
- aws.privateIP
- aws.publicDNS

- aws.publicIP
- aws.region
- aws.rootDeviceName
- aws.rootDeviceType
- aws.scanner.instanceID
- aws.scanner.instanceType
- aws.state
- aws.subnetID
- aws.tags
- aws.tenancy
- aws.virtualizationType
- aws.vpcID

runZero also reports other things that respond to the scan but don't match an EC2 entry. You may see Amazon RDS, temporary ELBs, and Lambdas in your inventory, but only EC2 assets carry EC2 metadata. For broader visibility across your AWS environment, set up the Amazon Web Services integration.

Censys Search & Data

runZero can import assets from the Censys Search API. For the Censys Platform API, see Censys Platform.

Censys Search API

To use the Censys Search API, register for a Censys Search account. Your API credentials are in the My Account section.

In runZero, go to the Credentials page and click Add Credential. Select *Censys Search API Key* as the credential type, and enter your API ID and API secret.

Then go to your asset inventory, click the **Connect** button, and choose **Censys Search API**. Select the credential you just created from the Censys Search credential dropdown.

Configuration

runZero connects to the Censys Search API in one of two modes.

- In **Custom Query** mode, runZero runs a Censys search query you specify and imports all of the results. Write the query in Censys Search Language, and test it in the main Censys Search 2.0 interface before running an import task.
- In **All Assets** mode, runZero assembles a list of public IP addresses from all of the assets in the selected site, then uses the API to find Censys Search information for those addresses. It imports what it finds and merges it into the matching assets.

As with a runZero scan, select a site to hold the scan data. The usual task scheduling options are available.

When you have finished editing the Censys Search configuration, click **Activate Connection**.

Troubleshooting

If the integration isn't working, the questions and answers below may help.

Why is the Censys Search integration unable to connect?

1. Check whether the Censys Search integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Censys Search integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Censys Platform

runZero imports assets from the Censys Platform API.

- Importing assets from the Censys Platform API

Censys Platform API

You need a Censys Platform account to use the API. Once registered, find your API credentials in the My Account section.

In runZero, go to the Credentials page and click Add Credential. Select *Censys Platform API Key* as the credential type, then enter your API ID and API secret.

Now go to your asset inventory, click **Connect**, and choose **Censys Platform API**. Select the credential you just created from the Censys Platform credential dropdown.

Configuration

runZero connects to the Censys Platform API in one of two modes.

- In **Custom Query** mode, runZero runs a Censys query you specify and imports all the results. Write the query in Censys Query Language. It's a good idea to test it in the main Censys Platform interface before you run an import task.

- In **All Assets** mode, runZero assembles a list of public IP addresses from all the assets in the selected site, then uses the API to look up Censys information for those

addresses. runZero imports what it finds and merges it into the corresponding assets.

As with a runZero scan, you'll need to select a site to hold the scan data. The usual task scheduling options are available.

When you're done editing the Censys Platform configuration, click **Activate Connection**. After the task runs, look for the imported assets in your inventory.

Troubleshooting

If you're having trouble with this integration, start with the questions and answers below.

Why is the Censys Platform integration unable to connect?

1. Check whether the Censys Platform integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Censys Platform integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Cisco Meraki Dashboard

runZero imports Cisco Meraki Dashboard data through the Cisco Meraki Dashboard API. With your Meraki devices and network clients in runZero, you get better visibility of your network.

Getting started with Meraki

To set up the Meraki integration, you'll need to:

1. Generate an API key for your Cisco Meraki Dashboard administrator account.
2. Add the Meraki credential to runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the integration to pull your data into runZero.

Requirements

Before you set up the Meraki integration:

- Make sure you have administrator access to the Meraki Dashboard.

Step 1: Generate an API key in Cisco Meraki Dashboard

1. Sign in to Cisco Meraki Dashboard with an administrator account.
2. Check that access to the Cisco Meraki Dashboard API is enabled. Each organization you want to sync needs API access enabled.
    1. Go to **Organization > Settings** and enable **API access** under **Dashboard API access**.
3. Go to **My profile** and generate an API key. The key inherits the permissions of the account that created it.

Step 2: Add the Meraki API key to runZero

1. Go to the Credentials page in runZero.
2. Choose **Meraki API Key** from the list of credential types.
3. Give the credential a name, like `Meraki`.
4. Fill in the connection details:
    - For **Meraki API URL**, enter the API endpoint URL for the Cisco Meraki Dashboard API.
    - For **Meraki API key**, enter the API key for the Cisco Meraki Dashboard administrator account.
5. To let other organizations use this credential, select *Make this a global credential*. Otherwise, you can configure access per organization.
6. Save the credential.

Step 3: Choose how to configure the Meraki integration

You can run the Meraki integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

After you add your Meraki credential, set up the sync that pulls your data from Meraki.

Step 4a: Configure the Meraki integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Meraki-only assets.

1. Activate a connection to Meraki. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they aren't listed, check that they have access to the organization you're currently in.
3. Optionally, list the organization IDs or names to include in the import, separated by commas. runZero imports data only for the organizations you list.

4. Optionally, list the network IDs or names to include in the import, separated by commas. runZero imports data only for the networks you list.
5. Optionally, list the VLANs to exclude from the import, separated by commas. runZero skips devices associated with those VLANs.
6. To exclude assets that have no VLAN, set the **Exclude clients with no VLAN** option to *Yes*.
7. Optionally, list the SSIDs to exclude from the import, separated by commas. runZero skips devices connected on those SSIDs.
8. Enter a name for the task, like `Meraki Sync` (optional).
9. Choose the Explorer that runs this connector task (optional).
10. Choose the site for your assets. runZero stores all newly discovered assets in this site.
11. Enter a description for the task (optional).
12. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes assets runZero has not scanned.
13. Schedule the sync to run once or on a recurring schedule. It starts on the date and time you set.
14. Activate the connection when you're done. The sync runs on the schedule you defined, and Scheduled tasks shows when the next sync will occur.

Step 4b: Configure the Meraki integration as a scan probe

When you run the Meraki integration as a scan probe, the runZero Explorer pulls your Meraki assets into the runZero Console.

In a new or existing scan configuration:

- Set the `MERAKI` option to *Yes* in the *Probes and SNMP* tab and change any default options if needed.
- Set the correct `MERAKI` credential to *Yes* in the *Credentials* tab.

Step 5: View Meraki assets

After a successful sync, go to your inventory to view your Meraki assets. They show a Cisco icon in the **Source** column.

To filter for Meraki assets, run this query:

- View all Meraki assets:

  ```
  source:Meraki
  ```

Click into any asset to see the attributes runZero gathered from Meraki. To narrow the results further, use the asset inventory search keywords.

CrowdStrike Falcon

runZero imports CrowdStrike data through the CrowdStrike Falcon API to sync and enrich your asset inventory, pull vulnerability data from Falcon Spotlight, and pull software data from Falcon Discover. With CrowdStrike data in runZero, finding endpoints that are missing an EDR agent takes one query.

Getting started

To set up the CrowdStrike integration, you'll need to:

1. Configure CrowdStrike to allow API access to runZero.
2. Add the CrowdStrike credentials to runZero: the client ID, the client secret, and the CrowdStrike base API URL.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the CrowdStrike integration to sync your data with runZero.

Requirements

Before you set up the CrowdStrike integration:

- Make sure you have access to the CrowdStrike admin portal.

Step 1: Configure CrowdStrike to allow API access to runZero

1. Sign in to CrowdStrike.
2. Go to **Support > API Clients and Keys**. When the **API Key** page appears, add a new API client.
3. Fill in the API client details:
    - For **Client name**, enter a name such as runZero.
    - For **API scope**, choose the permissions for the data you want to ingest:
        - To ingest host details, include read permissions for **Hosts** and **Host Groups**.
        - To ingest vulnerability data, include read permissions for **Vulnerabilities**.
        - To ingest software data, include read permissions for **Assets**.
4. Add the client. A window confirms the API client was created and shows the client ID and client secret. You'll need both to configure the integration in runZero.
5. Copy the client ID and client secret now. You may not be able to get them later.

Step 2: Add the CrowdStrike credentials to runZero

1. Go to the Credentials page in runZero. Give the credentials a name, like `CrowdStrike Falcon`.
2. Choose **CrowdStrike Falcon API key** from the list of credential types.
3. Fill in the connection details:
    - For **CrowdStrike client ID and CrowdStrike client secret**, enter the pair you generated under **Support > API Clients and Keys > OAuth2 API clients > Add new API Client** in your CrowdStrike portal.
    - For **CrowdStrike API URL**, enter your organization's base URL. It depends on your account type, and the CrowdStrike API Swagger documentation lists it.
        - For a **US-1** account, use `api.crowdstrike.com`
        - For a **US-2** account, use `api.us-2.crowdstrike.com`
        - For a **US-GOV-1** account, use `api.laggar.gcw.crowdstrike.com`
        - For an **EU-1** account, use `api.eu-1.crowdstrike.com`

4. To let other organizations use these credentials, select *Make this a global credential*. Otherwise, you can configure access per organization.
5. Save the credentials.

Step 3: Choose how to configure the CrowdStrike integration

You can run the CrowdStrike integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the CrowdStrike integration to sync data

With your CrowdStrike credential saved, set up a connector task or scan probe to sync your data.

Step 4a: Configure the CrowdStrike integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new CrowdStrike-only assets.

1. Activate a connection to CrowdStrike. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, check that it has access to the organization you're currently in.
3. Set the severity and risk levels you want to import (optional).
4. Set the *Fingerprint only* toggle to *Yes* to ingest vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
5. Add a filter for imported assets (optional).



If the CrowdStrike API key has access to Falcon Discover or Falcon Spotlight, runZero imports software and vulnerability data only for the assets in the filtered results.
6. Enter a name for the task, like `CrowdStrike sync` (optional). 7. Choose the Explorer that runs this connector task (optional). 8. Choose the site for your assets. runZero stores all newly discovered assets in this site. 9. Enter a description for the task (optional). 10. To leave out assets that runZero has not scanned, switch the _Exclude unknown assets_ toggle to _Yes_. By default, the integration includes assets runZero has not scanned. 11. Schedule the sync to run once or on a recurring schedule. It starts on the date and time you set. 12. Activate the connection when you're done. The sync runs on the schedule you defined, and [Scheduled tasks](https://console.runzero.com/tasks) shows when the next sync will occur.

Step 4b: Configure the CrowdStrike integration as a scan probe

1. Create a new scan task, or select a future or recurring scan task from your Tasks page.

2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose which additional probes to include, set the CrowdStrike toggle to *Yes*, and change any default options if needed.
4. On the Credentials tab, set the CrowdStrike toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 5: View CrowdStrike assets and vulnerabilities

After a successful sync, go to your inventory to view your CrowdStrike assets and vulnerabilities. They show a CrowdStrike icon in the **Source** column.

To filter by CrowdStrike attributes, try these queries:

- View all CrowdStrike assets

  ```
  source:crowdstrike
  ```

- Find assets that have a CrowdStrike EDR agent installed

  ```
  edr.name:crowdstrike
  ```

- Find Windows assets, excluding servers, that are missing a CrowdStrike EDR agent

  ```
  os:windows and not type:server and not edr.name:CrowdStrike
  ```

- View all CrowdStrike vulnerabilities

  ```
  source:crowdstrike
  ```

- View all CrowdStrike software results

  ```
  source:crowdstrike
  ```

Click into an asset or vulnerability to see its individual attributes. runZero shows the attributes the CrowdStrike API returned, except policies. If missing EDR agents are what you're after, continue with the Finding gaps in endpoint protection playbook.

Filtering CrowdStrike assets

You can add a filter to CrowdStrike integration tasks. runZero filters with CrowdStrike's Falcon Query Language (FQL), which follows the syntax `<property>:[operator]<value>`. Combine expressions with a `+` between them for more complex filters, or separate them with commas for an OR expression.

Properties

These CrowdStrike properties are useful in FQL filters. CrowdStrike's API documentation lists the other attributes you can filter on.

CrowdStrike Property	runZero Attribute	Description	Example
external_ip	externalIP	The external IP address of the device	18.191.169.203
first_seen	firstSeen	The timestamp of when CrowdStrike first saw the device	2022-01-08T19:42:34Z
hostname	hostname	The hostname of the device	EXPLORER-01
last_seen	lastSeen	The timestamp of when CrowdStrike last saw the device	2022-09-13T19:14:30Z
local_ip	localIP	The local IP address of the device	192.168.1.100
mac_address	macAddress	The MAC address of the interface communicating with CrowdStrike	0a-6e-20-4a-e6-56
os_version	osVersion	The operating system version of the device	Ubuntu 20.04
platform_name	platformName	The platform running on the device	Linux
product_type_desc	productTypeDesc	The type of device	Server

Operators

FQL filters accept these operators.

Operator	Description
!	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
~	Text match. Tokenizes the string, ignoring spaces, case and punctuation
!~	Does not text match. Tokenizes the string, ignoring spaces, case and punctuation
*	Wildcard matching. Matches one or more characters

Example filters

You can apply filters like these to the CrowdStrike sync.

Search Filter	Description
`hostname:'WIN10*'`	Import all devices where the hostname starts with WIN10
`platform_name:'Linux'`	Import all Linux devices
`product_type_desc:'Server'`	Import all devices that CrowdStrike identifies as a Server
`hostname:'PROD*'+platform_name:'Linux'`	Import all Linux devices with a hostname that starts with PROD
`local_ip:'192.168.1.100'`	Import only the device with a local IP address of 192.168.1.100
`local_ip:!'192.168.1.100'`	Import all devices, excluding 192.168.1.100
`local_ip.raw:*'192.168.1.*'`	Import all devices with a local IP address in the 192.168.1.0/24 range
`(local_ip.raw:*'192.168.1.*'), (local_ip.raw:*'192.168.2.*')`	Import all devices with a local IP address in the 192.168.1.0/24 or 192.168.2.0/24 range
`local_ip.raw:!*'192.168.1.*'`	Import all devices, excluding devices with a local IP address in the 192.168.1.0/24 range
`local_ip.raw:!*'192.168.1.*'+local_ip.raw:!*'192.168.2.*'`	Import all devices, excluding devices with a local IP in the 192.168.1.0/24 and 192.168.2.0/24 ranges

Troubleshooting

If you’re having trouble with this integration, start with the questions and answers below.

Why is the CrowdStrike integration unable to connect?

1. Check whether the CrowdStrike integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration’s configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the CrowdStrike integration can’t connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server

    - 403: not authorized, likely a credential issue
4. If the integration endpoint is on-premises, verify that the integration task runs from an Explorer with access to the CrowdStrike host.

How can I solve the following CrowdStrike error?

```
Unable to collect software data for CrowdStrike devices: invalid response 403 Forbidden
```

This error occurs when your API client is missing the **Assets** API scope. To fix it, return to step 1 above and enable read permissions for **Assets**. The integration needs read-only permissions for **Assets** to collect software information. Host and vulnerability data should still come through.

Custom Integration Scripts

Custom integration scripts let you import asset data from any source that runZero does not already integrate with, or export runZero data to another system. You write them in Starlark, a Python-like language, and they run in a sandbox on one of your Explorers or on the runZero console itself.

runZero 5.1 expanded what custom integration scripts can do:

- An embedded `CONFIG` block describes the integration and its parameters. From it, the console generates a typed credential form, validates input, applies defaults, and routes secrets through encrypted storage.
- Shared TLS and HTTP option suites (`OPTIONS_TLS` and `OPTIONS_HTTP`) add standard connection controls to any script without copying boilerplate parameters.
- `report_assets` streams assets to runZero page by page, so a large import no longer has to sit in memory.
- A CONFIG-level `matchBehavior` policy and per-asset `trust_*` flags let a script control how imported assets merge with existing inventory.
- A much larger library of built-in modules, including typed `kwargs` accessors, HTTP helpers with retry support, parsers for CSV/XML/streaming JSON, JWT and AWS SigV4 signing, and direct-protocol modules for SSH, SMB, WinRM, WMI, and SQL.
- The `runzero script` CLI command gained a `--validate` mode that smoke-tests a script's CONFIG and HTTP/TLS wiring against a local dummy server, and an `--output` mode that writes real scan output for inspection.
- The custom integration page in the console became a script workbench with an editor, test runs against real endpoints, a shipped example library, version history, and, where AI is enabled, a generative AI assistant that drafts and revises scripts.

The runzero-custom-integrations GitHub repository holds a library of ready-to-use integrations built on these capabilities, including a boilerplate template to start from.

Overview

To set up a custom integration script:

1. Write the script, starting with a `CONFIG` block and a `main` function.

2. Test it locally with the runZero CLI (`runzero script`).
3. Create the custom integration in the console and paste in the script.
4. Create a credential. The console builds the form from the script's `CONFIG` parameters.
5. Create an integration task to run the script on a schedule.

Writing a script

The Starlark language

Starlark is a dialect of Python with some notable differences:

1. Starlark has no exception handling (`try`/`except`). Use return values to signal errors.
2. F-string `f'{var}'` formatting is unavailable. Use `"{}".format(var)` for string interpolation.
3. The standard library is limited to the built-in modules that runZero provides.

The custom integration sandbox also rejects recursion, top-level `if`/`for` statements, and reassignment of global variables. `while` loops and the `set()` type are available.

The entrypoint

Every script must define a `main` function. runZero calls it with the task's arguments and keyword arguments:

```
load('runzero.types', 'ImportAsset')

def main(*args, **kwargs):
    asset = ImportAsset(
        id='device-1',
        hostnames=['web1.example.com'],
        os='Linux',
        osVersion='6.1',
        manufacturer='Example Corp',
        model='VM',
    )
    return [asset]
```

- `args` receives any positional arguments configured on the task, as strings.
- `kwargs` receives the credential fields and task keyword arguments, as strings. Declare every key your script reads in `CONFIG["params"]`, and read them with the typed `kwargs` accessors.
- For inbound integrations, `main` returns a `list` of `ImportAsset` objects (or a single `ImportAsset`). Returning `None` is valid when the script streams assets with `report_assets` instead.
- For outbound integrations, `main` does its export work and returns `None`.

Streaming large imports with report_assets

For sources with large inventories, stream each page to runZero as you process it. Returning one large list from `main` holds every asset in memory at once, along with the raw API responses used to build them. `report_assets` is a predeclared builtin, so it needs no `load()`:

```
load('runzero.types', 'ImportAsset')

def fetch_page(cursor):
    # Replace with a real paginated API call.
    if cursor == None:
        return [{"id": "a-1"}, {"id": "a-2"}], "page-2"
    return [{"id": "b-1"}], None

def main(*args, **kwargs):
    total = 0
    cursor = None
    while True:
        page, cursor = fetch_page(cursor)
        if not page:
            break
        assets = [ImportAsset(id=item["id"]) for item in page]
        total += report_assets(assets)   # stream this page to runZero
        if not cursor:
            break
    print("reported {} assets".format(total))
    return None                          # nothing buffered in main
```

`report_assets` accepts a single asset, several positional assets, or a list/tuple of assets, and returns the number reported. Assets streamed with `report_assets` and assets returned from `main` are both imported, so partial adoption is safe.

The singular `report_asset` (also predeclared) takes exactly one `ImportAsset` (or `None`, a no-op) and returns `1` or `0`. Streaming already batches internally, so the simplest correct pattern is to report each asset as you build it. Accumulating a page into a list first gains nothing:

```
for device in devices:
    report_asset(build_asset(device))
```

For the pagination loop itself, the predeclared `pager` builtin guards against a cursor that never advances. `p = pager(label="devices")` returns an object whose `p.next()` is the `while` condition. If the loop reaches the page ceiling (`maxPages`, default 1,000,000), the script stops with an error naming the label rather than truncating silently or spinning forever. `max_pages()` returns the effective ceiling.

The CONFIG block

The `CONFIG` block is an embedded, declarative description of the integration. When you save a script in the console, runZero extracts `CONFIG` (without executing the script) and uses it to:

- Render a typed credential form with labels, groups, and conditional fields.
- Validate submitted values (`required`, `min`, `max`, `pattern`, `options`) and apply `default` values before `main` runs.
- Route `secret` parameters through encrypted storage and redact their values from script logs and error messages.
- Refuse to run the script on runZero versions older than `minVersion`.

```
CONFIG = {
    "id": "runzero-example",
    "name": "Example integration",
    "type": "inbound",
    "description": "Imports devices from the Example API.",
    "version": "1",
    "minVersion": "5.1.0",
    "params": [
        {"key": "url", "label": "Example API URL", "type": "url", "required": True},
        {"key": "api_token", "label": "API token", "type": "secret", "required": True},
        {"key": "page_size", "label": "Page size", "type": "int", "required": False,
         "default": 100, "min": 1, "max": 1000},
    ],
    "includes": {
        "tls_": OPTIONS_TLS,
        "http_": OPTIONS_HTTP,
    },
}

load("runzero.types", "ImportAsset")
# ...rest of script
```

Rules for the block itself:

- `CONFIG` must be the first top-level statement in the script. Comments and blank lines may come before it; `load(...)` calls, constants, and other statements may not.
- Every value must be a literal: strings, numbers, `True`/`False`/`None`, lists, tuples, and dicts with string keys. Function calls, variable references, and arithmetic are rejected. The one exception is `CONFIG["includes"]`, which may reference the predeclared option-set identifiers `OPTIONS_TLS` and `OPTIONS_HTTP`.
- Scripts without a `CONFIG` block continue to run with the legacy `access_key` and `access_secret` credential fields, so existing integrations keep working unchanged.
- `sourceId` and `sourceName` are reserved for runZero's own shipped integrations. runZero refuses to save a script that declares either.

Top-level fields

Field	Description
`id`	Stable lower-case identifier, for example `runzero-tailscale`.
`name`	Display name for the integration.
`type`	`inbound`, `outbound`, or `internal` (descriptive).
`description`	Short summary of what the integration does.
`version`	The script's own revision, bumped when the script changes, for example `"1"`.
`minVersion`	Minimum runZero version that can run the script, for example `5.1.0`. Older Explorers refuse to run it and report an upgrade message.
`params`	List of parameter definitions.

`includes`	Shared option suites, keyed by prefix.
`atLeastOneOf`	List of parameter-key groups; at least one key in each group must be set.
`exactlyOneOf`	List of parameter-key groups; exactly one key in each group must be set.
`rejectUnknown`	Reject keys that are not declared in `params`. Unknown keyword arguments are always rejected at run time for CONFIG-based scripts.
`validationMode`	How `runzero script --validate` exercises the script: the default expects an HTTP request; `"compile"` only checks that CONFIG parses and `main` exists. Use `"compile"` for direct-protocol (SSH/SMB/WMI/WinRM/SQL) integrations.
`maxPages`	Page ceiling enforced by the predeclared `pager` builtin; defaults to 1,000,000. A script may lower the effective ceiling by passing `limit=` to `pager`, but not raise it.
`matchBehavior`	Integration-wide merge policy, a space-separated string of flags.
`assetType`	Asset type applied to every asset the integration reports, overridable per asset with the `assetType` field on `ImportAsset`.
`assetTypeBehavior`	Map of asset type to a `matchBehavior`-style flag string, layered over the integration-wide `matchBehavior` for assets of that type. Also accepted as `sourceTypeBehavior`; declaring both with different values is an error.
`ownershipAttributes`	Attribute keys the integration supplies that should be treated as ownership data.
`trustOS`, `trustOSVersion`, `trustType`	Trust this integration's OS, OS version, or device type over other sources, for every asset it reports. The per-asset `trust_*` fields on `ImportAsset` do the same thing for one asset.

Parameter definitions

Each entry in `params` describes one credential form field and one keyword argument delivered to `main`.

`type` accepts `string`, `secret`, `int`, `float`, `bool`, `enum` (requires `options`), `url`, `textarea`, and `json`.

Field	Description
`key`	Keyword argument name. Must match `^[a-zA-Z_][a-zA-Z0-9_]*$` and match the name the script reads.
`label`	Form label.
`description`	Help text shown under the field.
`type`	One of the types above. `secret` values get masked input, encrypted storage, and log redaction.

`secret`	Treat the value as a secret regardless of its `type`. Equivalent to `type: "secret"` for masking, storage and redaction.
`required`	Whether the field must be set.
`default`	Default value applied when the field is left blank. Not allowed on `secret` parameters.
`placeholder`	Placeholder text for the form field.
`options`	Allowed values for `enum` parameters.
`multi`	For `enum`: allow multiple comma-separated selections.
`min` / `max`	Numeric bounds for `int`/`float`; length bounds for `string`/`secret`/`textarea`.
`pattern`	Regular expression the value must fully match (`string`, `secret`, `textarea`, `url`).
`aliases` / `caseInsensitive`	For `enum`: alternate spellings that normalize to a canonical option before `main` runs.
`dependsOn`, `visibleIf`, `visibleIfValue`	Show this field only when another declared field is set (optionally to a specific value).
`requiredIf`, `requiredIfValue`	Make this field required when another declared field is set (optionally to a specific value).
`group`	Section heading used to group fields in the form.

How values reach your script

- Each declared parameter arrives in `kwargs` under its `key`, as a string. Read it with the `kwargs` module (`get_string`, `get_int`, `get_bool`, `get_list`, ...) to get type coercion and defaults.
- runZero applies declared `default` values, normalizes enum aliases, and runs validation before it calls `main`.
- CONFIG-based scripts reject unknown keyword arguments.
- Credential keys starting with `_` (for example `_integration_id`) stay on the credential and never reach the script.
- For backward compatibility, runZero still injects the credential's `access_key` and `access_secret` fields when the task does not provide them.

Shared TLS and HTTP option suites

Most integrations talk to an HTTPS API, and most of them need the same connection controls: trusting a private CA, pinning a certificate, presenting a client certificate, disabling validation in a lab, or sending a specific `User-Agent`. Rather than declaring these parameters in every script, add the predeclared option suites to `CONFIG["includes"]`:

```
    "includes": {
        "tls_": OPTIONS_TLS,
```

```
"http_": OPTIONS_HTTP,
    },
```

Each include expands into a set of parameters, and the dict key (`"tls_"`, `"http_"`) is prepended to every generated parameter key. In the credential form, suite parameters render after the script's own fields as collapsed sections named for the suite (**TLS**, **HTTP**); a section holding an empty required field is forced open with a **Needs attention** badge. When you include the same suite under two prefixes, each section is named for its prefix, for example **TLS (src_tls)**.

OPTIONS_TLS

With the conventional `tls_` prefix, `OPTIONS_TLS` generates these keyword arguments:

Generated kwarg	Type	Default	Description
`tls_disable_validation`	`bool`	`False`	Allow connections to endpoints with untrusted TLS certificates.
`tls_ca_cert`	`textarea`	no default	PEM-encoded certificate authorities to trust when validating the endpoint certificate.
`tls_peer_hash`	`string`	no default	SHA-256 fingerprint of the endpoint certificate to trust (certificate pinning). Separate multiple pins with commas, semicolons, spaces, or newlines.
`tls_client_cert`	`textarea`	no default	PEM-encoded client certificate to present for mutual TLS.
`tls_client_key`	`secret`	no default	PEM-encoded private key for the client certificate. Required when `tls_client_cert` is set.

OPTIONS_HTTP

With the conventional `http_` prefix, `OPTIONS_HTTP` generates:

Generated kwarg	Type	Default	Description
`http_user_agent`	`string`	`""`	Optional `User-Agent` header sent with HTTP requests.

Using the option suites

Declaring the includes only creates the form fields; the script must pass the collected values to the HTTP client. The `kwargs` module does this in one call:

```
load("http", "get_json", "bearer")
load("kwargs", "get_string", "get_url_base", "get_http_options")

def main(*args, **kwargs):
    http_options = get_http_options(kwargs, headers={
        "Authorization": bearer(get_string(kwargs, "api_token")),
        "Accept": "application/json",
```

```
})
    data, err = get_json("{}/v1/devices".format(get_url_base(kwargs)), **http_options)
```

`get_http_options(kwargs, prefix="http_", tls_prefix="tls_", headers=None)` gathers the suite values into a dict of keyword arguments (`headers=` and `tls=`) that you can splat into any `http` module function (`get`, `post`, `get_json`, `post_json`, `oauth2_token`, and so on). The mapping is:

Suite kwarg	Effect
`tls_disable_validation`	`tls={"insecure": True}`
`tls_ca_cert`	`tls={"ca_pem": ...}`
`tls_client_cert` / `tls_client_key`	`tls={"client_cert_pem": ..., "client_key_pem": ...}`
`tls_peer_hash`	`tls={"thumbprints": [...]}`
`http_user_agent`	`headers={"User-Agent": ...}` (only when the header is not already set)

Use `get_http_tls(kwargs, "tls_")` when a script only needs the `tls=` dict and manages headers itself. You can also build the `tls=` dict by hand. It accepts the keys `insecure`, `server_name`, `ca_pem`, `client_cert_pem`, `client_key_pem`, and `thumbprints`, and rejects anything else.

Two things to watch:

- `runzero script --validate` verifies the wiring: a script that declares `OPTIONS_HTTP` or `OPTIONS_TLS` but never passes the collected options to an HTTP call fails validation.
- The stateful `requests.Session` object does not accept the `tls=` dict; only its `insecure_skip_verify` constructor flag is available. Prefer the `http` module functions when the TLS suite matters.

Multiple endpoints

A script that talks to more than one endpoint can include a suite more than once under different prefixes and collect each set separately:

```
    "includes": {
        "src_tls_": OPTIONS_TLS,
        "src_http_": OPTIONS_HTTP,
        "dst_tls_": OPTIONS_TLS,
        "dst_http_": OPTIONS_HTTP,
    },
```

```
src_options = get_http_options(kwargs, "src_http_", "src_tls_", src_headers)
dst_options = get_http_options(kwargs, "dst_http_", "dst_tls_", dst_headers)
```

A complete example

The script below puts the pieces together: a CONFIG block with typed parameters and both option suites, option-suite plumbing through `get_http_options`, paginated fetching with `get_json`,

and streaming import through `report_assets`. It passes `runzero script --validate` as-is, and you can use it as the starting point for a real integration (see also the boilerplate template on GitHub).

```
CONFIG = {
    "id": "runzero-example",
    "name": "Example integration",
    "type": "inbound",
    "description": "Imports devices from the Example API.",
    "version": "1",
    "minVersion": "5.1.0",
    "params": [
        {"key": "url", "label": "Example API URL", "type": "url", "required": True},
        {"key": "api_token", "label": "API token", "type": "secret", "required": True},
    ],
    "includes": {
        "tls_": OPTIONS_TLS,
        "http_": OPTIONS_HTTP,
    },
}

load("runzero.types", "ImportAsset", "to_custom_attributes")
load("net", "network_interface")
load("http", "get_json", "bearer")
load("kwargs", "require", "get_string", "get_url_base", "get_http_options")

def build_asset(device):
    return ImportAsset(
        id=str(device["id"]),
        hostnames=[device.get("hostname", "")],
        os=device.get("os", ""),
        networkInterfaces=[network_interface(mac=device.get("mac"),
                                             ips=device.get("ips", []))],
        customAttributes=to_custom_attributes(device),
    )

def main(*args, **kwargs):
    require(kwargs, "url", "api_token")
    base_url = get_url_base(kwargs)
    http_options = get_http_options(kwargs, headers={
        "Authorization": bearer(get_string(kwargs, "api_token")),
        "Accept": "application/json",
    })

    total = 0
    page = 1
    while True:
        data, err = get_json("{}/v1/devices".format(base_url),
                             params={"page": str(page)}, retries=2, **http_options)
        if err:
            print("request failed: {}".format(err))
            break
```

```
if not devices:
            break
        total += report_assets([build_asset(d) for d in devices])
        page += 1

    print("imported {} devices".format(total))
    return None

runzero script --filename example-integration.star --validate

INFO example-integration.star validated with 1 HTTP request(s)
INFO validated 1 script(s) with dummy HTTP/TLS server https://127.0.0.1:52084
```

Controlling how assets merge

By default, imported assets merge with existing inventory on the asset `id`, MAC address, IP address, and hostname. You declare merge policy once for the whole integration, as the `matchBehavior` key of the CONFIG block:

```
CONFIG = {
    # ...
    "matchBehavior": "no-mac-break no-ip-break no-name-break",
}
```

`matchBehavior` accepts a space-separated string of flags built from `no-` + (`id`|`mac`|`ip`|`name`) + (`-match`|`-break`), plus the break-only `no-type-break`, which allows merges across an integration's own asset types. Two presets cover most cases:

- `"no-mac-break no-ip-break no-name-break"`: use this when your source supplies a stable, unique id (vendor UUID, serial number). The id still drives merges, but differing MACs, IPs, or names will not disqualify a merge with an existing asset.
- `"no-id-match no-id-break"`: use this when your source only emits ephemeral or per-run ids. runZero ignores the id and merges on MAC, IP, and hostname instead.

`matchBehavior` is no longer a per-asset field on `ImportAsset`: a script that reads or sets `ImportAsset.matchBehavior` fails with an error pointing at the CONFIG key.

Two related controls remain per-asset:

- `assetType` on `ImportAsset` overrides the CONFIG-level `assetType` for one asset. When the CONFIG also declares `assetTypeBehavior`, the asset's type selects which merge policy applies to it.
- `trust_device_type`, `trust_os`, and `trust_os_version` (booleans) apply the script's `deviceType`, `os`, and `osVersion` values to the asset fingerprint even when runZero cannot normalize them through its fingerprint engine. The CONFIG-level `trustType`, `trustOS`, and `trustOSVersion` do the same for every asset the integration reports.

Leave `matchBehavior` unset to keep the default matcher behavior, which is right for most integrations. The asset identity guidance in the custom integrations repository covers the subject in more depth.

Testing scripts with the runZero CLI

The runZero CLI includes a `script` sub-command for developing and debugging integration scripts locally, using the same Starlark engine and modules as the Explorer.

```
runzero script [--filename file] [--args a] [--kwargs key=value]
runzero script repl [--filename file]
```

Flag	Description
`-f, --filename`	Script file to load and run. `--validate` also accepts a directory.
`--args`	Positional argument passed to `main` (repeatable).
`--kwargs key=value`	Keyword argument passed to `main` (repeatable).
`--validate`	Validate the script CONFIG and smoke-test HTTP/TLS wiring against a local dummy server.
`-o, --output`	Directory to write scan output to (`scan.runzero.gz`).
`--overwrite`	Replace the output directory if it already exists.
`--custom-integration-id`	Integration UUID stamped on exported records; required with `--output`.
`--starlark-allow-cidrs`	Restrict every network connection the script makes to the listed CIDRs or IPs (default deny). Enforced after DNS resolution, across the `http`, `requests`, `socket`, direct-protocol, and SQL modules.
`--starlark-block-cidrs`	Block script connections to the listed CIDRs or IPs (default allow). Same coverage as `--starlark-allow-cidrs`.

Running scripts

Save a minimal script as `hello.star`:

```
def main(*args, **kwargs):
    print("Hello {}!".format(kwargs.get("name", "world")))

runzero script --filename hello.star --kwargs name=Dave

INFO script: Hello Dave!
INFO script completed successfully with 0 returned asset(s)
```

Repeat `--args` and `--kwargs` to pass as many values as you need:

```
runzero script --filename hello.star --args one --args two --kwargs api_token=foo --kwargs url=https://api.example.com
```

A local run calls `main` with exactly the arguments you provide. The console and Explorer apply CONFIG defaults and validation at task time, and `--validate` applies them locally.

Validating scripts

`--validate` checks a script end-to-end without touching the real vendor API:

```
runzero script --filename example.star --validate

INFO example.star validated with 1 HTTP request(s)
INFO validated 1 script(s) with dummy HTTP/TLS server https://127.0.0.1:52084
```

For HTTP integrations, validation parses the CONFIG block, generates type-appropriate placeholder values for every parameter, initializes the script, calls `main`, and transparently routes every HTTP request to a local TLS server that returns canned responses. Validation fails if the script never makes an HTTP request, and if declared `OPTIONS_HTTP`/`OPTIONS_TLS` options never reach the HTTP client.

Scripts with `"validationMode": "compile"` (templates and direct-protocol integrations) stop after verifying that CONFIG parses and `main` exists.

Passing a directory validates every `.star` file under it:

```
runzero script --filename ./my-integrations/ --validate
```

A successful validation confirms the CONFIG and the wiring. It does not prove that the vendor API accepts your credentials or returns the expected payload; for that, test against the real API with `--kwargs`.

Exporting scan output

To inspect exactly what a script would import, write real scan output to a directory:

```
runzero script --filename example.star --kwargs api_token=MY_TOKEN -o ./out --overwrite --custom-integration-id 1111111
```

This writes `./out/scan.runzero.gz`, a gzip-compressed stream of JSON records (one per asset) in the same format an integration task produces. Without `--output`, assets streamed through `report_assets` are counted, then discarded with a warning.

REPL

`runzero script repl` starts an interactive session with all modules preloaded. With `--filename`, you can call the script's functions and globals directly:

```
$ runzero script repl --filename hello.star
>>> main(**{"name": "Dave"})
INFO script: Hello Dave!
>>> load('json', json_decode='decode')
>>> data = json_decode('{"greeting": "hello"}')
>>> print(data["greeting"])
INFO script: hello
```

Exit the REPL with `^D`.

Building and testing a script in the console

The custom integration page in the console is a script workbench: an editor with a test-credentials panel, a live output console, and, where it is enabled, an AI assistant that can draft or revise the script for you. You do not have to write a script in a local editor and paste it in; everything lives on the page itself, with no separate workspace to open or close.

The workbench is laid out like this:

- A pinned action bar that stays on screen while the page scrolls. Its left half holds the verdict row, one line stating the result of the last thing you did (validated, ran, saved, loaded, or generated; empty until it has something to report), and the example picker. Its right half holds **Revert**, **Validate**, **Run test** (which becomes **Stop** while a run is going), and **Full screen**.
- The editor, with a rail beside it carrying two tabs: **Test credentials** (the default) and **Generative AI**. Switching tabs never interrupts a test run or a generation. The editor has its own resize grabber for height and no drag dividers.
- The output console directly under the editor, a capped scroller that carries the detail: test-run logs and reported assets, per-line compile errors, and the progress and transcript of an AI generation.

Save with the page's **Save** (or **Update**) button, or with **Cmd/Ctrl+S** from anywhere in the workbench. A workbench save reports **Script changes saved** in the verdict row rather than in a toast. On a brand-new integration, Cmd/Ctrl+S creates the integration in place: the URL moves to the edit page without disturbing the editor, a generation, or a running test.

Full screen fills the window with the editor alone. **Esc** (or the **Exit full screen** button) leaves it, Cmd/Ctrl+S still saves, and the label shows **unsaved changes** whenever the editor differs from the last saved script. Full screen is unavailable while a generation is writing into the editor.

While a test run or a generation is in flight, the **Enable custom integration script** toggle refuses to turn off and says why. Turning the toggle off otherwise stashes the editor content locally; turning it back on restores it.

An empty editor shows a starter panel with three ways in: **Generate with AI**, **Load a blank template**, and **Browse examples**.

Starting from an example

The example picker in the action bar (**Select an example...**) loads a shipped integration into the editor as a starting point. These are the same scripts published in the runzero-custom-integrations repository, grouped as **Start from scratch** (a blank template with commented examples), **Import assets from a vendor**, and **Other automations**. The library ships through content updates, so it grows and refreshes between releases.

Loading an example replaces what is in the editor, so the picker asks first when you have something there. On an integration you have already saved, your previous script goes into version history before it is replaced, so you can always load it back. Loading also fills in an

empty integration name and description from the example, and picking the example already in the editor reports **No change**.

The integration remembers which example a script came from and at which version. When the shipped copy no longer matches yours, a banner says **A newer version of the *name* example is available**, notes when you have local modifications, and offers **Update to v_N_**. Updating asks first too, and snapshots what you have.

Generating a script with AI

Where generative AI is enabled for your account, the workbench rail has a **Generative AI** tab (an empty editor also offers **Generate with AI** directly). The tab holds a short form:

- **What should this integration do?** takes your description of the vendor, the API, and the data you want imported, in your own words. (When revising an existing script, the field asks **What should change?**) The **Authentication**, **Asset fields** and **Scope** buttons under the box insert the three questions the assistant most needs answered and cannot guess: how the API authenticates, which response fields are the address, MAC and hostname, and what to leave out.
- **Documentation links (optional)** takes one vendor API documentation URL per line, up to 10. The field says how many of the links you entered will be read.

Press **Generate** to write a new script, or **Update script** to revise the one already in the editor. The assistant works for several minutes: it consults the shipped integration library for the same vendor, reads the documentation you linked, and writes the script. Before it answers, the finished script must pass runZero's real validators. A script that fails goes back to the assistant to fix, shown as **Fixing validation errors**, and any `load()` lines the script forgot are added mechanically afterward.

Progress appears in the output console below the editor (the pages it reads, the tools it calls, and its own commentary), and the script itself streams into the editor as it is written. You can switch back to the **Test credentials** tab while it works; the run keeps going and keeps reporting in the verdict row.

Stop, in the AI form or beside the transcript in the output console, cancels the run. Nothing else does: an AI run is deliberately independent of the browser, so leaving the page or closing the tab leaves it running. Returning to the integration page picks the run back up, including anything that streamed while you were away. Every attempt also runs as a live turn on its own AI thread, with the running indicator, the work log, and its own Stop, so a run that failed or that you left behind is still there to look at. The output console links it as **View this request in AI Threads**.

If a run fails or you stop it before it finishes, whatever script was in the editor beforehand comes back. A run can also finish without a script (the assistant declining, asking a question, or answering in prose), and that answer appears under **What the model said**.

When it succeeds, the assistant's own account of what it built stays in the output console under **What the model says it built**, beside the code. **Clear** removes the transcript when you are done with it.

What the assistant is given, and what it is not

The model receives only three things: your description, the documentation URLs you supplied, and, when you are revising, the script currently in the editor. No inventory, asset, organization or credential data is sent, ever.

The assistant has four tools: it can fetch a documentation page, list and read runZero's shipped example integrations, and validate a candidate script. Validation only parses and compiles the script; nothing runs, no network call is made, and nothing is saved.

Documentation pages are fetched under the same protections as the rest of the platform: private and internal addresses are blocked, and the content is treated as untrusted throughout, because someone else wrote the page the assistant reads.

Read the script before you save or run it. It is drafted from third-party documentation, and it will run with your integration credentials.

Limits

- An authoring run is capped at 10 minutes.
- Documentation fetching within a run is capped at 10 pages and 90 seconds in total.
- A script sent for revision must be under 48 KiB. Above that the run still works, but it is generated from your description alone, and the form says so.
- Generative AI requires the AI feature to be enabled for the account and the organization, in addition to the custom integration script entitlement and administrator access (the same requirement as authoring a script by hand).
- Generations count against the same AI run limits as threads: 4 running AI jobs per user and 12 per account.

Where the generated script goes

When the run finishes, the script loads into the editor. If you have already saved the integration, it is committed straight away as a new version (the output console notes **Saved as version N**), so a generation is never one navigation away from being lost. A script that does not compile is not saved as a version; the output console says so and leaves it in the editor for you to fix. On an integration you have not saved yet, the output console reminds you to save the integration to keep the script.

Each run also produces an AI artifact holding the script. Artifacts start private to you and can be shared; an unshared artifact is retained as long as the thread it came from. Artifacts are listed under **Artifacts** in the AI section of the console, where **Use in custom integration** loads one back into the integration it was written for (or into a new one). The handoff lands directly on the workbench: the editor flashes, and the verdict row confirms what was loaded. For a new integration, the verdict row also suggests a name derived from the artifact's title, which you can change before saving.

Validating and previewing

Validate runs the same checks the save path runs: the size cap, the Starlark parse and compile, the CONFIG block extraction and schema, and the reserved-name rules. Nothing is

executed. Failures are reported per line in the output console, and the editor marks the line.

The **Credential fields** disclosure below the workbench lists every parameter the CONFIG block declares (label, key, type, whether it is required and secret, and its default), refreshed when the integration loads and after each save. It is the quickest way to check that a CONFIG block says what you meant. To see the rendered form itself, switch the test rail to **Enter values for this run only**, which shows the same fields the person creating the credential will see.

Running a test

Run test executes the script in the editor immediately, including changes you have not saved, and streams back its output, a preview of the assets it reported, and a plain-language verdict.

A test run is not a task, and differs from one in ways that matter:

- It runs on the console, not on an Explorer. Internal and link-local addresses are blocked, on self-hosted consoles too, so an integration that targets an endpoint inside your network cannot be tested this way. Test that one from an Explorer with the runZero CLI instead.
- Nothing is imported. No task is created, no assets reach your inventory, and nothing is written to your organization. The assets shown are a preview, discarded when the run ends.
- Credentials are not stored. You can run against a saved credential, or switch the rail to **Enter values for this run only**. Values you type in for a run are held for the length of that run and never saved; to schedule an integration whose CONFIG requires values, you still need a saved credential.

Picking a saved credential shows its stored values in the rail, so the choice is not blind: non-secret values appear verbatim, and secret values show only that a value is stored, because they are never sent to the browser. Your non-secret selections (credential mode, credential choice, non-secret parameter values) are remembered per integration in the browser between visits; secret values start empty every time.

Before anything executes, the run checks the credential values against the script's CONFIG schema; values that can never satisfy it are refused up front, naming the parameter keys (never the values). The rail also warns beside **Run test** when no credential is selected and the CONFIG declares required fields.

The run redacts credential values from everything it prints, including error text, crash output, and the asset preview. That covers every value supplied to the run, saved or typed in, and every CONFIG `secret` value.

Test run limits

Limit	Value
Wall clock	120 seconds
Memory	256 MiB
Computation	2,000,000,000 interpreter steps; time spent waiting on an API does not count

Assets	100, after which the run stops and reports success
Assets shown in full	the first 10, with up to 24 attributes each, values truncated at 256 bytes
Log output	500 lines, 4 KiB per line

You can run one test at a time, up to six runs a minute; the account as a whole is capped at twenty starts a minute.

Reaching the 100-asset ceiling is reported as a **success**: the script worked and was stopped early because this is a test. A real task is not capped this way.

Every run ends with a named outcome rather than a bare failure: it completed, it stopped at the asset limit, you stopped it, the script raised an error, it ran out of time, memory, computation or stack, the sandbox stopped it, it stopped unexpectedly, or it never started. Each comes with a short explanation of what to look at next. A run that completes without reporting any assets says so; it usually means the response was parsed differently than expected, or a filter in the script excluded everything.

A run that could not be started at all is explained the same way, including the compiler's own line and column when the script did not compile, so you do not have to press **Validate** to find out what was wrong.

Version history

Every change to a script is snapshotted, including versions committed by an AI generation. The **Script version history** disclosure on the custom integration page (its title carries the count and the current version) opens a table listing each version with its author, timestamp, size, and the example it came from, if any. runZero keeps the most recent 25 versions.

Load, offered on every version other than the current one, puts a stored version into the editor so you can look at it or work from it. It asks first when the editor holds something else. Nothing is written until you save, and saving creates a new version rather than rewriting history.

Revert appears in the action bar whenever the editor differs from the last saved script, and restores that saved script locally. Like Load, it only changes the editor; it asks first, because the editor may be the only copy of what you have typed.

Adding the integration to your console

Step 1: Create the custom integration

1. Go to the Custom sources page and click **Add custom integration**.
2. Enter a name (no spaces) and, optionally, an icon (PNG, up to 256×256).
3. Choose whether the integration is available to every organization in the account or only to specific organizations. See organization access.
4. Toggle **Enable custom integration script**, then paste in your script, start from an example, or generate one with AI.

5. Click **Validate** to check the script syntax and CONFIG block, and optionally run the script against a real endpoint before you schedule it.
6. Click **Save**, or **Save & create task**, which saves and lands on the task form with the integration pre-selected.

Step 2: Create the credential

1. Go to the Credentials page and click **Add Credential**.
2. Choose **Custom Integration Script Secrets** as the credential type.
3. Select the custom integration. The form shows the fields declared in the script's `CONFIG` block, including any option-suite fields such as `tls_disable_validation` and `http_user_agent`.
4. For legacy scripts without a CONFIG block, provide the **Access key (legacy)** and **Access secret (legacy)** values, which reach the script as the `access_key` and `access_secret` kwargs.
5. To let other organizations use this credential, select the *Make this a global credential* option.
6. Save the credential.

Step 3: Create the task

1. Go to the custom integration task page, click **Integrate** on the Tasks page and choose **Custom Scripts**, or use one of the shortcuts that pre-select the integration: **Save & create task** on the integration page, or the play action on its row of the Custom sources grid.
2. Select the custom integration and the credential you created above. Saving the task validates the pair: the script must compile, and the credential must satisfy every parameter the CONFIG block requires. A save with no credential, or with a credential missing required parameters, fails with an error naming the parameter keys. (Legacy scripts without a CONFIG block skip this check.)
3. Select the Explorer to run the script from, or leave the selection empty to run the task on the console itself. Console-hosted tasks cannot reach private or link-local addresses, and on runZero-hosted consoles they run under a tighter budget, so use an Explorer for anything that targets your internal network.
4. Set the site, description, and schedule.
5. Activate the connection to start the task.

Once the task completes, assets appear in your inventory, and the search `custom_integration:<name>` finds them. Tasks list the integration with its own icon, and the task search `custom_integration:` accepts a name or a UUID, including the UUID of a deleted integration.

Organization access

A custom integration is available either to every organization in the account or only to the organizations it was granted to. Custom integrations that existed before this option was added are available to every organization, so their tasks keep working until an administrator narrows an integration's scope.

Managing custom integrations requires administrator-level permissions. Users with **Administrator** as their default role can manage every custom integration. Users with per-

organization permissions see only the integrations available to their organizations, and can edit an integration only if they hold administrator permissions in every organization it is available to. Only a superuser or a user whose default role is Administrator can edit an integration that is available to every organization.

Reading an integration's script and its configuration parameters requires administrator permissions in one of the organizations the integration was granted to; the script of an integration available to every organization is visible only to superusers and users whose default role is Administrator. Users, viewers and annotators can see that an integration exists, including its name and icon, so the asset data it collected stays readable, but they cannot see its script or its configuration parameters. Selecting an integration when configuring a task requires at least user-level (write) access in an organization the integration is available to; viewers and annotators cannot.

An integration task can only use a custom integration that is available to the organization the task belongs to. runZero checks access again each time the task runs, so removing an organization's access also stops tasks that were already scheduled, with an error naming the integration and organization. Deleting an organization removes it from every integration's access list; an integration that was scoped only to that organization is kept, but is visible only to superusers and account-wide administrators until it is re-scoped.

The account API exposes the same organization scope: custom integration objects carry `global` and `organizationIds` fields (mutually exclusive), and organization-scoped API tokens see only integrations shared with their organizations.

Deleting a custom integration

Deleting a custom integration requires typing the word `delete` to confirm, and the confirmation states the impact: the saved script versions are deleted, the data the integration added to assets is removed, and scheduled tasks that use it are **paused** rather than deleted. Each paused task shows the reason, and you can re-point it at another integration. Credentials are kept, and another integration can use them.

Script limits and sandbox behavior

- Scripts are limited to 1 MiB of source. A script sent to the AI assistant for revision has a much smaller ceiling of 48 KiB.
- Scripts can only `load()` the registered modules, with no filesystem access and no relative imports.
- A script run has an execution-step ceiling and a wall clock, and the deadline also bounds HTTP requests and `time.sleep` calls. A task on an Explorer or a self-hosted console gets 24 hours and 720 billion steps; one on a runZero-hosted console gets 2 hours and 240 billion steps. A console test run is much tighter again, at 120 seconds. Only computation consumes steps, so time spent waiting on an API does not count against the ceiling. Self-hosted operators can override both budgets with the `RUNZERO_CUSTOM_INTEGRATION_MAX_STEPS` and `RUNZERO_CUSTOM_INTEGRATION_MAX_RUN_SECONDS` environment variables.
- HTTP response bodies are capped at 1 GiB, and a script's cumulative network reads across `http`, `requests`, `socket`, SSH, and SMB are capped at 4 GiB per run. Other per-call

caps include 64 MiB per SMB file read, 16 MiB per SSH command output stream, and gzip decompression bounded at 1 GiB and a 500:1 expansion ratio (see the library reference).
- A script can hold at most 256 open connections and sessions (sockets, SSH, SMB, WinRM, WMI, SQL) at once; closing one frees its slot.
- runZero redacts the values of `secret` parameters from `print` output, progress messages, and error text.
- On an Explorer task, every network module (`http`, `requests`, `socket`, and the direct-protocol modules `runzero.ssh`, `runzero.smb`, `runzero.winrm`, `runzero.wmi`, and `runzero.sql`) can reach the internal addresses visible to that Explorer. On console-hosted runs (both test runs and tasks run without an Explorer), private and link-local addresses are blocked for every module. The CLI's `--starlark-allow-cidrs` and `--starlark-block-cidrs` flags pin script egress for local runs.

Available libraries

Load only what you use; each module comes in through `load(...)`. The Starlark libraries page has full signatures and runnable examples for every module.

Module	Provides
`runzero.types`	`ImportAsset`, `NetworkInterface`, `Service`, `ServiceProtocolData`, `Software`, `Vulnerability`, `to_custom_attributes`
`kwargs`	Typed accessors: `require`, `get_string`, `get_bool`, `get_int`, `get_float`, `get_list`, `get_url_base`, `get_http_tls`, `get_http_options`
`coerce`	Total (never-raising) conversions for messy API data: `as_text`, `as_dict`, `as_list`, `dicts`, `as_int`, `as_float`, `as_bool`, `dedupe`
`http`	HTTP verbs, `get_json`/`post_json` with retries, `bearer`/`basic`/`oauth2_token`, `url_encode`/`url_parse`/`url_join`, `multipart`
`requests`	Stateful HTTP `Session` with sticky headers and cookies
`net`	`ip_address`, `network_interface`, `normalize_mac`, `ip_network`, `ip_in_network`, `resolve`, plus identity screens `routable_ip`, `routable_ips`, `clean_hostname`, `clean_hostnames`, `mac_key`
`json`	`encode`, `decode`, `encode_indent`, `indent`
`jsonstream`	`iter_array`, `iter_lines` for large JSON/NDJSON responses
`csv`	`read_all`, `read_rows`, `write_all`, `write_dicts`
`xml`	`parse` into an element tree
`re`	RE2 regular expressions: `match`, `find_all`, `sub`, `split`, `compile`
`time`	`now`, `parse_time`, `parse_ts`, `parse_duration`, `from_timestamp`, `sleep`
`uuid`	`new_uuid`
`base64` / `hex` / `base32`	Standard encodings, including raw and URL-safe variants

`crypto`	Hashes, HMAC, AWS SigV4 signing, CSPRNG output
`jwt`	`encode`, `decode`, `decode_unverified`
`gzip`	`compress`, `decompress`
`flatten_json`	`flatten` nested structures
`runzero.progress`	`report`, `info`, `warn` task progress in the console
`socket`	Raw TCP/UDP/TLS connections
`runzero.ssh` / `runzero.smb` / `runzero.winrm` / `runzero.wmi` / `runzero.sql`	Direct-protocol collection from systems without a REST API

The predeclared names `report_assets`, `report_asset`, `pager`, `max_pages`, `OPTIONS_TLS`, and `OPTIONS_HTTP` are always available without a `load()`.

Existing custom integrations

Name	Setup Instructions	Integration Code
Absolute Secure Endpoint	Link	Link
AdGuard Home	Link	Link
Akamai Guardicore Centra	Link	Link
Armis Centrix	Link	Link
Asimily	Link	Link
Audit Log to Webhook	Link	Link
Automox	Link	Link
BMC Helix Discovery	Link	Link
Bitdefender GravityZone	Link	Link
Bitsight	Link	Link
Carbon Black	Link	Link
Checkmk Raw Edition	Link	Link
Cisco Cyber Vision	Link	Link
Cisco ISE	Link	Link
Cisco Secure Endpoint	Link	Link
Claroty CTD	Link	Link
Claroty xDome	Link	Link

Cortex XDR	Link	Link
CyberArk EPM	Link	Link
Cybereason	Link	Link
Cyberint	Link	Link
Cyberwatch	Link	Link
Device42	Link	Link
Digital Ocean	Link	Link
Drata	Link	Link
EfficientIP SOLIDserver	Link	Link
ExtraHop Reveal(x)	Link	Link
Extreme Networks CloudIQ	Link	Link
Fleet (osquery)	Link	Link
Foreman	Link	Link
Forescout CounterACT	Link	Link
Forescout Risk and Exposure Management	Link	Link
Forescout eyeInspect	Link	Link
Frontline VM	Link	Link
GLPI	Link	Link
Ghost Security	Link	Link
Greenbone (GMP) Import	Link	Link
Greenbone (GMP) Scan Launch	Link	Link
HCL BigFix	Link	Link
HPE Aruba ClearPass	Link	Link
Halcyon	Link	Link
Home Assistant	Link	Link
Icinga 2	Link	Link
Illumio Core	Link	Link
Infoblox NIOS	Link	Link
Ivanti Neurons	Link	Link
JAMF	Link	Link
JumpCloud	Link	Link

Kandji	Link	Link
Kenna Security	Link	Link
Kubernetes	Link	Link
Lansweeper	Link	Link
LibreNMS	Link	Link
LimaCharlie	Link	Link
Linux via SSH	Link	Link
ManageEngine Endpoint Central	Link	Link
Maze	Link	Link
Microsoft Defender for IoT	Link	Link
Microsoft SQL Server databases	Link	Link
Microsoft WSUS	Link	Link
MikroTik RouterOS	Link	Link
Miradore	Link	Link
Mosyle	Link	Link
Nautobot	Link	Link
Netdata	Link	Link
Netdisco	Link	Link
Netskope	Link	Link
Nexthink	Link	Link
NinjaOne	Link	Link
Nozomi Networks	Link	Link
Nutanix Prism	Link	Link
OCS Inventory NG	Link	Link
OPNsense	Link	Link
Open-AudIT Community	Link	Link
OpenNMS Horizon	Link	Link
OpenWrt	Link	Link
Palo Alto Networks Device Security	Link	Link
Pi-hole	Link	Link
Portainer / Docker Engine	Link	Link

Proxmox	Link	Link
PuppetDB	Link	Link
Quest KACE SMA	Link	Link
Red Hat Insights	Link	Link
SAP LeanIX	Link	Link
Scale Computing	Link	Link
Scan Passive Assets	Link	Link
Slurp'it	Link	Link
Snipe-IT	Link	Link
Snow License Manager	Link	Link
SolarWinds Information Service	Link	Link
Sophos Central	Link	Link
Stairwell	Link	Link
Sumo Logic	Link	Link
Synology DSM	Link	Link
TP-Link Omada	Link	Link
Tactical RMM	Link	Link
Tailscale	Link	Link
Tenable OT Security	Link	Link
Trellix ePolicy Orchestrator	Link	Link
Trend Micro Vision One	Link	Link
TrueNAS	Link	Link
Ubiquiti UniFi Network	Link	Link
Ubiquiti UniFi Protect	Link	Link
Ubiquiti UniFi Site Manager	Link	Link
Unraid	Link	Link
Uptycs	Link	Link
Vulnerability Workflow	Link	Link
Wazuh	Link	Link
Windows SMB shares	Link	Link
Windows WMI	Link	Link

Workspace ONE UEM	Link	Link
Zabbix	Link	Link
exe.dev	Link	Link
iTop	Link	Link
ntopng	Link	Link
pfSense	Link	Link
phpIPAM	Link	Link
runZero Task Sync	Link	Link

Starlark libraries

This page is the reference for the Starlark modules available to custom integration scripts, with a runnable example for each. Save any example as a `.star` file and run it with the runZero CLI:

```
runzero script --filename example.star
```

Every module comes in through `load(...)`; load only the names you use. The names `report_assets`, `report_asset`, `pager`, `max_pages`, `OPTIONS_TLS`, and `OPTIONS_HTTP` are predeclared and never need a `load()`.

```
load("runzero.types", "ImportAsset", "NetworkInterface", "Service",
                      "ServiceProtocolData", "Software", "Vulnerability",
                      "to_custom_attributes")
load("kwargs", "require", "has", "get", "get_string", "get_bool", "get_int",
               "get_float", "get_list", "get_url_base", "get_http_tls",
               "get_http_options")
load("http", http_get="get", http_post="post", "head", "put", "patch", "delete",
             "get_json", "post_json", "url_encode", "url_parse", "url_join",
             "multipart", "bearer", "basic", "oauth2_token")
load("requests", "Session", "Cookie")
load("net", "ip_address", "network_interface", "normalize_mac",
            "ip_network", "ip_in_network", "resolve",
            "routable_ip", "routable_ips", "clean_hostname", "clean_hostnames",
            "mac_key")
load("coerce", "as_text", "as_dict", "as_list", "dicts",
               "as_int", "as_float", "as_bool", "dedupe")
load("json", json_encode="encode", json_decode="decode")
load("jsonstream", "iter_array", "iter_lines")
load("csv", csv_read="read_all", csv_write="write_dicts")
load("xml", xml_parse="parse")
load("re", re_match="match", re_find_all="find_all", re_sub="sub")
load("time", "now", "parse_time", "parse_ts", "parse_duration", "sleep")
load("uuid", "new_uuid")
load("base64", base64_encode="encode", base64_decode="decode")
load("hex", hex_encode="encode", hex_decode="decode")
load("base32", b32_encode="encode", b32_decode="decode")
```

```
load("crypto", "sha256", "hmac_sha256", "sign_v4", "random_hex")
load("jwt", jwt_encode="encode", jwt_decode="decode")
load("gzip", gzip_compress="compress", gzip_decompress="decompress")
load("flatten_json", "flatten")
load("runzero.progress", progress_report="report", progress_info="info")
load("socket", "tcp", "udp", "tls")
load("runzero.ssh", ssh_dial="dial")
load("runzero.smb", smb_dial="dial")
load("runzero.winrm", winrm_dial="dial")
load("runzero.wmi", wmi_dial="dial")
load("runzero.sql", sql_connect="connect")
```

runzero.types

Constructors for the objects a script imports. Every keyword argument accepts both camelCase (`osVersion`) and snake_case (`os_version`) spellings.

- `ImportAsset(id, ...)`: `id` is required and must be a stable, unique string (up to 1024 characters). Optional fields: `hostnames`, `domain`, `os`, `osVersion`, `manufacturer`, `model`, `deviceType`, `assetType`, `tags`, `firstSeenTS`, `lastSeenTS`, `networkInterfaces`, `services`, `software`, `vulnerabilities`, `customAttributes`, `runZeroID`, `trust_device_type`, `trust_os`, `trust_os_version`. Merge policy is no longer set per asset: declare `matchBehavior` in the CONFIG block instead. Passing it to `ImportAsset` is an error.
- `NetworkInterface(macAddress=None, ipv4Addresses=[], ipv6Addresses=[])`: build one directly, or with `net.network_interface`, which handles mixed and messy input.
- `Service(address, port, transport, ...)`: the first three are required; `vendor`, `product`, `version`, `protocolData`, and `customAttributes` are optional.
- `ServiceProtocolData(name, attributes={})`: protocol-specific details attached to a `Service`.
- `Software(id, ...)`: optional `vendor`, `product`, `version`, `update`, `cpe23`, `serviceAddress`, `serviceTransport`, `servicePort`, `installedAt`, `installedSize`, `installedFrom`, and more.
- `Vulnerability(id, ...)`: optional `name`, `description`, `solution`, `cve` (must match `CVE-YYYY-NNNN`), `cpe23`, `cvss2BaseScore`, `cvss3BaseScore`, `severityRank`/`riskRank` (0=Info through 4=Critical), `severityScore`/`riskScore`, `exploitable`, service linkage, and timestamps.
- `to_custom_attributes(value, ...)`: coerces arbitrary nested data into the `string -> string` map required by `customAttributes`. Nested dicts are flattened with `separator` (default `"."`), lists are joined with `list_join` (default `","`), empty values are dropped unless `drop_empty=False`, and `prefix`, `exclude`, `max_key`, `max_value`, and `max_entries` give finer control. The platform caps custom attributes at 1024 entries, with keys up to 256 and values up to 1024 characters.

Each asset holds up to 256 network interfaces, 100 hostnames (260 characters each), and 100 tags. `hostnames` and `tags` silently drop empty strings, but a `None` entry in `hostnames` is an error that aborts the run, so `hostnames=[device.get("hostname")]` fails whenever the field is absent. Pass hostname fields through `net.clean_hostnames`, which also screens out placeholder names.

```
load("runzero.types", "ImportAsset", "Service", "ServiceProtocolData",
                      "Software", "Vulnerability", "to_custom_attributes")
load("net", "network_interface")

def main(*args, **kwargs):
    nic = network_interface(
```

```
ips=["192.0.2.5", "fe80::1%eth0", "[2001:db8::1]:443"],
    )
    svc = Service(
        address="192.0.2.5",
        port=443,
        transport="tcp",
        vendor="nginx",
        product="nginx",
        version="1.25.3",
        protocolData=[
            ServiceProtocolData(name="http", attributes={"server": "nginx"}),
            ServiceProtocolData(name="tls", attributes={"subject": "CN=acme"}),
        ],
    )
    asset = ImportAsset(
        id="device-123",
        hostnames=["web1.acme.local"],
        os="Linux",
        osVersion="6.1",
        deviceType="Server",
        manufacturer="Acme",
        model="Rack-42",
        tags=["prod", "web"],
        networkInterfaces=[nic],
        services=[svc],
        software=[Software(id="pkg-1", vendor="F5", product="NGINX", version="1.25.3",
                           serviceAddress="192.0.2.5")],
        vulnerabilities=[Vulnerability(id="vuln-1", name="Example finding",
                                       cve="CVE-2023-0001", severityRank=3,
                                       severityScore=8.1, riskRank=3, riskScore=8.1)],
        customAttributes=to_custom_attributes({
            "name": "web1",
            "labels": ["prod", "edge"],       # joined with ","
            "sys": {"os": "linux", "ver": 5}, # flattened with "."
            "empty": "",                      # dropped
        }),
    )
    return [asset]
```

kwargs

Typed, validating accessors over the `**kwargs` dict passed to `main`. Values arrive as strings; these helpers coerce them and apply defaults.

- `require(kwargs, *keys)`: an error if any key is missing or blank.
- `has(kwargs, key)`: `True` when the key is present and non-empty.
- `get(kwargs, key, default="")` / `get_string(...)`: aliases.
- `get_bool(kwargs, key, default=False)`: accepts `true`/`false`, `1`/`0`, `yes`/`no`, `y`/`n`, `on`/`off`.
- `get_int(kwargs, key, default=0)` / `get_float(kwargs, key, default=0.0)`
- `get_list(kwargs, key, default=[], sep=",")`: splits a delimited string, or passes a list through.
- `get_url_base(kwargs, key="url", default="")`: extracts scheme + host from a URL kwarg, dropping any path or query.
- `get_http_tls(kwargs, prefix="tls_")`: collects the `OPTIONS_TLS` include into a `tls=` dict.

- `get_http_options(kwargs, prefix="http_", tls_prefix="tls_", headers=None)`: collects the `OPTIONS_HTTP` and `OPTIONS_TLS` includes into a dict of keyword arguments for any `http` function.

```
load("kwargs", "require", "has", "get_string", "get_bool", "get_int", "get_list")

def main(*args, **kwargs):
    require(kwargs, "client_id", "client_secret")
    client_id = get_string(kwargs, "client_id")
    page_size = get_int(kwargs, "page_size", default=100)
    include_offline = get_bool(kwargs, "include_offline", default=False)
    regions = get_list(kwargs, "regions", default=[])
    print(client_id, page_size, include_offline, regions)
    if has(kwargs, "region"):
        print("region override:", get_string(kwargs, "region"))

runzero script --filename example.star --kwargs client_id=abc --kwargs client_secret=xyz --kwargs include_offline=yes -
```

coerce

Total conversions for messy API data: nothing in this module raises, so a malformed or missing field degrades to a default instead of aborting the import.

- `as_text(value, default="", join="")`: any value to a string; lists are joined with `join` when set.
- `as_dict(value)`: a dict, or `{}` for anything else.
- `as_list(value, wrap=True)`: a list; scalars are wrapped in a one-element list unless `wrap=False`.
- `dicts(value)`: the dict elements of a list, dropping everything else.
- `as_int(value, default=0)` / `as_float(value, default=0.0)`: numbers from ints, floats, or numeric strings.
- `as_bool(value, default=False)`: the usual true/false spellings.
- `dedupe(values, fold_case=False)`: order-preserving deduplication.

```
load("coerce", "as_text", "as_int", "as_list", "dicts")

def main(*args, **kwargs):
    device = {"name": None, "port": "8443", "ips": "192.0.2.5",
              "disks": [{"id": 1}, "oops", {"id": 2}]}
    print("name:", as_text(device["name"], default="unknown"))
    print("port:", as_int(device["port"]))
    print("ips:", as_list(device["ips"]))    # a scalar becomes a one-item list
    print("disks:", dicts(device["disks"]))  # non-dict entries dropped
```

http

Stateless HTTP requests, JSON convenience wrappers, auth helpers, and URL utilities.

`get(url, headers=None, params=None, timeout=60, insecure_skip_verify=False, tls=None)` and `head(...)` take the same arguments; `post`, `put`, `patch`, and `delete` also accept `body=` (bytes/string) or `json=` (auto-encoded dict). All verbs default to a 60-second timeout, and `params=` replaces any query

string already present in the URL. Each returns a response struct with `status_code` (int), `status` (string), `headers` (values are lists of strings), and `body`.

`get_json` and `post_json` wrap the common fetch-check-decode pattern in one call and return a `(data, err)` tuple. `err` is `None` on success, or a short string: `"status 401: ..."` for an HTTP failure, or the transport error text for a connection failure. A 2xx response with an empty body decodes to `None`. Both also accept `insecure_skip_verify=` and `tls=`. Transient failures are retried by default: `retries` defaults to 3, with exponential backoff (`retry_backoff`, default 1 second, doubling up to `retry_max_backoff`, default 30 seconds) on transient statuses (`408, 425, 429, 500, 502, 503, 504` by default; override with `retry_on=[...]`), honoring `Retry-After` headers. Pass `retries=0` to opt out, and do so for non-idempotent writes.

`bearer(token)` and `basic(username, password)` format `Authorization` header values. `oauth2_token(token_url, client_id, client_secret, scope=None, audience=None, grant_type="client_credentials", extra=None, headers=None, timeout=60, insecure_skip_verify=False, tls=None)` performs an OAuth token exchange and returns the access token string.

`url_encode(params)` builds a query string; `url_parse(url)` returns a struct (`scheme`, `host`, `hostname`, `port`, `path`, `query`, `fragment`, ...) or `None`; `url_join(base, ref)` resolves a relative reference (useful for pagination `next` links); and `multipart(fields)` builds a `multipart/form-data` body and returns `(body_bytes, content_type)`.

```
load("http", "get_json", "url_parse", "url_join", "bearer")

def main(*args, **kwargs):
    url = "https://hacker-news.firebaseio.com/v0/topstories.json"
    data, err = get_json(url, headers={"Accept": "application/json"}, retries=2)
    if err:
        print("fetch failed:", err)
        return None
    print("top story IDs:", data[:5])

    u = url_parse(url)
    print("host:", u.host, "path:", u.path)
    print("next page:", url_join(url, "/v0/beststories.json"))
    print("auth header:", bearer("example-token"))
```

In an integration, collect the option-suite settings once and splat them into every call:

```
http_options = get_http_options(kwargs, headers={"Authorization": bearer(token)})
data, err = get_json(url, params={"limit": 100}, **http_options)
```

The `tls=` argument accepts a dict with the keys `insecure`, `server_name`, `ca_pem`, `client_cert_pem`, `client_key_pem`, and `thumbprints` (SHA-256 pins); unknown keys are an error. Use the raw verbs instead of `get_json` when you need response headers, cookies, or the status code directly.

Where an egress filter applies (console-hosted runs, or the CLI's `--starlark-allow-cidrs`/`--starlark-block-cidrs` flags), a request to a refused address aborts the script rather than returning an `err` tuple.

requests

A stateful HTTP `Session` with sticky headers and a cookie jar. Verbs (`get`, `post`, `put`, `patch`, `delete`, `head`) accept `headers`, `cookies`, `params`, `body`, `json`, and `timeout` (default 60 seconds), and return the same response struct as the `http` module.

`Session` does not accept the `tls=` dict; only `Session(insecure_skip_verify=True)` is available. Prefer the `http` module when the TLS option suite matters.

```
load("requests", "Session", "Cookie")
load("json", json_decode="decode")

def main(*args, **kwargs):
    session = Session()
    session.headers.set("Accept", "application/json")
    session.headers.set("User-Agent", "runZero-Example/1.0")

    url = "https://hacker-news.firebaseio.com/v0/topstories.json"
    response = session.get(url)
    if response.status_code == 200:
        data = json_decode(response.body)
        print("top story IDs:", data[:5])
    else:
        print("request failed with status:", response.status_code)
```

`session.headers` exposes `get(key)` and `set(key, value)` (`value=None` deletes a header). `session.cookies` exposes `get(url)`, `set(url, cookies)`, and `clear()`; `Cookie(name, value, path="", domain="", secure=False, ...)` builds a full cookie object for `cookies.set`.

net

IP, MAC, and DNS helpers, plus identity screens that keep placeholder values out of asset matching.

- `ip_address(s)`: validates one address; the result exposes `.version` (4 or 6) and stringifies to canonical form.
- `network_interface(mac=None, ips=None, ipv4=None, ipv6=None)`: builds a `NetworkInterface` from messy input. It handles mixed v4/v6 lists, `addr:port` and `[addr]:port` suffixes, and `%zone` IDs, drops duplicates, and keeps up to 99 addresses per family. It returns `None` when nothing usable remains, so `if nic:` guards work.
- `normalize_mac(s, preserve_bits=True)`: canonical lowercase colon form from any common format (colons, dashes, Cisco dotted, bare hex); returns `None` for unparseable input. Pass `preserve_bits=False` to clear the locally-administered bit, the historical matching form; `network_interface` stores the asset MAC with the bit cleared regardless.
- `mac_key(s)`: `normalize_mac` plus rejection of values that cannot identify a device. All-zero and broadcast addresses and synthetic `ip-<addr>` placeholders return `None`.
- `routable_ip(value, exclude=None)` / `routable_ips(values, exclude=None)`: the canonical identity-usable address (or `None`), rejecting loopback, unspecified, link-local, multicast, and broadcast addresses while keeping RFC1918, CGNAT, and ULA space. Accepts

bracketed, `host:port`, `%zone`, and CIDR-suffixed input. `exclude=[...]` overrides the exclusion list; `exclude=[]` disables it.
- `clean_hostname(value, extra=None, max_length=253)` / `clean_hostnames(values, ...)`: a usable hostname or `None`, rejecting placeholders (`localhost`, `unknown`, `none`, `-`, `n/a`), IP-shaped values, all-numeric names, over-long names, and illegal DNS characters. `extra=[...]` adds source-specific placeholders; the plural form also deduplicates case-insensitively.
- `ip_network(cidr)`: a struct with `cidr`, `version`, `prefix`, `network`, `broadcast`, `netmask`, and a `contains(ip)` method.
- `ip_in_network(ip, cidr)`: a one-shot membership check; `False` for malformed or mixed-family input.
- `resolve(host, timeout=10)`: an A/AAAA lookup returning a list of IP addresses; returns an empty list (never an error) for unresolvable input.

```
load("net", "ip_address", "network_interface", "normalize_mac",
            "ip_network", "ip_in_network", "resolve",
            "routable_ip", "clean_hostnames", "mac_key")

def main(*args, **kwargs):
    addr = ip_address("192.0.2.5")
    print("ip:", addr, "version:", addr.version)
    print("mac:", normalize_mac("9CB6.D0F1.D204"))

    net10 = ip_network("10.0.0.0/8")
    print("contains:", net10.contains("10.1.2.3"))
    print("in network:", ip_in_network("10.1.2.3", "10.0.0.0/8"))

    for ip in resolve("localhost"):
        print("resolved:", ip, ip.version)

    nic = network_interface(mac="9c-b6-d0-f1-d2-04",
                            ips=["192.0.2.5", "fe80::1%eth0", "[2001:db8::1]:443"])
    print("nic:", nic)

    print("routable:", routable_ip("127.0.0.1"), routable_ip("10.1.2.3:8443"))
    print("hostnames:", clean_hostnames(["web1.acme.local", "localhost", None]))
    print("mac key:", mac_key("00:00:00:00:00:00"))  # None: not an identity
```

json

```
load("json", json_encode="encode", json_decode="decode")

def main(*args, **kwargs):
    data = {"name": "runZero", "features": ["scan", "API", "integrations"]}
    encoded = json_encode(data)
    print("encoded:", encoded)
    decoded = json_decode(encoded)
    print("decoded name:", decoded["name"])
```

The module also provides `encode_indent` and `indent` for pretty-printed output.

jsonstream

Stream large JSON documents without materializing them. `iter_array(body, path=None)` iterates the elements of an array (optionally at a dot-separated path inside the document); `iter_lines(body)` iterates NDJSON / JSON-lines input. Both accept strings or bytes; pass a response `body` directly.

```
load("jsonstream", "iter_array", "iter_lines")

def main(*args, **kwargs):
    big_json = '{"data": {"items": [{"id": 1}, {"id": 2}]}}'
    for item in iter_array(big_json, path="data.items"):
        print("streamed id:", item["id"])

    for line in iter_lines('{"a": 1}\n{"a": 2}'):
        print("ndjson:", line["a"])
```

csv

`read_all(text, delimiter=",", comment="", header=True)` returns a list of dicts keyed by the header row (or a list of lists with `header=False`); `read_rows` is the header-less form. `write_all(rows)` and `write_dicts(rows, fields=None)` serialize back to CSV strings.

```
load("csv", csv_read="read_all", csv_write="write_dicts")

def main(*args, **kwargs):
    rows = csv_read("id,name\n1,web1\n2,web2\n")
    print("first row:", rows[0]["name"])
    print("as csv:", csv_write(rows))
```

xml

`parse(text)` returns an element tree. Elements expose `tag`, `text`, `tail`, `attrib`, and `children`, plus the methods `find(path)`, `find_all(path)`, `get(name, default="")`, and `text_all()`.

```
load("xml", xml_parse="parse")

def main(*args, **kwargs):
    doc = xml_parse("<devices><device><name>web1</name></device></devices>")
    print("name:", doc.find("device/name").text)
    print("count:", len(doc.find_all("device")))
```

re

Regular expressions using Go RE2 syntax. `match(pattern, string)` and `search(...)` return a struct (`.match`, `.start`, `.end`, `.groups`, and `.named` for named groups) or `None`. Also available: `find_all`, `find_all_groups`, `sub(pattern, repl, string, count=-1)`, `split`, `escape`, and `compile(pattern)` for a reusable pattern object.

```
load("re", re_match="match", re_find_all="find_all", re_sub="sub")

def main(*args, **kwargs):
    matches = re_find_all(r"id=(\d+)", "id=10 id=20")
    print("matches:", matches)
    print("cleaned:", re_sub(r"\s+", " ", "a   b\tc"))
    m = re_match(r"(?P<major>\d+)\.(?P<minor>\d+)", "5.1")
    print("named groups:", m.named)
```

time

The standard Starlark `time` module plus a sandbox-aware `sleep`.

- `parse_time(s)`: parses RFC 3339 strings into a `time` value with `year`, `month`, `day`, `hour`, `minute`, `second`, `unix`, and `unix_nano` fields, plus `format(layout)` and `in_location(name)` methods. It raises on input it does not recognize, and a raise from a builtin aborts the whole script, so use it for literals and `parse_ts` for API-supplied values.
- `parse_ts(value, default=None, assume_utc=True, clamp_to_now=True, unit="s")`: a never-raising timestamp parser for values a vendor API returns. It accepts epoch ints, floats, and numeric strings (`unit` selects `s`/`ms`/`us`/`ns`) plus a wide set of datetime layouts, and reads zone-less values as UTC. Non-positive epochs yield `default`, and future values are clamped to now, because the platform drops an `ImportAsset` whose first- or last-seen timestamp is ahead of the clock.
- `parse_duration(s)`: parses `"90m"`, `"1h30m"`, `"250ms"`, and similar into a `duration` with `hours`, `minutes`, `seconds`, and related fields. Time and duration values support arithmetic.
- `now()`: the current time. `from_timestamp(sec)` converts a Unix timestamp.
- `sleep(duration)`: accepts a duration string or value, and respects the task deadline.

```
load("time", "now", "parse_time", "parse_ts", "parse_duration", "sleep")

def main(*args, **kwargs):
    t = parse_time("2026-08-14T15:00:00Z")
    print("unix:", t.unix, "year:", t.year)
    print("formatted:", t.format("2006-01-02"))

    seen = parse_ts("2026-08-14 15:00:00")   # zone-less, read as UTC
    print("api time:", seen)
    print("bad input:", parse_ts("not-a-date"))  # None, never a crash

    d = parse_duration("90m")
    print("minutes:", d.minutes)
    print("90 minutes from now:", now() + d)
    sleep("250ms")
```

uuid

```
load("uuid", "new_uuid")

def main(*args, **kwargs):
    print("generated UUID:", new_uuid())
```

base64, hex, and base32

`base64` provides `encode`/`decode` plus `raw_encode`/`raw_decode` (unpadded), `url_encode`/`url_decode` (URL-safe alphabet), and `raw_url_encode`/`raw_url_decode` (both); its decode functions return strings. `hex` provides `encode`/`decode`, and `base32` provides `encode`/`decode` plus unpadded `raw_encode`/`raw_decode`; their decode functions return bytes.

```
load("base64", base64_encode="encode", base64_decode="decode")
load("hex", hex_encode="encode", hex_decode="decode")
load("base32", b32_encode="encode")

def main(*args, **kwargs):
    enc = base64_encode("user:pass")
    print("base64:", enc, "->", base64_decode(enc))
    print("hex:", hex_encode("hi"))
    print("base32:", b32_encode("hi"))
```

crypto

Hashes, HMAC, AWS request signing, and random values.

- Hashes: `sha1`, `sha256`, `sha512`, and `md5` accept strings and return hex.
- HMAC: `hmac_sha1`/`hmac_sha256`/`hmac_sha512(key, data, output="hex")` and the generic `hmac(algorithm, key, data, output="hex")`; `output` may be `"hex"`, `"base64"`, `"base64_raw"`, or `"bytes"`.
- `sign_v4(method, url, headers, body, access_key, secret_key, region, service, session_token=None, timestamp=None)`: returns the AWS Signature V4 headers (`Authorization`, `X-Amz-Date`, `X-Amz-Content-Sha256`, and `X-Amz-Security-Token` when a session token is given).
- `random_bytes(n)` / `random_hex(n)`: cryptographically secure random output.

```
load("crypto", "sha256", "hmac_sha256", "random_hex", "sign_v4")

def main(*args, **kwargs):
    print("sha256:", sha256("test"))
    print("hmac:", hmac_sha256("key", "message"))
    print("random:", random_hex(8))

    headers = sign_v4("GET", "https://ec2.us-east-1.amazonaws.com/?Action=DescribeInstances",
                      {}, "", "AKIDEXAMPLE", "EXAMPLESECRET", "us-east-1", "ec2")
    print("signed headers:", sorted(headers.keys()))
```

jwt

`encode(claims, key, algorithm="HS256", headers=None)`, `decode(token, key, algorithms=None)` (verifies the signature), and `decode_unverified(token)` (returns `{"header": ..., "claims": ...}` without verification). HS, RS, PS, ES, and EdDSA algorithm families are supported; the `none` algorithm is rejected.

```
load("jwt", jwt_encode="encode", jwt_decode="decode", "decode_unverified")

def main(*args, **kwargs):
```

```
print("claims:", jwt_decode(token, "example-signing-key"))
    print("header:", decode_unverified(token)["header"])
```

gzip

`compress(data)` and `decompress(data)` operate on bytes and return bytes. Decompression is bounded at 1 GiB of output and a 500:1 expansion ratio to guard against decompression bombs.

```
load("gzip", gzip_compress="compress", gzip_decompress="decompress")

def main(*args, **kwargs):
    original = bytes("Hello, runZero!")
    compressed = gzip_compress(original)
    print("compressed length:", len(compressed))
    print("round trip:", gzip_decompress(compressed))
```

flatten_json

`flatten(input, separator="_", root_keys_to_ignore=None)` flattens nested structures into a single-level dict. For building `customAttributes`, prefer `to_custom_attributes`, which also stringifies values and applies the platform limits.

```
load("flatten_json", "flatten")

def main(*args, **kwargs):
    flat = flatten({"a": {"b": 1, "c": 2}, "d": 3})
    print("flattened:", flat)   # {"a_b": 1, "a_c": 2, "d": 3}
```

runzero.progress

Surface progress and log lines from a running task in the runZero console.

- `report(pct, msg="")`: clamps the percentage between 0 and 100 and coalesces calls within 250 ms; messages are truncated to 256 bytes.
- `info(msg)` / `warn(msg)`: emit log lines through the task logger.

```
load("runzero.progress", progress_report="report", progress_info="info",
                         progress_warn="warn")

def main(*args, **kwargs):
    progress_report(0, "starting sync")
    progress_info("fetched page 1")
    progress_warn("retrying after 429")
    progress_report(100, "done")
```

report_assets, report_asset, and pager (predeclared)

Stream `ImportAsset` values to runZero while the script runs, instead of returning them all from `main`. No `load()` is required. See streaming large imports for the pagination pattern.

- `report_asset(asset)`: reports exactly one asset (or `None`, a no-op) and returns `1` or `0`. Streaming batches internally, so report per record; accumulating a page into a list first gains nothing.
- `report_assets(...)`: accepts a single asset, several positional assets, or a list/tuple, returning the count.
- `pager(label="pages", limit=0)`: a loop guard for pagination. `p.next()` is the `while` condition; reaching the page ceiling (CONFIG `maxPages`, default 1,000,000) is an error naming the label, never a silent stop. `limit=` may lower the effective ceiling but not raise it, and `p.page` is the 1-based current page.
- `max_pages()`: the effective page ceiling.

```
load("runzero.types", "ImportAsset")

def main(*args, **kwargs):
    a = ImportAsset(id="asset-1")
    b = ImportAsset(id="asset-2")

    report_asset(a)              # one asset; the preferred streaming pattern
    report_assets(a, b)          # several positional assets
    n = report_assets([a, b])    # a list, returning the count
    print("reported", n)

    p = pager(label="devices")
    while p.next():
        print("page", p.page)
        break                    # a real script would fetch until the API runs dry
    return None
```

socket

Raw TCP, UDP, and TLS connections for simple line- or byte-oriented protocols.

- `tcp(host, port, timeout=30, tls=None, insecure_skip_verify=False, server_name=None)`: `tls=` accepts a bool or the same dict of TLS overrides as the `http` module (`insecure`, `server_name`, `ca_pem`, `client_cert_pem`, `client_key_pem`, `thumbprints`), so `kwargs.get_http_tls()` output can be splatted straight into a socket.
- `udp(host, port, timeout=30)`
- `tls(host, port, timeout=30, insecure_skip_verify=False, server_name=None, tls=None)`

Sockets expose `send(data, timeout=None)`, `recv(max=..., timeout=None)` (max defaults to 16 MiB, per-call ceiling 200 MiB), `recv_exact(n, ...)`, `recv_line(...)` and `recv_until(delim, ...)` (max defaults to 1 MiB), `starttls(...)` (TCP only, also accepts `tls=`), `set_timeout(seconds)`, and `close()`, plus the attributes `local_addr`, `remote_addr`, `is_tls`, `closed`, and `network`. Always `close()` sockets when done.

On an Explorer task, sockets can reach the internal addresses visible to that Explorer, the same as the direct-protocol modules, so scope the Explorer and the credential to the intended system. On console-hosted runs, private and internal addresses are blocked for every module, sockets included.

```
load("socket", "tls")

def main(*args, **kwargs):
    host = kwargs.get("host", "example.com")
    sock = tls(host, 443, timeout=10)
    print("connected to:", sock.remote_addr, "tls:", sock.is_tls)
    sock.close()
```

Direct-protocol modules

For sources without a REST API, these modules open authenticated connections from the Explorer the task runs on, and can reach internal addresses visible to that Explorer. They return session objects. Always `close()` them when done: a script can hold at most 256 open connections and sessions at once, and network reads across every module share a 4 GiB per-run budget. Scripts using these modules should declare `"validationMode": "compile"`, since `--validate` cannot dial a real endpoint.

The examples below read their targets from `--kwargs` so they can run against a test system:

```
runzero script --filename example.star --kwargs host=192.0.2.10 --kwargs username=svc --kwargs password=...
```

runzero.ssh

`dial(host, username, password=None, private_key=None, private_key_passphrase=None, host_key=None, host_keys=None, insecure_ignore_host_key=False, port=22, timeout=30)`. Host keys are never trusted on first use: pin the expected host key(s), in `authorized_keys` or PEM form (for example from `ssh-keyscan`), or explicitly pass `insecure_ignore_host_key=True`. Combining pins with the insecure flag is an error. Password auth also answers keyboard-interactive prompts, and timeouts clamp at 600 seconds.

Sessions provide:

- `run(command, stdin=None, timeout=0)`: returns `(stdout, stderr, exit_code)`; a `timeout` of 0 uses the session timeout. `stdin` is capped at 1 MiB and each output stream at 16 MiB; `run_command` is an alias.
- `stream(command, stdin=None, timeout=0)`: starts a long-lived remote command and returns a stream with `send(data)`, `recv(max=..., timeout=None)` (max defaults to 1 MiB; `b""` on EOF), `close_stdin()`, `close()`, and the attributes `exit_code` (`None` until exit), `stderr`, and `closed`. Output is consumed incrementally with no per-stream cap, so a stream can process more than `run` can return.
- `open_unix(path)` / `open_tcp(host, port)`: port forwarding to a remote Unix socket or a TCP address reachable from the remote host, returning a full socket object. This is how a script reaches a service that only listens locally on the remote system, such as a management socket or a loopback-only listener.

Sessions also expose `host`, `port`, `username`, and `close()`.

```
load("runzero.ssh", ssh_dial="dial")

def main(*args, **kwargs):
```

```
if not host:
        print("pass --kwargs host=... --kwargs username=... --kwargs password=... to run")
        return None
    session = ssh_dial(host=host, username=kwargs["username"],
                       password=kwargs.get("password"),
                       insecure_ignore_host_key=True)
    stdout, stderr, code = session.run("uname -a")
    session.close()
    print("exit:", code, "output:", stdout)
```

runzero.smb

`dial(host, username, password="", domain="", nt_hash=None, port=445, timeout=30)` accepts `DOMAIN\user` shorthand in `username`, and timeouts clamp at 600 seconds. Sessions provide `list_shares()` and `mount(share)`; a mounted share has `read(path, limit=0)` (single reads capped at 64 MiB), `list(path="/")`, `stat(path)`, `exists(path)`, `unmount()`, `close()`, and a `name` attribute.

Access refusals are values, not errors: `mount()` and `list()` return `None` on access denied (distinct from an empty list), so a script can skip a share it cannot read and keep going. `read()` and `stat()` still raise; `exists()` is `stat`'s soft counterpart.

```
load("runzero.smb", smb_dial="dial")

def main(*args, **kwargs):
    host = kwargs.get("host")
    if not host:
        print("pass --kwargs host=... --kwargs username=... --kwargs password=... to run")
        return None
    session = smb_dial(host=host, username=kwargs["username"], password=kwargs["password"])
    print("shares:", session.list_shares())
    session.close()
```

runzero.winrm

`dial(host, username, password, port=0, https=False, insecure_skip_verify=False, ca_cert=None, auth="ntlm", timeout=60)` (port 0 selects the default, 5985 or 5986 with `https=True`). Sessions provide `run(command)` (alias `run_command`), `run_powershell(script)` (alias `run_ps`), and `wql(query, namespace="root/cimv2")`, plus the attributes `host`, `port`, and `https`.

```
load("runzero.winrm", winrm_dial="dial")

def main(*args, **kwargs):
    host = kwargs.get("host")
    if not host:
        print("pass --kwargs host=... --kwargs username=... --kwargs password=... to run")
        return None
    session = winrm_dial(host, kwargs["username"], kwargs["password"])
    stdout, stderr, code = session.run_powershell("Get-ComputerInfo | Select-Object OsName")
    session.close()
    print(stdout)
```

runzero.wmi

`dial(host, username, password, transport="tcp", namespace="//./root/cimv2", target_name="", timeout=60, port=0, smb_port=445)` (port 0 selects the default, 135). Sessions provide `query(query, limit=0, page=100)` returning a list of dicts.

```
load("runzero.wmi", wmi_dial="dial")

def main(*args, **kwargs):
    host = kwargs.get("host")
    if not host:
        print("pass --kwargs host=... --kwargs username=... --kwargs password=... to run")
        return None
    session = wmi_dial(host, kwargs["username"], kwargs["password"])
    rows = session.query("SELECT Caption, Version FROM Win32_OperatingSystem")
    session.close()
    print(rows)
```

runzero.sql

`connect(driver, dsn, timeout=30, max_open_conns=4, max_idle_conns=4, conn_max_lifetime=0)` with drivers `postgres`, `mysql`, and `mssql` (and common aliases). Sessions provide `query(query, params=None, limit=100000, timeout=0)` returning a list of dicts (row limit up to 1,000,000), and `exec(query, params=None, timeout=0)` returning a dict with `rows_affected` and `last_insert_id`. A `timeout` of 0 uses the session timeout; timeouts clamp at 600 seconds. DSNs are restricted to network connections: file-reading options (such as `sslkey`, `passfile`, or `allowAllFiles`) and Unix-socket or named-pipe transports are rejected.

```
load("runzero.sql", sql_connect="connect")

def main(*args, **kwargs):
    dsn = kwargs.get("dsn")
    if not dsn:
        print("pass --kwargs dsn=postgres://user:pass@host:5432/db to run")
        return None
    db = sql_connect("postgres", dsn)
    rows = db.query("SELECT version()")
    db.close()
    print(rows)
```

Dragos

runZero imports assets and vulnerabilities from the Dragos API to help expand your OT visibility.

Getting started with Dragos

To set up the Dragos integration:

1.

1. Generate a Dragos API ID and API Secret with these privileges:
    - asset:read
    - detection:read
    - vulnerability:read
2. Configure the Dragos credential in runZero.
3. Activate the integration to pull your data into runZero.

Step 1: Generate a Dragos API ID and API Secret

1. Log into the Dragos console, go to Admin, then Users. Under the user, click the Add New API Key button.
2. In the Generate New API Key box, name your API ID and API Secret, then click Generate Key.
3. Copy the ID and Secret. You'll need both in the next step.

Step 2: Add the Dragos API ID and API Secret to runZero

1. Go to the Credentials page in runZero.
2. Choose **Dragos API ID & Secret** from the list of credential types.
3. Name the credential, for example `Dragos`.
4. Fill in the credential fields:
    - **Dragos API ID** is the API ID from Step 1.
    - **Dragos API Secret** is the API Secret from Step 1.
    - **Dragos API URL** is the URL of your Dragos instance, without a trailing slash. For example, `https://my.instance.dragos.cloud`.
    - **Insecure** allows authentication to untrusted endpoints. Enabling it disables certificate validation, so use it with caution.
5. If other organizations should be able to use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
6. Verify and save the credential. If the URL is an internal IP address, verification is unsupported, but you can still save the credential and use the integration from an Explorer.

Step 3: Choose how to configure the Dragos integration

You can run the Dragos integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

With your Dragos credential saved, set up a connector task or scan probe to sync your data.

Step 4a: Configure the Dragos integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Dragos-only assets.

1. Activate a connection to Dragos. All third-party connections are also available from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they are missing from the list, confirm that they have access to the organization you are working in.
3. Name the task, for example `Dragos Sync` (optional).
4. Describe the task (optional).
5. Choose the Explorer that runs this connector task (optional).
6. Choose the site to add your assets to. Every newly discovered asset is stored in this site.
7. To leave out assets that runZero has not scanned, select the **Exclude assets that cannot be merged into an existing asset** option. By default, the integration includes assets that runZero has not scanned.
8. To skip importing vulnerabilities detected on Dragos assets, select the **Disable importing vulnerabilities from the Dragos inventory** option. By default, the integration imports both assets and vulnerabilities.
9. Choose the combination of **Priorities**, **Severities**, **Tags**, and **Subnets** to filter assets by. These match the filter criteria available in Dragos.
10. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
11. Activate the connection. The sync runs on the schedule you set, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Dragos integration as a scan probe

When the Dragos integration runs as a scan probe, the runZero Explorer pulls your Dragos assets into the runZero Console.

In a new or existing scan configuration:

- Set the *DRAGOS* option to *Yes* in the *Probes and SNMP* tab, and adjust the default options if needed.
- Set the correct *DRAGOS* credential to *Yes* in the *Credentials* tab.

Step 5: View Dragos assets

After a successful sync, open your inventory to see your Dragos assets. They show a Dragos icon in the **Source** column.

To filter for Dragos assets, run this query:

- View all Dragos assets:

```
source:Dragos
```

Click into any asset to see its attributes, including those gathered from Dragos. For more ways to filter, see the asset inventory search keywords.

Google Cloud Platform

The Google Cloud Platform (GCP) integration syncs your GCP cloud inventories with runZero to give you visibility into your cloud assets. runZero also integrates with other cloud providers, such as Microsoft Azure and Amazon AWS. As with other integrations, you add the credentials needed to authenticate to GCP and set up a connector in runZero. runZero pulls in GCP compute instance VMs along with their GCP attributes, which you can view on each asset.

The integration supports these GCP asset types:

- Compute Engine instances
- Load balancers
- Cloud SQL

Requirements

Before you begin, you need:

- A Google Cloud service account with the `Compute Network Viewer` and `Cloud SQL Viewer` roles.
  - Grant this service account access to each project you want the integration to gather data from.
- A downloaded key for the GCP service account.
- These GCP APIs enabled on each project:
  - Compute Engine
  - Cloud SQL Admin

How to set up the Google Cloud Platform integration

Work through these steps in order:

- Create a Google Cloud Platform credential in runZero.
- Choose whether to run the integration as a scan probe or connector task.
- Activate the integration for Google Cloud Platform.
- View your GCP assets.

Step 1: Create Google Cloud Platform credentials

1. Go to the Credentials page and click **Add Credential**.
2. From the **Credentials type** dropdown, choose **GCP Service Account Key**.
3. Name the credential, for example `GCP`.
4. Set the **Include all projects** toggle to *Yes* to gather asset data from every GCP project the service account can access. With the toggle at *No*, the integration gathers asset data only from the project named in the key file.
5. Click **Choose file** and upload the service account key file you downloaded from GCP.
6. If other organizations should be able to use this credential, select the **Make this a global credential** option. Otherwise, you can grant access per organization.
7. Save the credential.

Step 2: Choose how to configure the GCP integration

You can run the GCP integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 3: Activate the Google Cloud Platform integration

With your GCP credential saved, set up a connector task or scan probe to sync your data.

Step 3a: Configure the GCP integration as a connector task

1. Activate a connection to GCP. All third-party connections are also available from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it is missing from the list, confirm that the credential has access to the organization you are working in.
3. Name the task, for example `Google Cloud Platform sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Choose the site to add your assets to, either an existing site or a new one created when the sync runs. Sites group related assets, and you can have runZero automatically generate a new site per GCP project by selecting that option in the task configuration.
6. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes assets that runZero has not scanned.
7. Activate the connection. The sync runs on the schedule you set, and the Scheduled tasks page shows when the next sync will occur.

Step 3b: Configure the GCP integration as a scan probe

1. Create a new scan task, or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters as needed.
3. On the Probes and SNMP tab, choose any additional probes to include, set the GCP toggle to *Yes*, and adjust the default options if needed.
4. On the Credentials tab, set the GCP toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 4: View your Google Cloud Platform assets

After a successful sync, open your inventory to see your GCP assets. They show a Google icon in the **Source** column.

To filter for GCP assets, run this query:

- View all GCP assets:

```
source:gcp
```

Click into any asset to see its attributes, including those returned by GCP. For more ways to filter, see the asset inventory search keywords.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Google Cloud Platform integration unable to connect?

1. Check whether the GCP integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the GCP integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Google Workspace

runZero imports Google Workspace data to sync and enrich your asset inventory and to give you visibility into users and groups. With Google Workspace data in runZero, unmanaged assets on your network are easier to find. The integration supports the ChromeOS, Mobile, and Endpoint registered asset types.

Requirements

- Use an existing Google service account or create one in whichever project suits you best.
- Create and download a key for the service account, and save the JSON file.
- Enable the `Admin SDK` and `Cloud Identity` APIs for the project. Use the search box in the API Library to find each API and enable it.
- Enable domain-wide delegation in the Google Workspace console.
    - Add a new API client, using the service account's unique numeric ID as the Client ID.
    - Enable these OAuth scopes for the API client:

```
https://www.googleapis.com/auth/admin.directory.user.readonly,
https://www.googleapis.com/auth/admin.directory.group.readonly,
https://www.googleapis.com/auth/admin.directory.device.mobile.readonly,
https://www.googleapis.com/auth/admin.directory.device.chromeos.readonly,
https://www.googleapis.com/auth/cloud-identity.devices.readonly
```

- Or enter each OAuth scope individually:
- `https://www.googleapis.com/auth/admin.directory.user.readonly`
- `https://www.googleapis.com/auth/admin.directory.group.readonly`
- `https://www.googleapis.com/auth/admin.directory.device.mobile.readonly`
- `https://www.googleapis.com/auth/admin.directory.device.chromeos.readonly`
- `https://www.googleapis.com/auth/cloud-identity.devices.readonly`

How to set up the Google Workspace integration

Work through these steps in order:

- Create a Google Workspace credential in runZero.
- Choose whether to run the integration as a scan probe or connector task.
- Activate the connection for Google Workspace.
- View your results.

Step 1: Create a Google Workspace credential

1. Go to the Credentials page and click **Add Credential**.
2. From the **Credentials type** dropdown, choose **Google Workspace Client Secret**.
3. Name the credential, for example `Google Workspace`.
4. In the *Admin account email* field, enter the email address of an administrator account with access to the assets, users, or groups you want to import.
5. Optionally, enter a *Customer ID* to import from an organization other than the one your administrator account belongs to. By default, runZero uses the Customer ID associated with the service account.
6. Click **Choose file** and upload the service account key file you downloaded from Google Workspace.
7. If other organizations should be able to use this credential, select the **Make this a global credential** option. Otherwise, you can grant access per organization.
8. Save the credential.

Step 2: Choose how to configure the Google Workspace integration

You can run the Google Workspace integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task, and is useful when you run a self-hosted runZero Platform and your console cannot reach Google Workspace. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync. In most situations, a scheduled connection is the easier way to sync your data from Google Workspace.

Step 3: Activate the Google Workspace integration

With your Google Workspace credential saved, set up a connector task or scan probe to sync your data.

Step 3a: Configure the Google Workspace integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero organizes the data.

1. Activate a connection to Google Workspace. All third-party connections are also available from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it is missing from the list, confirm that the credential has access to the organization you are working in.
3. Name the task, for example `Google Workspace sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Choose the site to add your assets to, either an existing site or a new one created when the sync runs. Sites group related assets.
6. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes assets that runZero has not scanned.
7. Activate the connection. The sync runs on the schedule you set, and the Scheduled tasks page shows when the next sync will occur.

Step 3b: Configure the Google Workspace integration as a scan probe

1. Create a new scan task, or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters as needed.
3. On the Probes and SNMP tab, choose any additional probes to include, set the GoogleWorkspace toggle to *Yes*, and adjust the default options if needed.
4. On the Credentials tab, set the GoogleWorkspace toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 4: View Google Workspace assets

After a successful sync, open your inventory to see your Google Workspace assets. They show a Google Workspace icon in the **Source** column.

To filter for Google Workspace assets, run these queries:

- View all Google Workspace assets:

  ```
  source:googleworkspace
  ```

- View runZero assets not connected to Google Workspace:

  ```
  source:runzero AND NOT source:googleworkspace
  ```

Click into any asset to see its attributes, including those returned by Google Workspace.



Besides enriching asset inventory data, the Google Workspace integration imports details about users and groups. Go to Inventory > Users or Inventory > Groups to view them, and use the query `source:googleworkspace` to filter your results.

To search and filter those users, see the Users inventory keywords.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Google Workspace integration unable to connect?

1. Check whether the Google Workspace integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Google Workspace integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Microsoft 365 Defender

runZero imports Microsoft 365 Defender data to sync and enrich your asset, software, and vulnerability inventory. With Microsoft 365 Defender data in runZero, assets missing EDR protection are easier to find.

Getting started

To set up the Microsoft 365 Defender integration:

1. Configure Microsoft 365 Defender to allow API access for runZero.
2. Add the Microsoft 365 Defender credential in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the Microsoft 365 Defender integration to sync your data with runZero.

Requirements

Before you set up the Microsoft 365 Defender integration:

- Make sure you have access to the Microsoft Azure portal.

Step 1: Register an Azure application for Microsoft 365 Defender API access

runZero can authenticate to the Microsoft 365 Defender API with a client secret. To set that up, register an application with Microsoft 365 Defender API access.

1. Sign in to the Microsoft Azure portal.
2. Go to *App registrations* and click *+ New registration*.
    - Provide a name.
    - Select the supported account types.
    - Optionally add a redirect URI.
3. Click *Register*.
4. After the application is created, you should see the Overview dashboard. Record these values:
    - Application (client) ID
    - Directory (tenant) ID
5. On the application's details page, go to *Manage > API permissions* and choose *+ Add a permission*.
6. Select the second tab, *APIs my organization uses*, to view the available APIs.
7. Select `WindowsDefenderATP` from the list of Microsoft APIs.
8. Select the `Application permissions` permission type to configure a *client secret*.
9. Search for and select these required permissions:
    - Windows Defender ATP API permissions:
        - `Machine.Read.All`
        - `Software.Read.All`
        - `Vulnerability.Read.All`
10. Click *Add permissions* to save them to the application.
11. Click *Grant admin consent* to approve the permissions for the application.
12. Generate the client secret from the app management pages:
    - Go to *App registrations* and select the application you created.
    - Go to *Certificates & secrets* and click *+ New client secret*.
        - Enter a description.

        - Select the expiration.
    - Click *Add* to create the client secret, and save the client secret value.

Step 2: Add an Azure Client Secret credential to runZero

This credential type can sync all resources in a single directory, across multiple subscriptions.

1. Go to the Credentials page in runZero and click *Add Credential*.
2. Name the credential, for example `Azure Client Secret`.
3. Choose *Azure Client Secret* from the list of credential types.
4. Fill in the credential fields:
    - **Azure application (client) ID** is the unique ID of the registered application. Find it in the Azure portal under *App registrations* after selecting the application.
    - **Azure client secret** is generated under *App registrations*: select your application, go to *Manage > Certificates & secrets*, and click *New client secret*.
    - **Azure directory (tenant) ID** is the unique ID of the tenant. It is also in the Azure portal under *App registrations* after selecting the application.
5. If other organizations should be able to use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
6. Save the credential.

Step 3: Choose how to configure the Microsoft 365 Defender integration

You can run the Microsoft 365 Defender integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the Microsoft 365 Defender integration to sync data

With your Microsoft 365 Defender credential saved, set up a connector task or scan probe to sync your data.

Step 4a: Configure the Microsoft 365 Defender integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Microsoft 365 Defender-only assets.

1. Activate a connection to Microsoft 365 Defender. All third-party connections are also available from the integrations page, your inventory, or the tasks page.

2. Choose the credential you added earlier. If it is missing from the list, confirm that it has access to the organization you are working in.
3. Name the task, for example `Microsoft 365 Defender sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Under **Task configuration**, choose the site to add your assets to.
6. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes assets that runZero has not scanned.
7. To include assets that have stopped reporting to the Microsoft 365 Defender service, switch the **Include inactive assets** toggle to *Yes*. By default, the integration leaves out assets marked as inactive.
8. Activate the connection. The sync runs on the schedule you set, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Microsoft 365 Defender integration as a scan probe

1. Create a new scan task, or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters as needed.
3. On the Probes and SNMP tab, choose any additional probes to include, set the Defender365 toggle to `Yes`, and adjust the default options if needed.
4. On the Credentials tab, set the MS365Defender toggle to `Yes` for the credential you want to use.
5. Click *Initialize scan* to save the scan task. It runs immediately or at the scheduled time.

Step 5: View Microsoft 365 Defender assets

After a successful sync, open your inventory to see your Microsoft 365 Defender assets. They show an Active Directory icon in the **Source** column.

To filter for Microsoft 365 Defender assets, run these queries:

- View all Microsoft 365 Defender assets:

  ```
  source:ms365defender
  ```

- View runZero assets not connected to Microsoft 365 Defender:

  ```
  source:runzero AND NOT source:ms365defender
  ```

Click into any asset to see its attributes, including those returned by Microsoft 365 Defender. To search and filter the vulnerabilities this integration imports, see the vulnerability instance inventory keywords.

Filtering Microsoft 365 Defender assets

You can apply an optional filter to Microsoft 365 Defender integration tasks. runZero uses the Microsoft Graph $filter query parameter to filter assets. A filter follows the syntax `<property> [operator] <value>`, and you can combine expressions with `and` or `or` for more complex filtering.

Properties

You can filter on any property that runZero imports from Microsoft 365 Defender. Some examples:

Defender Property	runZero Attribute	Description	Example
`computerDNSName`	@ms365defender.dev.computerDNSname	The hostname of the device	EXPLORER-01
`osPlatform`	@ms365defender.dev.osPlatform	The operating system of the device	Windows11, Android
`heatlhStatus`	@ms365defender.dev.heatlhStatus	The health status of the Defender agent that is installed	Active, Inactive
`isAADJoined`	@ms365defender.dev.isAADJoined	Boolean value specifying whether the device is registered in Azure AD	true, false
`managedBy`	@ms365defender.dev.managedBy	Description of how the device is managed	Intune

Operators

Common operators in a Microsoft 365 Defender filter:

- Equal to (`eq`)
- Not equal to (`ne`)
- Has (`has`)
- Less than (`lt`)
- Greater than (`gt`)

- Less than or equal to (`le`)
- Greater than or equal to (`ge`)

Common functions in a Microsoft 365 Defender filter follow the syntax `function(<property>, <value>)`:

- Starts with (`startswith`)
- Ends with (`endswith`)
- Contains (`contains`)

Example filters

Search Filter	Description
`not(osPlatform eq 'Android')`	Import all assets except those with an Android operating system
`not(osPlatform eq 'Android') and not(osPlatform eq 'iOS')`	Import all assets except those with an Android or iOS operating system
`startswith(computerDNSname, 'PROD')`	Import all devices with a hostname that starts with PROD
`not(startswith(computerDNSname, 'DEV'))`	Import all devices except those with a hostname that starts with DEV

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Microsoft 365 Defender integration unable to connect?

1. Check whether the Microsoft 365 Defender integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Microsoft 365 Defender integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Microsoft Active Directory

runZero integrates with Microsoft Active Directory (AD) over LDAP to sync and enrich your asset inventory and to show you domain users and groups. With your AD data in runZero, assets that aren't part of your domain are easier to find.

Getting started

To set up the Active Directory integration:

1. Add the AD credential in runZero.
2. Choose whether to run the integration as a scan probe or connector task.
3. Activate the Active Directory integration to sync your data with runZero.

Requirements

You need credentials for an LDAP account.

Step 1: Add the LDAP credential to runZero

1. Go to the Add credential page in runZero and name the credential, like `LDAP`.
2. Choose **LDAP Username & Password** from the list of credential types.
3. Provide the following information:
    - **LDAP username**: The username the integration uses to sign in. The account doesn't need any special permissions. runZero accepts these username formats:
        - Distinguished Name (DN): `CN=[username],CN=Users,DC=[domain],DC=[tld]`
        - User Principal Name (UPN): `[username]@[domain].[tld]`
        - Domain\Username: `[domain]\[username]`
    - **LDAP password**: The password for that username.
    - **LDAP base DN**: The base distinguished name for LDAP searches, in distinguished name format: `DC=[domain],DC=[tld]`. runZero imports only entities underneath this base DN.
    - **LDAP URL**: The URL for your LDAP server, as `IP[:port]` or `hostname.domain.tld[:port]`. It must begin with `ldap://` (for insecure LDAP connections) or `ldaps://` (for secure LDAP connections). For example: `ldaps://ad.example.com:636`
    - **LDAP insecure**: Set this to *Yes* to attempt authentication without a verified thumbprint. By default, runZero tries LDAPS first, then falls back to LDAP+StartTLS, then LDAP. LDAP without StartTLS only works when this toggle is *Yes*.
    - **LDAP thumbprints** (optional): A set of `IP[:port]=SHA256:B64HASH` or `hostname.domain.tld=SHA256:B64HASH` pairs to trust for authentication.
        - To get the TLS thumbprint, scan your LDAP server with runZero. The TLS fingerprints service attribute report lists every fingerprint seen so far. Thumbprints for self-signed certificates only work with LDAPS. To use LDAP+StartTLS with a self-signed certificate, set the **Insecure** option to `Yes`.
        - If *LDAP insecure* is *No*

and you provide no thumbprints:
    - With a self-signed certificate, the connection fails because the certificate chain can't be verified.
    - With a valid certificate from a public CA, the connection works without thumbprints.
4. To let all other organizations use this credential, select the **Make this a global credential** option. Otherwise, you can configure access per organization.
5. Save the credential.

Step 2: Choose how to configure the Active Directory integration

You can run the Active Directory integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 3: Set up and activate the Active Directory integration to sync data

With your Active Directory credential saved, set up a connector task or a scan probe to sync your data.

Step 3a: Configure the Active Directory integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new LDAP-only assets.

1. Activate a connection to Active Directory. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.
3. Enter a name for the task, like `LDAP sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Under **Task configuration**, choose the site to add your assets to.
6. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
7. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 3b: Configure the Active Directory integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose any additional probes to include, set the LDAP toggle to *Yes*, and change any default options as needed.
4. On the Credentials tab, set the LDAP toggle to *Yes* for the credential you want to use.

5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 4: View Active Directory assets

After a successful sync, go to your inventory to view your LDAP assets. They show an Active Directory icon in the **Source** column.

To filter for LDAP assets, try these queries:

- View all LDAP assets:

  ```
  source:ldap
  ```

- View runZero assets not connected to LDAP:

  ```
  source:runzero AND NOT source:ldap
  ```

Click into an asset to see the attributes LDAP returned. To write your own queries, start with the search query syntax.

> The LDAP integration also imports details about users and groups. Go to Inventory > Users or Inventory > Groups to see them.

Troubleshooting

If the integration isn't working, start with these checks.

Why is the Microsoft Active Directory integration unable to connect?

1. Check whether the Microsoft Active Directory integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Microsoft Active Directory integration can't connect, check the task log for errors. Common errors include:
   - 500: server error, unable to connect to the endpoint
   - 404: hitting an unknown endpoint on the server
   - 403: not authorized, likely a credential issue
4. Verify that the integration task runs from an Explorer with access to the Microsoft Active Directory host.

Microsoft Azure

runZero integrates with Microsoft Azure to give you visibility into your cloud assets. The integration imports data through each applicable API to enrich your asset inventory:

- Virtual Machines API
- Virtual Machine Scale Sets API
- Load Balancers API
- AzureSQL API
- Web Apps API

A sync shows you each asset's OS profile, storage profile, and more. The integration imports assets that are in a running state.

Getting started

runZero supports these Azure resource types:

- Virtual Machines
- Virtual Machine Scale Sets
- Azure SQL
- Azure Cosmos DB
- Load Balancers
- Function Apps

To set up the Azure integration:

1. Configure Azure to allow API access through runZero.
2. Add an Azure credential to runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the Azure integration to sync your data with runZero.

Requirements

You need access to the Microsoft Azure portal.

Step 1: Configure Azure to allow API access through runZero

1. Sign in to the Microsoft Azure portal.
2. Go to **Azure Active Directory > App registrations** and click **New registration**.
    - Provide a name.
    - Select the supported account types.
    - Optionally add a redirect URI.
3. Click **Register**.
4. Once the application is created, you should see the **Overview** dashboard. Save the following information:
    - Application (client) ID

    - Directory (tenant) ID
5. Give the client access to the subscriptions you want to sync. From the subscription details page, go to **Access Control (IAM)** and select **Add > Add role assignment**. Enter the following:
    - Role: Reader
    - Assign access to: User, group, or service principal
    - Under **Select**, search for the name of the application you created. Click your application to add it to the **Selected members** list below.
6. Click **Save**.
7. Go to **Azure Active Directory > App registrations** and select the application you created.
8. Go to **Certificates & secrets** and click **New client secret**.
    - Enter a description.
    - Select the expiration.
9. Click **Add** to create the client secret. Save the following information:
    - Client secret value

Step 2: Add the Azure credential to runZero

The Azure integration accepts either a client secret or a username and password as its credential.

Step 2a: Add an Azure Client Secret credential to runZero

You can use this credential type to sync all resources in a single directory (across multiple subscriptions).

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Provide a name for the credential, like `Azure Client Secret`.
3. Choose **Azure Client Secret** from the list of credential types.
4. Provide the following information:
    - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to **Azure Active Directory > App registrations** and select the application.
    - **Azure client secret**: To generate a client secret, go to **Azure Active Directory > App registrations**, select your application, go to **Certificates & secrets**, and click **New client secret**.
    - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to **Azure Active Directory > App registrations** and select the application.
    - Enable the **All subscriptions** option to sync all resources in your directory. With it enabled, you can list subscription IDs in the **Exclude subscription IDs** text area to leave specific ones out. With it disabled, you can list specific subscription IDs in the **Include subscription IDs** text area. To find a subscription ID in the Azure portal, go to **Subscriptions** and select the subscription. Enter one subscription ID per line in either text area.
5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.
6. Save the credential.

Step 2b: Add an Azure Username & Password credential to runZero

You can use this credential type to sync all resources across directories. Alternatively, add one **Azure Client Secret** credential for each Azure directory you want to sync.

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Provide a name for the credential, like `Azure User/Pass`.
3. Choose **Azure Username & Password** from the list of credential types.
4. Provide the following information:
    - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to **Azure Active Directory > App registrations** and select the application.
    - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to **Azure Active Directory > App registrations** and select the application.
    - **Azure username**: The username for your Azure cloud account. It can't be a federated user account.
    - **Azure password**: The password for your Azure cloud account.
5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.
6. Save the credential.

Step 3: Choose how to configure the Azure integration

You can run the Azure integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the Azure integration to sync data

With your Azure credential saved, set up a connector task or a scan probe to sync your data.

Step 4a: Configure the Azure integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Azure-only assets.

1. Activate a connection to Azure. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.
3. Enter a name for the task, like `Azure sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Under **Task configuration**, choose the site to add your assets to. runZero stores all newly discovered assets in this site.

6. Under **Service options**, select the services to sync data from. You must choose at least one.
7. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
8. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Azure integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose any additional probes to include, set the Azure toggle to *Yes*, and change any default options as needed.
4. On the Credentials tab, set the Azure toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 5: View Azure assets

After a successful sync, go to your inventory to view your Azure assets. They show an Azure icon in the **Source** column.

To filter Azure assets, try these queries:

- View all Azure assets:

  ```
  source:azure
  ```

- View all Azure VMs:

  ```
  has:"@azure.vm.vmID"
  ```

- View all Azure virtual machine scale set VMs:

  ```
  has:"@azure.vmss.vmID"
  ```

- View all Azure load balancers:

  ```
  has:"@azure.lb.id"
  ```

- View all AzureSQL instances:

  ```
  has:"@azure.azsql.id"
  ```

- View all Azure Cosmos DB instances:

  ```
  has:"@azure.cosmos.id""
  ```

- View all Azure Function Apps:

  ```
  @azure.functionapp.kind:"functionapp"
  ```

Click into an asset to see the attributes the Azure APIs returned. To write your own queries, start with the search query syntax.

Troubleshooting

If the integration isn't working, start with these checks.

Why is the Microsoft Azure integration unable to connect?

1. Check whether the Microsoft Azure integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Microsoft Azure integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Microsoft Endpoint Configuration Manager (MECM)

runZero integrates with Microsoft Endpoint Configuration Manager (MECM), formerly System Center Configuration Manager (SCCM), by importing data from the MECM MSSQL database. Syncing your MECM device data makes unmanaged devices on your network easier to find.

Getting started with MECM

To set up an integration with MECM:

1. Identify or create a database user with read access to the MECM database.
2. Configure the MECM credential in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the integration to pull your data into runZero.

Step 1: Identify or create a database user for access to MECM

1. Identify an existing database user with read access to the database.
2. Alternatively, create a dedicated read-only database user for this integration. Microsoft's documentation covers the details: Create a database user.

Step 2: Add the MECM database connection string to runZero

1. Go to the Credentials page in runZero.
2. Choose **MECM Database Connection String** from the list of credential types.
3. Provide a name for the credential, like `MECM`.
4. Provide the database connection string in one of these formats:
    - `Server=host,port;Database=database-name;User Id=user-id;Password=password;`
      With this format, the values should not contain a semicolon (;). You can use single or double quotes to escape special characters.
    - `sqlserver://username:password@host/instance?database=value&param=value`
5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.
6. Save the credential.

Step 3: Choose how to configure the MECM integration

You can run the MECM integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

With your MECM credential saved, set up the sync from MECM.

Step 4a: Configure the MECM integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new assets.

1. Activate a connection to MECM. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.
3. Enter a name for the task, like `MECM Sync` (optional).
4. Choose the Explorer that runs this connector task (optional).
5. Choose the site to add your assets to. runZero stores all newly discovered assets in this site.
6. Enter a description for the task (optional).
7. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
8. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
9. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the MECM integration as a scan probe

When you run the MECM integration as a scan probe, the runZero Explorer pulls your MECM devices into the runZero Console.

In a new or existing scan configuration:

- Set the *MECM* option to *Yes* in the *Probes and SNMP* tab and change any default options as needed.
- Set the correct *MECM* credential to *Yes* in the *Credentials* tab.

Step 5: View MECM assets

After a successful sync, go to your inventory to view your MECM assets. They show a Microsoft icon in the **Source** column.

To filter for MECM assets, try this query:

- View all MECM assets:

  ```
  source:mecm
  ```

Click into an asset to see the attributes runZero gathered from MECM. To write your own queries, start with the search query syntax.

Microsoft Entra ID

runZero integrates with Microsoft Entra ID (formerly Azure AD) to sync and enrich your asset inventory and to show you Entra ID users and groups. With your Entra ID data in runZero, assets that aren't part of your domain are easier to find.

The runZero product still refers to Entra ID as Azure AD.

Getting started

To set up the Entra ID integration:

1. Configure Entra ID to allow API access through runZero.
2. Add the Entra ID credential in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the Entra ID integration to sync your data with runZero.

Requirements

You need access to the Microsoft Azure portal.

Step 1: Register an Azure application for Entra ID API access

runZero can authenticate to the Entra ID API with either a username and password or a client secret. Either way, start by registering an application for Entra ID API access.

1. Sign in to the Microsoft Azure portal.
2. Go to *App registrations* and click *+ New registration*.
    - Provide a name.
    - Select the supported account types.
    - Optionally add a redirect URI.
3. Click *Register*.
4. Once the application is created, go back to the main Azure portal page and select *App registrations*. You should find the application you just registered (selecting the *Owned applications* tab may help). Click the app's name.
5. On the app registration overview, note the following information:
    - Application (client) ID
    - Directory (tenant) ID
6. Click the application's display name to open its management pages.
7. Select *Manage > Authentication* on the left. Set **Allow public client flows** to *Yes* and save the configuration.
8. Go to *API permissions* and click *+ Add a permission*.
9. Select *Microsoft Graph* from the list of Microsoft APIs.
10. Select the permissions type for the credential you plan to use:
    - **Username & password**: select *Delegated permissions*
    - **Client secret**: select *Application permissions*
11. Search for and select these required permissions:
    - *Device.Read.All*
    - *Group.Read.All*
    - *User.Read.All*
12. Click *Add permissions* to save the permissions to the application.
13. Click *Grant admin consent* to grant the application consent for those permissions.
14. If you're using a client secret, also do the following:
    - Go to *Azure Active Directory > App registrations* and select the application you created.

- Go to *Certificates & secrets* and click *New client secret*.
  - Enter a description.
  - Select the expiration.
- Click *Add* to create the client secret, then save the client secret value.

Step 2: Add the Entra ID credential to runZero

Step 2a: Add an Azure Username & Password credential to runZero

1. Go to the Credentials page in runZero and click *Add Credential*.
2. Provide a name for the credential, like `Azure Username & Password`.
3. Choose *Azure Username & Password* from the list of credential types.
4. Provide the following information:
   - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to *App registrations* and select the application.
   - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to *App registrations* and select the application.
   - **Azure username**: The username for your Azure cloud account. It can't be a federated user account.
   - **Azure password**: The password for your Azure cloud account.
5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.
6. Save the credential.

Step 2b: Add an Azure Client Secret credential to runZero

You can use this credential type to sync all resources in a single directory (across multiple subscriptions).

1. Go to the Credentials page in runZero and click *Add Credential*.
2. Provide a name for the credential, like `Azure Client Secret`.
3. Choose *Azure Client Secret* from the list of credential types.
4. Provide the following information:
   - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to *App registrations* and select the application.
   - **Azure client secret**: To generate a client secret, go to *App registrations*, select your application, go to *Certificates & secrets*, and click *+ New client secret*.
   - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to *App registrations* and select the application.
   - Select the **Access all subscriptions in this directory (tenant)** option to sync all resources in your directory. Otherwise, specify the Azure subscription ID, the

unique ID for the subscription you want to sync. To find it in the Azure portal, go to *Subscriptions* and select the subscription.

5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.

6. Save the credential.

Step 3: Choose how to configure the Entra ID integration

You can run the Entra ID integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the Entra ID integration to sync data

With your Entra ID credential saved, set up a connector task or a scan probe to sync your data.

Step 4a: Configure the Entra ID integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Entra ID-only assets.

1. Activate a connection to Entra ID. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.

2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.

3. Optionally provide a filter in the Microsoft Graph API filter syntax. runZero imports only the devices that match it.

4. Enter a name for the task, like `Entra ID sync`.

5. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.

6. Under **Task configuration**, choose the site to add your assets to.

7. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.

8. To import assets that the Entra ID account has marked as inactive, switch the **Include inactive assets** toggle to *Yes*. By default, the integration leaves out assets marked as inactive.

9. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Entra ID integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose any additional probes to include, set the **Azure AD** toggle to *Yes*, and change any default options as needed.
4. On the Credentials tab, set the **Azure AD** toggle to *Yes* for the credential you want to use.
5. Click *Initialize scan* to save the scan task. It runs immediately or at the scheduled time.

Step 5: View Entra ID assets

After a successful sync, go to your inventory to view your Entra ID assets. They show an Active Directory icon in the *Source* column.

To filter for Entra ID assets, try these queries:

- View all Entra ID assets:

  ```
  source:azuread
  ```

- View runZero assets not connected to Entra ID:

  ```
  source:runzero AND NOT source:azuread
  ```

Click into an asset to see the attributes Entra ID returned. To write your own queries, start with the search query syntax.

> The Entra ID integration also imports details about users and groups. Go to Inventory > Users or Inventory > Groups to see them.

Filtering Entra ID assets

You can apply an optional filter to Entra ID integration tasks. runZero filters assets with the Microsoft Graph `$filter` query parameter. Each expression follows the syntax `<property> [operator] <value>`, and you can combine expressions with `and` or `or` for more complex filtering.

Properties

You can filter on any property that runZero imports from Entra ID. Some examples:

Entra ID Property	runZero Attribute	Description	Example
`displayName`	@azuread.dev.displayName	The hostname of the device	EXPLORER-01
`operatingSystem`	@azuread.dev.operatingSystem	The operating system of the device	Windows
`operatingSystemVersion`	@azuread.dev.operatingSystemVersion	The version of the operating system	10.0.x
`manufacturer`	@azuread.dev.manufacturer	The manufacturer of the device	Dell Inc.
`model`	@azuread.dev.model	The model of the device	Precision 3560
`isManaged`	@azuread.dev.isManaged	Boolean value specifying whether device is managed	true, false
`managementType`	@azuread.dev.managementType	Description of how the device is managed	MDM, MicrosoftSense
`deviceOwnership`	@azuread.dev.deviceOwnership	Description of who owns the device	Company, Personal

Operators

Common operators for an Entra ID filter:

- Equal to (`eq`)
- Not equal to (`ne`)
- Has (`has`)
- Less than (`lt`)
- Greater than (`gt`)
- Less than or equal to (`le`)
- Greater than or equal to (`ge`)

Common functions, which follow the syntax `function(<property>, <value>)`:

- Starts with (`startswith`)
- Ends with (`endswith`)

Example filters

Search Filter	Description
`not(startsWith(operatingSystem, 'Android'))`	Import all assets except those with an Android operating system
`not(operatingSystem eq 'iOS') and not(operatingSystem eq 'iPad')`	Import all assets except those with an iOS or iPad operating system
`startswith(displayName, 'PROD')`	Import all devices with a hostname that starts with PROD
`not(startswith(displayName, 'DEV'))`	Import all devices except those with a hostname that starts with DEV
`deviceOwnership eq 'Company' or isManaged eq true`	Import all devices that are owned by company or that are configured as managed devices

For more on the filter syntax and additional examples, see Microsoft's Graph API documentation.

Troubleshooting

If the integration isn't working, start with these checks.

Why is the Azure Active Directory integration unable to connect?

1. Check whether the Entra ID integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Entra ID integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

How do I solve the following Entra ID errors?

- `(invalid_client) AADSTS7000218: The request body must contain the following parameter: 'client_assertion' or 'client_secret'`

    1. Enable **Allow Public Client Flows** in Entra ID. On the application details page, go to *Authentication > Advanced Settings* and toggle the **Allow Public Client Flows** setting to *Yes*.

    2. Also confirm that you granted application permissions correctly when registering the Azure application for Entra ID API access. On the API permissions settings page, check that each API/Permission has its *type* set to *application* and that the permission granted is *Grant Admin Consent for Default Directory*.

- `failed to get Entra ID groups: invalid response: 403 (403 Forbidden)`

    Check that you entered every credential value correctly and followed every step on this page. A credential problem is the likely cause.

Microsoft Intune

runZero integrates with Microsoft Intune to sync and enrich your asset inventory. With your Microsoft Intune data in runZero, unmanaged assets on your network are easier to find. The import includes the discovered apps from Intune. runZero does not currently report managed apps (those Intune pushes to devices).

Getting started

To set up the Microsoft Intune integration:

1. Configure Microsoft Intune to allow API access from runZero.
2. Add the Microsoft Intune credential in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the Microsoft Intune integration to sync your data with runZero.

Requirements

Before you set up the Microsoft Intune integration:

- Make sure you have access to the Microsoft Azure portal.

Step 1: Register an Azure application for Microsoft Intune API access

runZero can authenticate to the Microsoft Intune API with either a username and password or a client secret. Either way, start by registering an application for Microsoft Intune API access.

1. Sign in to the Microsoft Azure portal.
2. Go to *App registrations* and click *+ New registration*.
    - Provide a name.
    - Select the supported account types.
    - Optionally add a redirect URI.
3. Click *Register*.
4. Once the application is created, go back to the main Azure portal page and select *App registrations*. You should find the application you just registered (selecting the *Owned applications* tab may help). Click the app's name.
5. On the app registration overview, note the following information:
    - Application (client) ID
    - Directory (tenant) ID
6. From the application's details page, go to *Manage > API permissions* and choose *+ Add a permission*.
7. Select *Microsoft Graph* from the list of Microsoft APIs.
8. Select the permissions type for the credential you plan to use:
    - **Username & password**: select *Delegated permissions*
    - **Client secret**: select *Application permissions*
9. Search for and select these required permissions:
    - `DeviceManagementManagedDevices.Read.All`
    - `User.Read.All`
    - `DeviceLocalCredential.ReadBasic.All` (only required if you enable the Include LAPS Information option within the integration)
10. Click *Add permissions* to save the permissions to the application.
11. Click *Grant admin consent* to grant the application consent for those permissions.
12. If you're using a client secret, also do the following from the app management pages:
    - Go to *App registrations* and select the application you created.
    - Go to *Certificates & secrets* and click *+ New client secret*.
        - Enter a description.

        - Select the expiration.
    - Click *Add* to create the client secret, then save the client secret value.

Step 2: Add the Microsoft Intune credential to runZero

The Microsoft Intune credential is either an Azure username and password or an Azure Client Secret.

Step 2a: Add an Azure Username & Password credential to runZero

1. Go to the Credentials page in runZero and click *Add Credential*.
2. Provide a name for the credential, like `Azure User/Pass`.
3. Choose *Azure Username & Password* from the list of credential types.
4. Provide the following information:
    - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to *App registrations* and select the application.
    - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to *App registrations* and select the application.
    - **Azure username**: The username for your Azure cloud account. It can't be a federated user account.
    - **Azure password**: The password for your Azure cloud account.
5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.
6. Save the credential.

Step 2b: Add an Azure Client Secret credential to runZero

You can use this credential type to sync all resources in a single directory (across multiple subscriptions).

1. Go to the Credentials page in runZero and click *Add Credential*.
2. Provide a name for the credential, like `Azure Client Secret`.
3. Choose *Azure Client Secret* from the list of credential types.
4. Provide the following information:
    - **Azure application (client) ID**: The unique ID for the registered application. To find it in the Azure portal, go to *App registrations* and select the application.
    - **Azure client secret**: To generate a client secret, go to *App registrations*, select your application, go to *Manage > Certificates & secrets*, and click *New client secret*.
    - **Azure directory (tenant) ID**: The unique ID for the tenant. To find it in the Azure portal, go to *App registrations* and select the application.

5. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can configure access per organization.

6. Save the credential.

Step 3: Choose how to configure the Microsoft Intune integration

You can run the Microsoft Intune integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the Microsoft Intune integration to sync data

With your Microsoft Intune credential saved, set up a connector task or a scan probe to sync your data.

Step 4a: Configure the Microsoft Intune integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Microsoft Intune-only assets.

1. Activate a connection to Microsoft Intune. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.

2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.

3. Optionally provide a filter to import only the devices that match it. See Filtering Intune assets below for details.

4. To retrieve LAPS information from Intune, enable the Include LAPS Information option. This option has additional permission requirements; see Intune API Access above. **With this option enabled, runZero does not collect passwords from LAPS.**

5. Enter a name for the task, like `Microsoft Intune sync`.

6. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.

7. Under **Task configuration**, choose the site to add your assets to.

8. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.

9. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Microsoft Intune integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose any additional probes to include, set the Intune toggle to `Yes`, and change any default options as needed.
4. On the Credentials tab, set the Intune toggle to `Yes` for the credential you want to use.
5. Click *Initialize scan* to save the scan task. It runs immediately or at the scheduled time.

Step 5: View Microsoft Intune assets

After a successful sync, go to your inventory to view your Microsoft Intune assets. They show an Active Directory icon in the *Source* column.

To filter for Microsoft Intune assets, try these queries:

- View all Microsoft Intune assets:

  ```
  source:intune
  ```

- View runZero assets not connected to Microsoft Intune:

  ```
  source:runzero AND NOT source:intune
  ```

Click into an asset to see the attributes Microsoft Intune returned. To write your own queries, start with the search query syntax.

Filtering Intune assets

You can apply an optional filter to Intune integration tasks. runZero retrieves assets from the `DevicesWithInventory` report through the Intune reports API, which accepts a filter field to narrow the results.

Properties

The `DevicesWithInventory` report supports filtering on these properties:

- CreatedDate
- LastContact
- CategoryName
- CompliantState
- ManagementAgents
- OwnerType
- ManagementState
- DeviceType

- JailBroken
- EnrollmentType
- PartnerFeaturesBitmask

For the current list of properties, see Microsoft's documentation on the DevicesWithInventory report. Not every property in that table is filterable; a separate list just below the table shows the properties you can use in filters.

Operators

The Intune reports API has no official documentation of its filter syntax or supported operators. Based on our testing, the following operators may be supported. In practice, some of them, such as `not` and `in`, can produce errors, so your results may vary.

- Equal to (`eq`)
- Not equal to (`ne`)
- Logical negation (`not`)
- In (`in`)
- Has (`has`)
- Less than (`lt`)
- Greater than (`gt`)
- Less than or equal to (`le`)
- Greater than or equal to (`ge`)

The syntax appears similar to the `$filter` query parameter used elsewhere in Microsoft Graph, and Use the $filter query parameter has examples. Test filters carefully, and start with basic operators like `eq` and `ne` for best results.

Example filters

Search Filter	Description
`DeviceType eq 'android'`	Import all Android devices
`LastContact ge '2023-02-23 23:50:01.0000000'`	Import devices that checked in after a specific date and time

Troubleshooting

If the integration isn't working, start with these checks.

Why is the Microsoft Intune integration unable to connect?

1. Check whether the Microsoft Intune integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the

steps exactly.
3. If the Microsoft Intune integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

How do I solve the following Microsoft Intune error?

- `(invalid_client) AADSTS7000218: The request body must contain the following parameter: 'client_assertion' or 'client_secret'`

This error means you need to enable `Public Client Flows` in Azure:

1. Go to the App Registration page in the Azure portal.
2. Choose *Authentication* from the left navigation.
3. Select *Advanced Settings*.
4. Toggle the *Allow Public Client Flows* switch at the bottom of the page to *Yes*.

Miradore MDM

runZero integrates with Miradore mobile device management (MDM) to give you visibility into your mobile assets. The integration imports data from the Miradore API to enrich your asset inventory, so you can see each device's hardware, OS version, associated user, and more. It imports all enrolled devices.

Getting started

To set up the Miradore integration:

1. Sign in to your Miradore web portal and create a new API key.
2. Add the Miradore credential to runZero, which includes the endpoint hostname and API key.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the Miradore integration to sync your data with runZero.

Requirements

Before you set up the Miradore integration:

- Make sure you have administrator access to the Miradore MDM portal.

Step 1: Create a Miradore API key

1. Sign in to your Miradore portal as an administrator.
2. Click **System > Infrastructure diagram** in the left-side navigation.
3. Scroll down to the **API** node in the diagram. Click it and choose **Create API key**.

4. Name the new key and copy the secret value.

Step 2: Add the Miradore credential to runZero

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Provide a name for the credential, like `Miradore MDM`.
3. Choose **Miradore MDM API Key** from the list of credential types.
4. Provide the following information:
    - **Name**: A unique name for this credential (for example, Miradore)
    - **Miradore endpoint hostname**: The URL for your Miradore portal.
    - **Miradore API key**: The API key you created in step 1.
5. Save the credential.

Step 3: Choose how to configure the Miradore integration

You can run the Miradore integration as either a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the Miradore MDM integration to sync data

With your Miradore credential saved, set up a connector task or a scan probe to sync your data.

Step 4a: Configure the Miradore integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Miradore-only assets.

1. Activate a connection to Miradore. You can reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, make sure it has access to the organization you're currently in.
3. Enter a name for the task, like `Miradore sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
5. Under **Task configuration**, choose the site to add your assets to. runZero stores all newly discovered assets in this site.
6. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
7. Activate the connection. The sync runs on the schedule you defined, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Miradore integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.

2. Add or update the scan parameters to fit any additional requirements.
3. On the Probes and SNMP tab, choose any additional probes to include, set the Miradore toggle to *Yes*, and change any default options as needed.
4. On the Credentials tab, set the Miradore toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 5: View Miradore assets

After a successful sync, go to your inventory to view your Miradore assets. They show a Miradore icon in the **Source** column.

To filter for Miradore assets, try this query:

- View all Miradore assets:

```
source:miradore
```

Click into an asset to see the attributes the Miradore API returned. To write your own queries, start with the search query syntax.

Troubleshooting

If the integration isn't working, start with these checks.

Why is the Miradore integration unable to connect?

1. Check whether the Miradore integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Miradore integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue
4. If Miradore is on-premises, verify that the integration task runs from an Explorer with access to the Miradore host.

NetBox CMDB

runZero imports records from your NetBox configuration management database (CMDB) through the NetBox REST API to enrich your asset inventory.

NetBox limitations

runZero supports NetBox version 4. Older versions (3.x) may be compatible, but runZero does not test or support them.

NetBox data entry is free-form, so the integration may not be a good fit for your organization. We strongly recommend testing it in a Project before you configure it for a production Organization.

The integration imports Devices and Virtual Machines, along with their associated resources, but creates entities only for those two types.

Device and Virtual Machine records become runZero assets with these fields:

- Device or Virtual Machine
  - ID in the form of `device.<ID>` or `vm.<ID>`
  - Name
  - FirstSeenTS and LastSeenTS
  - Serial
  - AssetTag (Device only)
  - Interfaces
    - IP addresses for PrimaryIP / PrimaryIP4 / Primary IP6 / OOB IP
    - Additional IP addresses using the `ip<index>.field` syntax
    - NAT Inside and NAT Outside addresses
    - MAC Addresses (if entered in NetBox)
    - CreatedTS and UpdatedTS
    - ID, DNSName, Name, Comments, and Description
    - Module Name, MTU, Speed (Device only)
    - Tagged and Untagged VLANs
    - Custom Fields
  - Prefixes
    - VLAN and CIDR
    - ID, Comments, and Description
    - Role Name
    - Custom Fields
  - Location
    - ID and Name
    - Latitude and Longitude
    - Chained Location Path
    - Custom Fields
  - Rack
    - ID and Name
    - Position and Rack Face
  - Status
  - Manufacturer
    - ID, Name, and Model
    - Comments, Description, and Part Number
    - Custom Fields
  -

    - Device Role
        - ID, Name, Description
        - Custom Fields
    - Platform
        - ID & Name
        - Custom Fields
    - Site
        - ID, Name, Description, Comments
        - Facility, Physical Address, Shipping Address
        - Region, Group
        - Custom Fields
    - Virtual Chassis
        - ID, Name, Comments, and Description
        - Master ID and Master Name
        - VC Position and Priority
    - Interface Count
    - Inventory Item Count
    - Custom Fields

These top-level attributes roll up values across all sub-fields:

- Addrs
- Names
- MACs
- VLANs

NetBox asset matching

runZero matches NetBox Devices and Virtual Machines to runZero assets by Name, IP Address, and MAC Addresses. Without a direct match, runZero creates a new asset record, unless you enable the “Exclude assets that cannot be merged into an existing asset” option. NetBox data is typically static while network assets tend to change over time, so there is a good chance that many assets will not match. MAC Addresses are the most reliable way to correlate them, but many organizations leave the MAC Address fields in NetBox empty.

NetBox asset fingerprinting

For Devices, runZero uses `mfg.name` as the hardware vendor and `mfg.model` as the hardware model. For Virtual Machines, runZero reports the vendor by matching the Cluster Type field against a list of common hypervisors (Xen, VMware, Proxmox, etc.). To find an operating system match, runZero tries the `platform.name` field both with and without `mfg.name` as a prefix.

runZero treats the NetBox operating system data in `platform.name` as low-confidence relative to agent-based sources like EDRs and CMDB.

NetBox hardware values get the same treatment, with live agent-based sources taking priority.

NetBox-to-runZero custom field mapping

You can map custom fields in NetBox to attributes in runZero. These mappings override the `mfg.name`, `mfg.model`, and `platform.name` attributes described above.

In the field mapping widget of the NetBox task configuration, the “NetBox field” inputs must be the **name** properties of the custom fields in NetBox.

You can map custom fields to these runZero attributes:

- `hostnames`: The hostnames to apply to the asset in runZero. Separate multiple hostnames in the NetBox custom field with commas, and no hostname may contain tab characters. This custom field must be a `Text` type.
- `manufacturer`: The manufacturer to apply to the asset in runZero. This custom field must be a `Text` type.
- `model`: The model to apply to the asset in runZero. This custom field must be a `Text` type.
- `device_type`: The type of device to apply to the asset in runZero. This custom field must be a `Text` type.
- `first_seen_ts`: The “first seen time” to apply to the asset in runZero. This custom field must be either a `Date & time` or `Date` type.
- `os`: The type of operating system to apply to the asset in runZero. This custom field must be a `Text` type.
- `os_version`: The version of the operating system to apply to the asset in runZero. This custom field must be a `Text` type.

Usage within the runZero CLI scanner

To map custom fields with the runZero CLI scanner, pass a JSON array like this one as an argument. In each array item, the “key” is the NetBox custom field name and the “value” is the runZero attribute:

```
--netbox-field-mappings '[{"key":"netboxHostnames","value":"hostnames"},{"key":"netboxManufacturer","value":"manufactu
```

This example maps the NetBox custom field `netboxHostnames` to the runZero `hostnames` attribute and the NetBox custom field `netboxManufacturer` to the runZero `manufacturer` attribute.

Searchable time attributes

The integration creates a searchable attribute in runZero from every `Date & time` or `Date` NetBox attribute. For example, a NetBox attribute called “lastChecked” with a value of `2024-07-26T15:30:00Z` produces a new attribute called `lastCheckedTS` with a Unix epoch timestamp value of `1722017400`.

Getting started

To set up the NetBox integration, you need:

1. A NetBox instance in the cloud or reachable from one of your runZero Explorers.

2. An administrative account in the NetBox instance.
3. A v1 NetBox API key. runZero does not currently support v2 API keys, which carry the prefix `nbt_`.

Step 1: Create a NetBox API key

1. Sign in to your NetBox portal.
2. Click your name in the upper right and select **API Tokens** from the drop-down.
3. Click **Add a Token** and define a new token with no expiration and without Write permissions.
4. Give this new key a name and copy the secret value.

Step 2: Add the NetBox credential to runZero

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Provide a name for the credential, like `NetBox CMDB`.
3. Choose **NetBox API Key** from the list of credential types.
4. Fill in these fields:
    - **Name** - A unique name for this credential (for example, NetBox)
    - **NetBox URL** - The URL for your NetBox portal.
    - **NetBox API key** - The API key from step 1.
    - **Insecure** - Set this to `Yes` to attempt authentication without a verified thumbprint.
    - **NetBox thumbprints** (optional) - A set of `IP=SHA256:B64HASH` or `domain.tld=SHA256:B64HASH` pairs to trust for authentication.
        - To get the TLS thumbprint, you may scan your NetBox instance with runZero. The TLS fingerprints service attribute report lists all previously seen fingerprints.
        - If *insecure* is set to *No* and you provide no thumbprints:
            - With a self-signed certificate, the connection fails because the certificate chain cannot be verified.
            - With a valid certificate from a public CA, the connection can work without thumbprints.
5. Save the credential.

Step 3: Configure the NetBox integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new NetBox-only assets.

1. Create a new Project. We strongly recommend testing NetBox in a new Project before applying it to your production Organization.
2. Activate a connection to NetBox. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
3. Choose the credential you added earlier. If it is not listed, check that the credential has access to the organization or project you are in.
4. Enter a name for the task, like `NetBox sync`.
5. Leave the default settings to create a one-off sync task.
6. Under **Task configuration**, choose the site to add your assets to. NetBox assets that match no existing asset land in this site by default.

7. To leave out assets that runZero has not scanned, tick the **Exclude assets that cannot be merged into an existing asset** checkbox.
8. Activate the connection when you are done. The task runs, and the Tasks page tracks its status.

Step 4: View NetBox assets

After a successful sync, go to your inventory to view your NetBox assets. They show a NetBox icon in the **Source** column.

To filter to NetBox assets, run this query:

- View all NetBox assets:

```
source:netbox
```

Click an asset to see its attributes. runZero shows the attributes returned by the NetBox API.

Review the new NetBox assets and confirm that the NetBox-sourced asset count roughly lines up with your overall inventory and the total in NetBox itself.

If runZero holds far fewer NetBox-sourced assets, you may want to skip the final step until NetBox is updated with better correlation data.

Step 5: Enable unknown assets in recurring sync

Configure a new recurring NetBox task with **Exclude assets that cannot be merged into an existing asset** disabled. runZero then creates new assets for anything in NetBox it can't correlate to your asset inventory. If this creates significant duplicates, delete all assets matching the filter `source_count:1 AND source:netbox`, and re-enable the *Exclude unknown assets* option in your recurring task until the NetBox records have better correlation data (MAC addresses, hostnames, etc.).

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the NetBox integration unable to connect?

1. Check whether the NetBox integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the NetBox integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the URL
    - 404: hitting an unknown URL on the server
    - 401: not authorized, likely a credential issue
    - 403: access denied, likely a credential issue
4. If NetBox is on-premises, verify that the integration task runs from an Explorer with access to the NetBox host.

Why aren't my custom field mappings working properly?

1. Use the `name` of the custom field, not the `label`.
2. Apply the custom fields to the correct types. The NetBox integration supports only the "Device" and "Virtual Machine" types in NetBox.
3. If hostnames arrive malformed in runZero, check that multiple hostnames in the custom field are comma-separated and that the field contains no tab characters.
4. Every custom field except `first_seen_ts` must be the `Text` type in NetBox. `first_seen_ts` must be either a `Date & time` or `Date` type.

Qualys VMDR

runZero imports asset and vulnerability data from Qualys VMDR through the Qualys KnowledgeBase API.

Asset inventory

The asset inventory page has a column with the count of vulnerabilities Qualys detected for each asset. Select a single asset and the vulnerabilities table lists every result for it. The type of vulnerability scan and the import settings you select can both affect the vulnerability count.

Vulnerabilities table

The **Vulnerabilities** tab of the inventory lists every vulnerability result imported from Qualys. Selecting a result takes you to the page for the affected asset.

Severity and risk scores

Qualys assigns every vulnerability a severity rating (Minimal, Medium, Serious, Critical, Urgent). runZero normalizes these severities so the vulnerability inventory is consistent across the runZero Console.

runZero Severity	Qualys Severity
Info	1 / Minimal
Low	2 / Medium
Medium	3 / Serious
High	4 / Critical
Critical	5 / Urgent

runZero also normalizes the risk scores Qualys assigns. A risk score of 0.0 appears as `none` in the runZero Console, and every other risk score matches the assigned severity level.

Getting started with Qualys

To set up the Qualys VMDR integration, you'll need to:

1. Create or obtain user credentials with access to the Qualys API.
2. Configure CVSS scoring in Qualys.
3. Add the Qualys API username, password, and account API URL in runZero.
4. Choose whether to run the integration as a scan probe or connector task.
5. Activate the Qualys integration to pull your data into runZero.

Requirements

Before you set up the Qualys VMDR integration:

- Make sure you have access to the Qualys Cloud Platform portal.

Step 1: Add the Qualys credentials to runZero

1. Go to the Credentials page in runZero. Provide a name for the credentials, like `Qualys`.
2. Choose **Qualys Username & Password** from the list of credential types.
3. Fill in these fields:
    - **Qualys username** - the username you use to connect to the Qualys API.
    - **Qualys password** - the password for your Qualys API username.
    - **Qualys account API URL** - the URL of the Qualys API for the relevant account, in the form `https://ip:port` or `https://domain.tld:port`. This URL is unique for each Qualys user.
4. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
5. Save the credential.



The role of the Qualys API user determines which VM scanned hosts the integration can see. Managers may view all VM scanned hosts in the subscription. Auditors have no permission to view VM scanned hosts. Unit Managers may view VM scanned hosts in the user's assigned business unit. Scanners and Readers may view VM scanned hosts in the user's account, but must be assigned to the assets through asset groups in VM/VMDR.

Step 2: Choose how to configure the Qualys integration

You can run the Qualys integration as either a scan probe or a connector task. A scan probe gathers integration data during scan tasks. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 3: Set up and activate the Qualys integration to sync data

After you add your Qualys credential, sync your data with a scan probe or a connector.



The Qualys Cloud Platform limits the API calls subscription users can make. Both API controls are limited per subscription based on your service level, so the tasks this integration generates may run slowly or fail when they hit those limits. The Qualys API documentation covers these limits.

Step 3a: Configure the Qualys scan probe

Run the Qualys VMDR integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *QUALYS* option to *Yes* in the *Probes and SNMP* tab and change any default options as needed.
- Set the correct Qualys credential to *Yes* in the *Credentials* tab.
- Set the severity and risk levels for ingested vulnerability scan results (optional).

Step 3b: Configure the Qualys connector

A connector needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Qualys-only assets.

1. Activate a connection to Qualys. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they are not listed, check that the credentials have access to the organization you are in.
3. Set the severity and risk levels you want to import (optional).
4. Set the **Fingerprint only** toggle to *Yes* to ingest vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
5. Specify the tags to include in the import as a comma-separated list. Only assets that match any of the provided tags are imported (optional).
6. Specify the network IDs to include in the import as a comma-separated list. Only assets that match any of the provided network IDs are imported (optional).
7. Enter a name for the task, like `Qualys sync` (optional).
8. Choose the Explorer to run this connector task from (optional).
9. Choose the site to add your assets to. runZero stores all newly discovered assets in this site.
10. Enter a description for the task (optional).
11. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
12. To include assets that have not been assessed for vulnerabilities, switch the **Include unscanned assets** toggle to *Yes*.
13. Enter a number of hours in the `Last update filter` to exclude assets that have not been scanned within that time (optional). A blank value uses the default of 30 days.
14. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
15. Activate the connection when you are done. The sync runs on the defined schedule, and the Scheduled tasks page shows when the next sync will occur.

Step 4: View Qualys assets and vulnerabilities

After a successful sync, go to your inventory to view your Qualys assets. They show a Qualys icon in the **Source** column.

The integration also gathers the vulnerabilities Qualys detected. Go to Inventory > Vulnerabilities to view the vulnerability data from Qualys VMDR. To search and filter those results, use the vulnerability instance inventory keywords.

To filter to Qualys assets, run this query:

- View all Qualys assets:

```
source:Qualys
```

Click an asset to see its attributes. runZero shows the attributes gathered from the Qualys VMDR scan data.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Qualys integration unable to connect?

1. Check whether the Qualys integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Qualys integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue
4. If Qualys is on-premises, verify that the integration task runs from an Explorer with access to the Qualys host.

Palo Alto Networks Firewall

runZero pulls data from your Palo Alto Networks Firewalls through the PAN-OS XML API to add network visibility and context, and to improve reporting.

Getting started

To set up the Palo Alto Networks Firewall integration, you'll need to:

- Create or obtain API Keys to use with the Palo Alto Networks Firewall XML API.
- Add the Palo Alto Networks Firewall API key in runZero.
- Synchronize your Palo Alto Networks Firewall data.

Requirements

- Before you set up the Palo Alto Networks Firewall integration, make sure you have an API Key for the PAN-OS XML API.
- To use trusted authentication, scan your Palo Alto Networks Firewall with a runZero Explorer before you add the credential (optional).

Step 1: Add the Palo Alto Networks Firewall credential to runZero

1. Go to the Add credential page in runZero. Provide a name for the credentials, like **PAN-OS Firewall**.
2. Choose **Palo Alto Networks Firewall API Key** from the list of credential types.
3. Fill in these fields:
    - **Palo Alto Networks API key** - The API key to use with the Palo Alto Networks Firewall integration. First enable the XML API by following this guide: https://docs.paloaltonetworks.com/ngfw/api/api-authentication-and-security/pan-os-api-authentication. Then generate the API key by following this guide: https://docs.paloaltonetworks.com/ngfw/api/api-authentication-and-security/generate-api-key
    - **Palo Alto Networks insecure** - Set this to `Yes` to attempt authentication without a verified thumbprint.
    - **Palo Alto Networks thumbprints** (optional) - A set of `IP[:port]=SHA256:B64HASH` or `hostname.domain.tld=SHA256:B64HASH` pairs to trust for authentication.
        - To get the TLS thumbprint, scan your Palo Alto Networks firewalls with runZero. The PAN API thumbprints service attribute report lists all previously seen thumbprints.
    - **CIDR allow list** - The IP addresses allowed to receive this API Key.
4. To let all other organizations use this credential, select the Make this a global credential option. Otherwise, you can grant access per organization.
5. Save the credential.

Step 2: Synchronize Palo Alto Networks Firewall data

Enable Palo Alto Networks Firewall synchronization as part of a scan task. Any task that scans your Palo Alto Networks Firewalls can synchronize PAN-OS data.

On the Credentials tab of the scan setup, use the toggle switch to enable the Palo Alto Networks credentials you want.

When the scan runs, the Explorer uses the credentials to authenticate with any Palo Alto Networks Firewall it finds that the credentials are configured to trust. runZero imports the data it discovers about the firewalls automatically and merges it with the other information it finds by scanning.

Step 3: View Palo Alto Networks Firewall assets

To filter to Palo Alto Networks Firewall assets, run this query:

View all Palo Alto Networks Firewall assets:

```
attribute:pan.api.thumbprint
```

Click an asset to see its attributes. runZero shows the attributes returned by Palo Alto Networks.

Palo Alto Prisma Cloud

runZero imports cloud asset and vulnerability data from Palo Alto Prisma Cloud through the Prisma API for better visibility of your cloud assets and security posture. The supported Prisma cloud sources are AWS, Azure, and GCP.

Getting started with Prisma

To set up an integration with Prisma, you'll need to:

1. Create a Palo Alto Prisma Cloud credential in runZero.
2. Choose whether to run the integration as a scan probe or connector task.
3. Activate the integration to pull your data into runZero.

Step 1: Obtain your Prisma API credentials

1. Follow the Prisma documentation to create a Prisma cloud user role with sufficient permissions. See the user role descriptions for more information.
2. Obtain your Prisma Cloud API access key and secret key, which authenticate against the Prisma Cloud API. Follow the Prisma documentation to configure them properly.
3. Identify your API URL. Palo Alto sends it in your fulfillment email. See possible URL values.

Step 2: Add the Prisma credential to runZero

1. Go to the Credentials page in runZero. Provide a name for the credentials, like `Palo Alto Networks Prisma Cloud`.
2. Choose **Prisma Client Secret** from the list of credential types.
3. Create your Prisma service account from the settings page in the Prisma portal, then fill in these fields:
    - **Prisma Cloud Access Key** - The access key you obtained in the steps above.
    - **Prisma Cloud Secret Key** - The secret key you obtained in the steps above.
    - **Prisma API URL** - The API Endpoint URL for reaching the Prisma API.
4. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.

5. Save the credential.

Step 3: Choose how to configure the Prisma integration

You can run the Prisma integration as either a scan probe or a connector task. A scan probe gathers integration data during scan tasks. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

After you add your Prisma credential, sync your data from Prisma with a connector task or a scan probe.

Step 4a: Configure the Prisma integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Prisma-only assets.

1. Activate a connection to Prisma. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they are not listed, check that the credentials have access to the organization you are in.
3. Set the **Exclude importing vulnerabilities** toggle to control whether vulnerability records are saved.
4. Set the **Exclude importing software** toggle to control whether software records are saved.
5. Set the **Exclude assets that can not be merged into an existing asset** toggle to enrich existing assets only, with no new asset records created.
6. Set the severity and risk levels you want to import.
7. Enter a name for the task, like `Prisma Sync` (optional).
8. In the Run task with section, choose the Explorer or Explorer Group to run this connector task from (optional).
9. Choose the site to add your assets to. runZero stores all newly discovered assets in this site.
10. Enter a description for the task (optional).
11. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
12. Activate the connection when you are done. The sync runs on the defined schedule, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the Prisma integration as a scan probe

Run the Prisma integration as a scan probe to have the runZero Explorer pull your Prisma assets into the runZero Console.

In a new or existing scan configuration:

- Set the *Prisma* option to *Yes* in the *Probes and SNMP* tab and change any default options as needed.

- Set the severity and risk levels for ingested vulnerability results (optional).
- Set the correct *Prisma* credential to *Yes* in the *Credentials* tab.

Step 5: View Prisma assets and vulnerabilities

After a successful sync, go to your inventory to view your Prisma assets. They show a Prisma icon in the **Source** column.

The integration also gathers the vulnerabilities Prisma detected. Go to **Inventory** > **Vulnerabilities** to view the vulnerability data from Prisma. To search and filter those results, use the vulnerability instance inventory keywords.

To filter to Prisma assets, run this query:

- View all Prisma assets:

  ```
  source:Prisma
  ```

Click an asset to see its attributes. runZero shows the attributes gathered from Prisma Cloud.

Troubleshooting

If you see authentication or access errors when the integration runs, the API URL on your credential is likely incorrect. Check the fulfillment email you received from Palo Alto. It contains the correct API URL.

Rapid7

runZero integrates with Rapid7's InsightVM and Nexpose to enrich your asset inventory and show you the vulnerabilities detected in your environment.

Asset inventory

The asset inventory page has a column with the count of vulnerabilities Rapid7 detected for each asset. Select a single asset and the vulnerabilities table lists every result for it. The type of vulnerability scan and the import settings you select can both affect the vulnerability count.

Vulnerabilities table

The **Vulnerabilities** tab of the inventory lists every vulnerability result imported from Rapid7. Selecting a result takes you to the page for the affected asset.

Severity and risk scores

Rapid7 assigns every vulnerability a severity rating (Moderate, Severe, or Critical) based on its CVSSv2 score. runZero normalizes these severities so the vulnerability inventory is consistent across the runZero Console.

runZero Severity	Rapid7 Severity	CVSS Range
Info	Moderate	0.0
Medium	Moderate	0.1 - 3.4
High	Severe	3.5 - 7.4
Critical	Critical	7.5 - 10.0

runZero also normalizes the risk scores Rapid7 assigns. A risk score of 0.0 appears as `none` in the runZero Console, and every other risk score matches the assigned severity level.

Rapid7 InsightVM

runZero imports asset and vulnerability data from Rapid7 InsightVM through the InsightVM API.

runZero supports both Rapid7 InsightVM Cloud and on-premises InsightVM.

On-premises InsightVM

For an on-premises installation, run the InsightVM connector as a scan probe from a runZero Explorer with network access to the InsightVM deployment.

To set up the InsightVM integration, you'll need to:

1. Create or obtain user credentials to use with the InsightVM API.
2. Add the InsightVM API username, password, and API URL in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the InsightVM integration to pull your data into runZero.

Requirements

Before you set up the InsightVM integration:

- Obtain credentials for an InsightVM user with administrator access to the InsightVM portal.
- To use trusted authentication, scan your InsightVM with a runZero Explorer (optional).

Step 1: Add the InsightVM credentials to runZero

1. Create a new credential from the runZero Credentials page.
2. Provide a name for the credentials, like `InsightVM`.
3. Choose **InsightVM Username & Password** from the list of credential types.
4. Fill in these fields:
    - **InsightVM username** - The username you use to connect to the InsightVM API. This account needs the `User` role or greater permissions in InsightVM.
    - **InsightVM password** - The password for your InsightVM username.
    - **InsightVM API URL** - The URL of your InsightVM API instance, in the form `https://ip:3780` or `https://domain.tld:3780`. By default, the InsightVM API uses port

3780. runZero appends `api/3/sites?size=500` to this URL automatically.
    - **InsightVM insecure** - Set this to `Yes` to attempt authentication without a verified thumbprint.
    - **InsightVM thumbprints** (optional) - A set of `IP=SHA256:B64HASH` or `domain.tld=SHA256:B64HASH` pairs to trust for authentication.
        - To get the TLS thumbprint, scan your InsightVM instance with runZero. The TLS fingerprints service attribute report lists all previously seen fingerprints.
        - If *InsightVM insecure* is set to *No* and you provide no thumbprints:
            - With a self-signed certificate, the connection fails because the certificate chain cannot be verified.
            - With a valid certificate from a public CA, the connection can work without thumbprints.
5. To let all other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
6. Save the credential.

Step 2: Choose how to configure the Rapid7 integration

You can run the Rapid7 InsightVM integration as either a scan probe or a connector task. A scan probe gathers integration data during scan tasks. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync. A scan probe is the right option for most users. A connector works only if you self-host runZero or your InsightVM instance is publicly accessible.

Step 3: Sync your InsightVM data

After you add your InsightVM credential, activate the integration to sync your data.

Step 3a: Configure the InsightVM scan probe

Run the InsightVM integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *INSIGHTVM* option to *Yes* in the *Probes and SNMP* tab and change any default options as needed.
- Set the correct InsightVM credential to *Yes* in the *Credentials* tab.
- Set the severity and risk levels for ingested vulnerability scan results (optional).
- Enter a Site Name Filter to filter results by InsightVM site using regular expressions (optional).

Step 3b: Configure the InsightVM connector

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new InsightVM-only assets.

1. Activate a connection to InsightVM. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they are not listed, check that the credentials have access to the organization you are in.

3. Set the severity and risk levels you want to import (optional).
4. Enter a Site Name Filter to limit your results to specific sites (optional). The filter is a regular expression and supports the re2 syntax.
5. Set the **Fingerprint only** toggle to *Yes* to ingest vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
6. Enter a name for the task, like `InsightVM sync` (optional).
7. Choose the Explorer to run this connector task from (optional).
8. Choose the site to add your assets to. runZero stores all newly discovered assets in this site.
9. Enter a description for the task (optional).
10. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
11. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
12. Activate the connection when you are done. The sync runs on the defined schedule, and the Scheduled tasks page shows when the next sync will occur.

Step 4: View InsightVM assets and vulnerabilities

After a successful sync, go to your inventory to view your InsightVM assets. They show a Rapid7 icon in the **Source** column.

The integration also gathers the vulnerabilities InsightVM detected. Go to Inventory > Vulnerabilities to view the vulnerability data from InsightVM. To search and filter those results, use the vulnerability instance inventory keywords.

To filter to Rapid7 assets, run this query:

- View all Rapid7 assets:

  ```
  source:Rapid7
  ```

Click an asset to see its attributes. runZero shows the attributes gathered from the Rapid7 scan data.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Rapid7 InsightVM integration unable to connect?

1. Check whether the Rapid7 InsightVM integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3.

3. If the Rapid7 InsightVM integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

How can I get a TLS thumbprint for the InsightVM credential?

These example commands calculate your TLS thumbprint, where <IP address> is the IP address of your InsightVM instance:

```
$ (echo|openssl s_client -connect <IP address>:3780 -showcerts 2>/dev/null) | openssl x509 -inform PEM -outform DER |
```

The commands return a value like this: `X1NWttnkIQprK6zSre/VweKpbRlj7Dt4M6hNfUacytE=` Use the following entry for the corresponding TLS thumbprint: 192.168.0.3:3780=SHA256:X1NWttnkIQprK6zSre/VweKpbRlj7Dt4M6hNfUacytE=

InsightVM Cloud

The Rapid7 InsightVM Cloud integration enriches your runZero inventory with data from the Rapid7 InsightVM Cloud service.

To set up the Rapid7 InsightVM Cloud integration, you'll need to:

1. Create or obtain a Rapid7 Platform API Key to use with the Insight API.
2. Create a credential in runZero with the Rapid7 Platform API Key.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the integration to pull your data into runZero.

Step 1: Add the Rapid7 Platform API credentials to runZero

1. Create a new credential from the runZero Credentials page.
2. Provide a name for the credential, like `Rapid7 InsightVM Cloud`.
3. Choose **InsightVM Cloud API Key** from the list of credential types.
4. Fill in these fields:
    - **InsightVM API Key** - The API key for your InsightVM account.
    - **InsightVM Host** - The host of your InsightVM API instance. Use the hostname only, for example, "us2.api.insight.rapid7.com".
5. To let all other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
6. Save the credential.

Step 2: Choose how to configure the Rapid7 integration

You can run the Rapid7 InsightVM integration as either a scan probe or a connector task. A scan probe gathers integration data during scan tasks. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync. Most use cases work fine as a connector task.

Step 3: Sync your InsightVM data

After you add your InsightVM credential, activate the integration to sync your data.

Step 3a: Configure the InsightVM Cloud scan probe

Run the InsightVM integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *INSIGHTVMCLOUD* option to *Yes* in the *Probes and SNMP* tab and change any default options as needed.
- Set the correct InsightCloud credential to *Yes* in the *Credentials* tab.
- Set the severity and risk levels for ingested vulnerability scan results (optional).

Step 3b: Configure the InsightVM connector

A connector needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new InsightVM-only assets.

1. Activate a connection to InsightVM Cloud. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it is not listed, check that the credential has access to the organization you are in.
3. Set the severity and risk levels you want to import (optional).
4. Enter a name for the task, like `InsightVM Cloud sync` (optional).
5. Choose the Explorer to run this connector task from (optional).
6. Choose the site to add your assets to. runZero stores all newly discovered assets in this site.
7. Enter a description for the task (optional).
8. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
9. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.
10. Activate the connection when you are done. The sync runs on the defined schedule, and the Scheduled tasks page shows when the next sync will occur.

Nexpose

runZero integrates with Rapid7 Nexpose by importing report files exported from your Nexpose instance.

Getting started with Rapid7 Nexpose

To use the Rapid7 Nexpose integration, you'll need to:

1. Download an XML Export or XML Export 2.0 report from Nexpose.
2. Import the Nexpose files through the inventory pages.

Requirements

Before you set up the Nexpose integration:

- Make sure you have access to the Nexpose portal.

Step 1: Export Nexpose vulnerability scan report

1. Sign in to Nexpose with the account you use for the runZero integration.
2. Go to the Reports page and select *Create a report*.
3. From the Export tab, select either XML Report or XML Report 2.0.
4. Set the scan, asset, asset group, or site scope.
5. Click *Save & Run the Report*.
6. When the report completes, save it to a local file.

Step 2: Import the Nexpose files into runZero

1. Go to the Inventory page in runZero.
2. Choose **Import** > **Nexpose XML Export (.xml)** from the list of import types.
3. On the import data page:
    - Choose the site to add your assets to.
    - Set tags to apply to the imported assets (optional).
    - Set the severity and risk levels to ingest (optional).
    - Set the **Fingerprint only** toggle to *Yes* to ingest vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).

Step 3: View Nexpose assets and vulnerabilities

After a successful import, go to your inventory to view your Nexpose assets. They show a Rapid7 icon in the **Source** column.

The integration also gathers the vulnerabilities Nexpose detected. Go to Inventory > Vulnerabilities to view the vulnerability data from Nexpose. To search and filter those results, use the vulnerability instance inventory keywords.

To filter to Rapid7 assets, run this query:

- View all Rapid7 assets:

  ```
  source:Rapid7
  ```

Click an asset to see its attributes. runZero shows the attributes gathered from the Nexpose scan file.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the Rapid7 Nexpose integration unable to connect?

1. Check whether the Rapid7 Nexpose integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Rapid7 Nexpose integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue
4. Verify that the integration task runs from an Explorer with access to the Nexpose host.

SentinelOne

runZero imports SentinelOne data through the SentinelOne API to sync and enrich your asset inventory, import the software installed on assets, and import the vulnerabilities affecting that software. With SentinelOne data in runZero, vulnerable endpoints and endpoints missing required software are easier to find.



runZero treats any IP address reported by SentinelOne as a secondary address rather than a primary address, because these IPs can be stale and may not belong to a specific network or site.

Getting started

To set up the SentinelOne integration, you'll need to:

1. Configure SentinelOne to allow API access to runZero.
2. Add the SentinelOne API key and SentinelOne base API URL in runZero.
3. Choose whether to run the integration as a scan probe or connector task.
4. Activate the SentinelOne integration to sync your data with runZero.

Requirements

Before you set up the SentinelOne integration:

- Make sure you have access to the SentinelOne admin portal.

Step 1: Configure SentinelOne to allow API access to runZero

1. Sign in to SentinelOne with the account you use for the runZero integration.
2. Go to **User > My User**.
3. **Generate** the API token, then download or copy it. This API key expires every six months and must be regenerated.

Step 2: Add the SentinelOne credential to runZero

1. Go to the Credentials page in runZero. Provide a name for the credentials, like `SentinelOne`.
2. Choose **SentinelOne API key** from the list of credential types.
3. Fill in these fields:
    - **SentinelOne API URL** - Your organization-specific base URL, which depends on your account type and looks like `organization.sentinelone.net`.
    - **SentinelOne API key** - The API key from **User > My User** in your SentinelOne portal, where you can generate, regenerate, or revoke a key.
4. To let other organizations use this credential, select the *Make this a global credential* option. Otherwise, you can grant access per organization.
5. Save the credential.

Step 3: Choose how to configure the SentinelOne integration

You can run the SentinelOne integration as either a scan probe or a connector task. A scan probe gathers integration data during scan tasks. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the SentinelOne integration to sync data

After you add your SentinelOne credential, set up a connector task or scan probe to sync your data.

Step 4a: Configure the SentinelOne integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new SentinelOne-only assets.

1. Activate a connection to SentinelOne. You can reach every available third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they are not listed, check that the credentials have access to the organization you are in.
3. Enter a name for the task, like `SentinelOne sync`.
4. Schedule the sync to run once or on a recurring schedule. The schedule starts on the date and time you set.

5. Under **Task configuration**, choose the site to add your assets to.
6. To skip importing software, switch the **Import Software** toggle to *No*.
7. To skip importing vulnerabilities, switch the **Import Vulnerabilities** toggle to *No*.
8. With the **Import Vulnerabilities** toggle set to *Yes*, use the **Severities** checkboxes to choose which vulnerability severities to import.
9. To leave out assets that runZero has not scanned, switch the **Exclude unknown assets** toggle to *Yes*. By default, the integration includes them.
10. Activate the connection when you are done. The sync runs on the defined schedule, and the Scheduled tasks page shows when the next sync will occur.

Step 4b: Configure the SentinelOne integration as a scan probe

1. Create a new scan task or select a future or recurring scan task from your Tasks page.
2. Add or update the scan parameters to match any additional requirements.
3. On the Probes and SNMP tab, choose which additional probes to include, set the SentinelOne toggle to *Yes*, and change any default options as needed.
4. On the Credentials tab, set the toggle to *Yes* for the SentinelOne credential you want to use.
5. Click **Initialize scan** to save the scan task. It runs immediately or at the scheduled time.

Step 5: View SentinelOne assets and software

After a successful sync, go to your inventory to view your SentinelOne assets. They show a SentinelOne icon in the **Source** column.

The integration also gathers details about installed software. Go to Inventory > Software to view the software data from SentinelOne. To search and filter that software, use the software instance inventory keywords.

To filter to SentinelOne assets, run one of these queries:

- View all SentinelOne assets:

  ```
  source:SentinelOne
  ```

- Find assets that have a SentinelOne agent installed:

  ```
  edr.name:SentinelOne
  ```

- Find Windows assets, excluding servers, that are missing a SentinelOne agent:

  ```
  os:windows and not type:server and not edr.name:SentinelOne
  ```

Click an asset to see its attributes. runZero shows the attributes returned by the SentinelOne API, except policies.

Troubleshooting

If you are having trouble with this integration, the questions and answers below may help.

Why is the SentinelOne integration unable to connect?

1. Check whether the SentinelOne integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the SentinelOne integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue
4. If SentinelOne is on-premises, verify that the integration task runs from an Explorer with access to the SentinelOne host.

Shodan

runZero pulls your externally facing assets and services from Shodan through the Shodan API for better visibility of your internet footprint and cyber hygiene.

Getting started

To set up the Shodan integration:

1. Add your Shodan API key to runZero.
2. Choose whether to run the integration as a scan probe or a connector task.
3. Activate the integration to sync your data into runZero.

Requirements

- A Shodan account with a license that fits your needs.

Step 1: Add the Shodan credential to runZero

1. Go to the new credential page in runZero and name the credential, for example `Shodan`.
2. Choose **Shodan Search API key** from the list of credential types.
3. Enter your **Shodan Search API key**. The key is on the Account page of the Shodan portal, where you can also reset it.
4. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
5. Save the credential.

Step 2: Choose how to configure the Shodan integration

You can run the Shodan integration as a scan probe or a connector task. A scan probe gathers Shodan data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 3: Set up and activate the Shodan integration to sync data

With the credential saved, set up a connection or a scan probe to sync your data from Shodan.

Step 3a: Configure the Shodan integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Shodan-only assets.

1. Activate a connection to Shodan. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credential you added earlier. If it isn't listed, check that it has access to your current organization.
3. Either enter a Shodan search in Shodan's search syntax, or let runZero generate a search for the public IP addresses of all live assets.
4. Optionally, use the "Ignore Shodan tags" input to enter a comma-separated list of tags to exclude from Shodan assets.
5. Enter a name for the task, like `Shodan sync`.
6. Schedule the sync to run once or on a recurring schedule, starting at the date and time you set.
7. Under **Task configuration**, choose the site for your assets.
8. To leave out assets runZero hasn't scanned, switch the **Exclude unknown assets** toggle to *Yes*. The import includes them by default.
9. Activate the connection. The sync runs on your schedule, and the tasks page shows when the next sync will run.

Step 3b: Configure the Shodan integration as a scan probe

1. Create a new scan task, or pick a future or recurring one from your Tasks page.
2. Add or update the scan parameters as needed.
3. On the Probes and SNMP tab, choose any other probes to include, set the Shodan toggle to *Yes*, and adjust the default options if needed. As with a connector task, you can enter a Shodan search string directly or choose *assets* mode to have runZero generate a search for the public IP addresses of all live assets.
4. On the Credentials tab, set the Shodan toggle to *Yes* for the credential you want to use.
5. Click **Initialize scan** to save the task. It runs immediately or at its scheduled time.

Step 4: View Shodan assets and services

After a successful sync, your Shodan assets appear in your inventory with a Shodan icon in the **Source** column.

The integration also gathers service details. Go to Inventory > Services to view the service data from Shodan.

To filter by Shodan assets or services, run these queries:

- View all Shodan assets:

  ```
  source:shodan
  ```

- View all Shodan services:

  ```
  source:shodan
  ```

Open an asset or service to see the attributes runZero pulled from the Shodan Search API.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the Shodan integration unable to connect?

1. Check whether the Shodan integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Shodan integration can't connect, check the task log for errors. Common errors include:
   - 500: server error, unable to connect to the endpoint
   - 404: hitting an unknown endpoint on the server
   - 403: not authorized, likely a credential issue

Tailscale

runZero imports the devices in your Tailscale tailnet, including their addresses, operating system, hostnames, tags, and the Tailscale account that owns each node. Tailscale is the first integration runZero ships as a platform-packaged Starlark integration. It runs on the same engine as custom integration scripts, with the script embedded in the product, so you configure it like any other connector and never edit a script.

Getting started

To set up the Tailscale integration:

1. Create a Tailscale API access token, or an OAuth client, in the Tailscale admin console.
2. Add the Tailscale credential to runZero.
3. Set up and activate the connector task that syncs your data into runZero.

Requirements

- Access to the Tailscale admin console with permission to create API access tokens or OAuth clients.
- The runZero custom integrations entitlement.
- An Explorer running version 5.0.260723.0 or newer, if the task runs on an Explorer. The Tailscale API is reachable from the cloud, so the task can also run without one.

Step 1: Create a Tailscale API token or OAuth client

The integration accepts either kind of credential:

- For an **API access token**, go to **Settings > Keys** in the Tailscale admin console and generate one (it starts with `tskey-api-`). Tokens expire, so prefer an OAuth client for a scheduled sync.
- For an **OAuth client**, go to **Settings > OAuth clients** in the Tailscale admin console and create a client with at least the `devices:core:read` scope. Copy the client ID and client secret.

Step 2: Add the Tailscale credential to runZero

1. Go to the Credentials page in runZero and click **Add Credential**.
2. Choose **Tailscale Settings & Secrets** from the list of credential types.
3. Fill in the fields:
    - Leave **Tailscale API URL** at the default `https://api.tailscale.com` unless you proxy the API.
    - Set **Tailnet** to the tailnet ID or name, for example `T1234CNTRL` or `example.com`.
    - Enter your client ID in **OAuth client ID**, or leave it blank to use a plain API key.
    - Set **API key / OAuth client secret** to the API access token, or to the OAuth client secret when a client ID is set.
4. Save the credential.

Step 3: Set up and activate the Tailscale integration

1. Activate a connection to Tailscale. You can also reach it from the Tailscale card on the integrations page.

2. Choose the credential you added earlier. If it isn't listed, check that it has access to your current organization.
3. Enter a name for the task, like `Tailscale sync`.
4. Schedule the sync and choose the site for newly discovered assets.
5. To stop the sync from creating assets runZero hasn't seen any other way, enable **Exclude assets that cannot be merged into an existing asset**.
6. Select an Explorer, or leave the selection empty to run the sync from the runZero cloud.
7. Activate the connection. The sync runs on the schedule you set.

Step 4: View Tailscale assets

After a successful sync, your Tailscale devices appear in your inventory with the Tailscale icon in the **Source** column. Their attributes live under `tailscale.device.*`, and runZero resolves the Tailscale account that owns each node into asset ownership. Managing ownership explains the owner types and how to assign an owner to an asset.

To filter by Tailscale assets:

```
source:tailscale
```

Tanium API Gateway

runZero imports data from the Tanium Gateway API to sync your endpoints, applications, and vulnerabilities from Tanium for better visibility over your network.

Getting started with Tanium

To set up the Tanium integration:

1. Generate an API token with the permissions the integration needs.
2. Add the Tanium credential to runZero.
3. Choose whether to run the integration as a scan probe or a connector task.
4. Activate the integration to pull your data into runZero.

Step 1: Generate an API key in Tanium Dashboard

1. Sign in to Tanium and go to **Administration > Roles**.
2. Create a role with the permissions the integration needs:
    1. Search for the **Gateway User** role.
    2. Select it and click the **Clone** button that appears.
    3. On the **Clone Role** screen, enable **Platform Content Permissions > Sensor > Read** and add these Content Sets with the **n+** button beside the green check:
        - Base
        - Comply
        - Comply Reporting
        - Core AD Query Content
        - Core Content

        - Reserved
        - Tanium Data Service
    4. Save the role.
3. Go to **Administration > Personas** and click **New Persona** to create a persona with the role you just created:
    1. Name the persona.
    2. Under **Manage Roles**, search for and apply your new role.
    3. Under **Computer Groups**, add the groups you need, or check **Unrestricted Management Rights** to allow access to all Computer Groups.
    4. Assign a user or service account that has the permissions granted to the persona.
    5. Save the persona.
4. Go to **Administration > API Tokens** and click **New API Token**.
    1. Enter a name and select a TTL.
    2. Select the persona you just created from the dropdown (you may need to refresh the page for it to appear).
    3. Enter the IP addresses to allow requests from:
        - If you run the integration from an Explorer or the CLI, enter the IP addresses or ranges of your hosts.
        - Otherwise, enter `0.0.0.0/0`.
    4. Save the API token.

Step 2: Add the Tanium API token to runZero

1. Go to the Credentials page in runZero.
2. Choose **Tanium API Token** from the list of credential types.
3. Name the credential, for example `Tanium`.
4. Fill in the fields:
    - Enter your Tanium API Gateway URL in **Tanium API URL**. The full URL looks like `https://<customername>-api.cloud.tanium.com/plugin/products/gateway/graphql`. If you leave off the path (`/plugin/products/gateway/graphql`), runZero adds it when it calls the API.
    - In **Tanium API token**, enter the token (including the `token-` prefix) you created in step 1.
    - Enable **Insecure** to allow authentication with untrusted endpoints. It turns off certificate validation, so use it with caution.
5. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
6. Verify and save the credential.

Step 3: Choose how to configure the Tanium integration

You can run the Tanium integration as a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

Step 4a: Configure the Tanium integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Tanium-only assets.

1. Activate a connection to Tanium. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they aren't listed, check that they have access to your current organization.
3. Optionally, enter a comma-separated list of computer groups to include. runZero then imports data only for those groups.
4. Enter a name for the task, like `Tanium Sync` (optional).
5. Choose the Explorer that runs this connector task (optional).
6. Choose the site for your assets. runZero stores all newly discovered assets there.
7. Enter a description for the task (optional).
8. To leave out assets runZero hasn't scanned, switch the **Exclude unknown assets** toggle to *Yes*. The import includes them by default.
9. Schedule the sync to run once or on a recurring schedule, starting on the date and time you set.
10. Activate the connection. The sync runs on your schedule, and the Scheduled tasks page shows when the next one will run.

Step 4b: Configure the Tanium integration as a scan probe

Run the Tanium integration as a scan probe to have the runZero Explorer pull your Tanium assets into the runZero Console.

In a new or existing scan configuration:

- Set the *TANIUM* option to *Yes* in the *Probes and SNMP* tab and change any default options you need.
- Set the correct *TANIUM* credential to *Yes* in the *Credentials* tab.

Step 5: View Tanium assets

After a successful sync, your Tanium assets appear in your inventory with a Tanium icon in the **Source** column.

To filter by Tanium assets, run this query:

- View all Tanium assets:

  ```
  source:Tanium
  ```

Open an asset to see the attributes runZero gathered from Tanium.

Tenable

runZero integrates with Tenable Vulnerability Management (previously Tenable.io), Tenable Nessus, and Tenable Security Center to enrich your asset inventory and show you the vulnerabilities detected in your environment. The Tenable Vulnerability Management, Nessus Professional, and Tenable Security Center integrations pull data from the Tenable API. All versions of Tenable Nessus and Tenable Security Center (previously Tenable.sc) also work through Nessus v2 file imports (`.nessus`).

The API integration currently supports only the main Tenable Vulnerability Management cloud endpoint at `https://cloud.tenable.com`.

Asset inventory

The asset inventory page has a column with the count of vulnerabilities Tenable detected on each asset. Select an asset, and its vulnerabilities table lists every result for that asset. The type of vulnerability scan and your import settings can both affect the count.

Vulnerabilities table

The **Vulnerabilities** tab of the inventory lists every vulnerability result imported from Tenable. Selecting a result opens the page for the affected asset.

Severity and risk scores

Tenable describes vulnerability severity with the third-party Common Vulnerability Scoring System (CVSS) values from the National Vulnerability Database (NVD). Every vulnerability gets a severity rating (Info, Low, Medium, High, or Critical) based on its static CVSSv2 or CVSSv3 score. Tenable uses CVSSv2 by default, though you can opt for CVSSv3 instead. runZero displays the CVSSv2-based severity in the Inventory and Asset views.

Tenable also calculates a dynamic Vulnerability Priority Rating (VPR) for most vulnerabilities to supplement the severity rating. Where the CVSS score is static, Tenable updates the VPR to reflect current threats. VPR values range from 0.1-10.0, and a higher value means a higher likelihood of exploit. runZero displays the VPR-based risk in the Inventory and Asset views. Tenable's documentation has more on how it handles severity and risk.

Tenable Vulnerability Management

runZero imports data from Tenable Vulnerability Management (previously Tenable.io) through the Tenable API.

Getting started with Tenable Vulnerability Management

To set up the Tenable Vulnerability Management integration:

1. Create an Administrator API key in an access group with `Can View` permission to `Manage Assets`. To retrieve agent health data, the key also needs the Scan Manager role.
2. Add the Tenable Vulnerability Management credential to runZero.
3. Choose whether to run the integration as a scan probe or a connector task.
4. Activate the integration to pull your data into runZero.

Requirements

- Administrator access to the Tenable portal.

Step 1: Create an Administrator API key

1. Sign in to Tenable Vulnerability Management with the Administrator account you're using for the runZero integration.
2. Go to **My Profile** > **My Account** > **API Keys**.
3. Click **Generate**, then download or copy the API token.

Step 2: Add the Tenable credential to runZero

1. Go to the Credentials page in runZero and name the credential, for example `Tenable Vulnerability Management`.
2. Choose **Tenable.io Access & Secret** from the list of credential types.
3. Generate your Tenable access and secret keys from your account page in the Tenable portal, then fill in the fields:
    - Enter your 64-character Tenable access key in **Access key**.
    - Enter your 64-character Tenable secret key in **Secret key**.
4. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
5. Save the credential.

Step 3: Choose how to configure the Tenable integration

You can run the Tenable Vulnerability Management integration as a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync. A connector task works if you're self-hosting runZero or integrating with Tenable Vulnerability Management.

Step 4: Set up and activate the integration to sync data

Step 4a: Configure the Tenable integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Tenable-only assets.

1. Activate a connection to Tenable Vulnerability Management. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.

2. Choose the credentials you added earlier. If they aren't listed, check that they have access to your current organization.
3. To leave out assets runZero hasn't scanned, check **Exclude assets that cannot be merged into an existing asset**. The import includes them by default.
4. To include assets that haven't been assessed for vulnerabilities, check **Include assets that have not been assessed for vulnerabilities**.
5. To skip asset agent health data, check **Disable importing asset agent data**.
6. Check **Disable importing software from the Tenable inventory** to use software records for fingerprint analysis without storing them in your runZero software inventory (optional).
7. Check **Disable importing vulnerabilities from the Tenable inventory** to use vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
8. Set the severity and risk levels to import (optional). Much of the host information from Tenable comes from Info-level plugins, so if you import only higher severities, assets runZero hasn't scanned may show little information.
9. Optionally, enter a comma-separated list of tags to include, in the format `category:value`. runZero imports assets that match at least one of the tags.
10. To filter by asset source, set **Filter by asset source** to *All supported sources* or to a combination of sources from the checklist.
11. Enter a name for the task, like `Tenable Vulnerability Management sync` (optional).
12. Choose the Explorer that runs this connector task (optional).
13. Choose the site for your assets. runZero stores all newly discovered assets there.
14. Enter a description for the task (optional).
15. Schedule the sync to run once or on a recurring schedule, starting on the date and time you set.
16. Activate the connection. The sync runs on your schedule, and the Scheduled tasks page shows when the next one will run.

Step 4b: Configure the Tenable integration as a scan probe

Run the Tenable Vulnerability Management integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *TENABLE* option to *Yes* in the *Probes and SNMP* tab and change any default options you need.
- Optionally, set the severity and risk levels for ingested vulnerability scan results.
- Set the correct `Tenable` credential to *Yes* in the *Credentials* tab.

Step 5: View Tenable assets and vulnerabilities

After a successful sync, your Tenable assets appear in your inventory with a Tenable icon in the **Source** column.

The integration also gathers the vulnerabilities Tenable detected. Go to **Inventory** > **Vulnerabilities** to view the vulnerability data from Tenable Vulnerability Management.

To filter by Tenable assets, run this query:

- View all Tenable assets:

  ```
  source:Tenable
  ```

Open an asset to see the attributes runZero gathered from the Tenable scan data.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the Tenable Vulnerability Management integration unable to connect?

1. Check whether the Tenable Vulnerability Management integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Tenable Vulnerability Management integration can't connect, check the task log for errors. Common errors include:
   - 500: server error, unable to connect to the endpoint
   - 404: hitting an unknown endpoint on the server
   - 403: not authorized, likely a credential issue
4. If Tenable Vulnerability Management is on-premises, verify that the integration task runs from an Explorer with access to the Tenable host.

How do I solve the following error in Tenable Vulnerability Management:

```
"error-message":"no tenable assets match import criteria",
"level":"error","msg":"could not load scan result data to writer"
```

Enabling the *Include Unscanned Assets* toggle in the Tenable task configuration usually resolves this error, which Tenable returns intermittently. The toggle disables the filters runZero applies for live assets scanned in the last 30 days.

Nessus Professional

runZero imports data from Nessus Professional through the Tenable API.

Getting started with Nessus Professional

To set up the Nessus Professional integration:

1. Create an Administrator API key in an access group with *Can View* permission to *Manage Assets*.

2. Add the Nessus Professional credential to runZero.
3. Choose whether to run the integration as a scan probe or a connector task.
4. Activate the integration to pull your data into runZero.

Requirements

- Administrator access to the Nessus Professional portal.

Step 1: Create an Administrator API key

1. Sign in to Nessus Professional with the Administrator account you're using for the runZero integration.
2. Go to **My Profile** > **My Account** > **API Keys**.
3. Click **Generate**, then download or copy the API token.
    - When you create the runZero credential, either configure it to skip TLS verification or provide the TLS thumbprint.

Step 2: Add the Nessus Professional credential to runZero

1. Go to the Credentials page in runZero and name the credential, for example `Nessus Professional`.
2. Choose **Nessus Professional Access & Secret** from the list of credential types.
3. Generate your Tenable access and secret keys from your account page in the Tenable portal, then fill in the fields:
    - Enter your 64-character Tenable access key in **Access key**.
    - Enter your 64-character Tenable secret key in **Secret key**.
    - Enter the API URL of your Nessus Professional instance in **Nessus API URL**, in the form `https://ip:port` or `https://domain.tld:port`. The default Nessus Professional port is `8834`.
    - Set **Nessus insecure** to *Yes* to attempt authentication without a verified thumbprint.
    - Optionally, enter a set of IP=SHA256:B64HASH pairs to trust for authentication in **Nessus thumbprints**.
        - To get the TLS thumbprint, scan your Nessus instance with runZero. The TLS fingerprints service attribute report lists every fingerprint runZero has seen.
    - If *Nessus insecure* is *No* and you provide no thumbprints:
        - With a self-signed certificate, the connection fails because the certificate chain can't be verified.
        - With a valid certificate from a public CA, the connection can work without thumbprints.
4. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
5. Save the credential.

Step 3: Choose how to configure the Nessus Professional integration

You can run the Nessus Professional integration as a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the

cloud or from one of your Explorers, and performs only the integration sync. A connector task works if you're self-hosting runZero or your Nessus Professional instance is publicly accessible.

Step 4: Set up and activate the integration to sync data

Step 4a: Configure the Nessus Professional integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Tenable-only assets.

1. Activate a connection to Nessus Professional. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they aren't listed, check that they have access to your current organization.
3. Set the severity and risk levels to import (optional). Much of the host information from Tenable comes from Info-level plugins, so if you import only higher severities, assets runZero hasn't scanned may show little information.
4. Set the **Fingerprint only** toggle to *Yes* to use vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
5. Enter a name for the task, like `Nessus Professional sync` (optional).
6. Choose the Explorer that runs this connector task (optional).
7. Choose the site for your assets. runZero stores all newly discovered assets there.
8. Enter a description for the task (optional).
9. To leave out assets runZero hasn't scanned, switch the **Exclude unknown assets** toggle to *Yes*. The import includes them by default.
10. To include assets that haven't been assessed for vulnerabilities, switch the **Include unscanned assets** toggle to *Yes*.
11. Schedule the sync to run once or on a recurring schedule, starting on the date and time you set.
12. Activate the connection. The sync runs on your schedule, and the Scheduled tasks page shows when the next one will run.

Step 4b: Configure the Nessus Professional integration as a scan probe

Run the Nessus Professional integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *NESSUS* option to *Yes* in the *Probes and SNMP* tab and change any default options you need.
- Set the correct *Nessus* credential to *Yes* in the *Credentials* tab.
- Optionally, set the severity and risk levels for ingested vulnerability scan results.

Step 5: View Tenable assets and vulnerabilities

After a successful sync, your Tenable assets appear in your inventory with a Tenable icon in the **Source** column.

The integration also gathers the vulnerabilities Nessus Professional detected. Go to Inventory > Vulnerabilities to view the vulnerability data from Nessus Professional.

To filter by Tenable assets, run this query:

- View all Tenable assets:

  ```
  source:Tenable
  ```

Open an asset to see the attributes runZero gathered from the Tenable scan data.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the Nessus Professional integration unable to connect?

1. Check whether the Nessus Professional integration is returning any data.
   - Query the inventory rather than the task details to review all the data available from this integration.
   - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the Nessus Professional integration can't connect, check the task log for errors. Common errors include:
   - 500: server error, unable to connect to the endpoint
   - 404: hitting an unknown endpoint on the server
   - 403: not authorized, likely a credential issue
4. If Nessus Professional is on-premises, verify that the integration task runs from an Explorer with access to the Tenable host.

How do I solve the following error in Nessus Professional:

- `"error-message":"no tenable assets match import criteria","level":"error","msg":"could not load scan result data to writer"` {#nessuspro-import-error}

Enabling the *Include Unscanned Assets* toggle in the Tenable task configuration usually resolves this error, which Tenable returns intermittently. The toggle disables the filters runZero applies for live assets scanned in the last 30 days.

Tenable Nessus

runZero integrates with Tenable Nessus in two ways. For every version of Nessus, runZero can import Nessus files (`.nessus`) exported from your Nessus instance, and exports from Tenable Security Center work too. For Nessus Professional, the integration can also pull scan data from the Nessus Professional API.

Getting started with Tenable Nessus

To use the Tenable Nessus integration:

1. Export vulnerability scan results as Nessus files.
2. Import the Nessus files through the inventory pages.

Requirements

- Access to the Nessus portal.

Step 1: Export vulnerability scan results

1. Sign in to Nessus with the account you're using for the runZero integration.
2. Open the scan results you want to import into runZero.
3. Choose **Export** > **Nessus** to download the scan results.

Step 2: Import the Nessus files into runZero

1. Go to the Inventory page in runZero.
2. Choose **Import** > **Nessus scan (.nessus)** from the list of import types.
3. On the import data page:
    - Choose the site for your assets.
    - Set the severity levels and minimum risk level to ingest. Much of the host information from Tenable comes from Info-level plugins, so if you import only higher severities, assets runZero hasn't scanned may show little information.

Step 3: View Nessus assets and vulnerabilities

After a successful import, your Nessus assets appear in your inventory with a Tenable icon in the **Source** column.

The integration also gathers the vulnerabilities Nessus detected. Go to Inventory > Vulnerabilities to view the vulnerability data from Nessus.

To filter by Nessus assets, run this query:

- View all Nessus assets:

  ```
  source:Tenable
  ```

Open an asset to see the attributes runZero gathered from the Nessus scan file.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the Tenable Nessus integration unable to connect?

1.

1. Check whether the Tenable Nessus integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.

Tenable Security Center

runZero imports data from Tenable Security Center (previously Tenable.sc) through the Tenable Security Center API.

Getting started with Tenable Security Center

To set up the Tenable Security Center integration:

1. Create an API key for a user who can view and query vulnerabilities in Tenable Security Center.
2. Add the Tenable Security Center credential to runZero.
3. Choose whether to run the integration as a scan probe or a connector task.
4. Activate the integration to pull your data into runZero.

Requirements

- Administrator access to the Tenable Security Center portal.

Step 1: Create an API key

1. Sign in to Tenable Security Center with an Administrator account.
2. Make sure API key authentication is enabled.
3. Go to **Users** > **Users**.
4. Check the box for the user you want to create an API key for. The key gets the same access as that user, so pick one who can view and query vulnerabilities in the organization you want.
5. At the top of the table, click **API Keys** > **Generate API Key**.
6. Click **Generate**, then download or copy the API token.

Step 2: Add the Tenable Security Center credential to runZero

1. Go to the Credentials page in runZero and name the credential, for example `Tenable Security Center`.
2. Choose **Tenable Security Center Access & Secret** from the list of credential types.
3. Generate your Tenable Security Center API key as described in Step 1, then fill in the fields:
    - Enter your 64-character Tenable Security Center access key in **Access key**.
    - Enter your 64-character Tenable Security Center secret key in **Secret key**.

4. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
5. Save the credential.

Step 3: Choose how to configure the Tenable Security Center integration

You can run the Tenable Security Center integration as a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from one of your Explorers, and performs only the integration sync. For an internal Tenable Security Center instance, we recommend a connector that runs from one of your Explorers. For an external-facing instance, the connector can run from the cloud. If you self-host runZero, run the connector from an Explorer or from your runZero host, whichever can reach your Tenable Security Center install.

Step 4: Set up and activate the integration to sync data

Step 4a: Configure the Tenable Security Center integration as a connector task

A connection needs a schedule, which sets when the sync runs.

1. Activate a connection to Tenable Security Center. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they aren't listed, check that they have access to your current organization.
3. Set the *Tenable Security Center query mode* (optional):
    - Select *Define filters* to filter by vulnerability severity and risk level. Much of the host information from Tenable comes from Info-level plugins, so if you import only higher severities, assets may show little information.
    - Select *Use existing query ID* to specify the Tenable Security Center query to use. The query must be the *Vulnerability* type and use the *Vulnerability Detail List* tool.
4. Set the **Fingerprint only** toggle to *Yes* to use vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
5. Enter a name for the task, like `Tenable Security Center sync` (optional).
6. Choose the Explorer that runs this connector task (optional).
7. Choose the site for the connector.
8. Enter a description for the task (optional).
9. Schedule the sync to run once or on a recurring schedule, starting on the date and time you set.
10. Activate the connection. The sync runs on your schedule, and the Scheduled tasks page shows when the next one will run.

Step 4b: Configure the Tenable Security Center integration as a scan probe

Run the Tenable Security Center integration as a scan probe to have the runZero Explorer pull your vulnerability data into the runZero Console.

In a new or existing scan configuration:

- Set the *TENABLESECURITYCENTER* option to *Yes* in the *Probes and SNMP* tab and change any default options you need.
- Optionally, set the severity and risk levels for ingested vulnerability scan results or provide a query ID.
- Set the correct `TenableSecurityCenter` credential to *Yes* in the *Credentials* tab.

Step 5: View Tenable Security Center assets and vulnerabilities

After a successful sync, your Tenable Security Center assets appear in your inventory with a Tenable icon in the **Source** column.

The integration also gathers the vulnerabilities Tenable Security Center detected. Go to **Inventory** > **Vulnerabilities** to view the vulnerability data from Tenable Security Center.

To filter by Tenable Security Center assets or vulnerabilities, run these queries:

- View all Tenable Security Center assets:

  ```
  source:tenablesecuritycenter
  ```

- View all Tenable Security Center vulnerabilities:

  ```
  source:tenablesecuritycenter
  ```

Open an asset or vulnerability to see the attributes runZero gathered from the Tenable Security Center API.

VMware

runZero Platform syncs your VMware vCenter and ESXi virtual machine inventories.

Setting up VMware credentials

Unlike other API integrations, VMware synchronization runs as part of your regular runZero Explorer scans. Start by creating a set of VMware credentials.

On the Scanning with credentials page, click *Add Credential*, choose the *VMware vCenter/ESXi Username and Password* credential type, and enter the username and password. In most cases the username takes the form `user@domain.com`. The VMware account needs at least **read-only** access.

The CIDR allow list field limits which addresses receive the credentials, so the runZero Explorer doesn't hand them to unexpected VMware systems it encounters on the network.

If runZero has already found VMware API endpoints, the optional *VMware thumbprints* field lists their IP addresses and TLS fingerprints. Remove any system you don't want to trust with your VMware credentials. If you'd rather not limit authentication to a fixed list of IP addresses

and TLS certificates, set *VMware insecure* to *Yes* to allow authentication with untrusted endpoints.

Set organization access for the credentials as you would for any other stored credential; see Scanning with credentials.

Performing VMware synchronization

With the credentials defined, enable VMware synchronization in a scan task. Any task that scans the VMware host systems can synchronize VMware VM data.

The Probes tab of the scan setup has a section for enabling and disabling the VMware probe. Synchronization needs the probe enabled, which it is by default.

On the Credentials tab, use the toggle to enable the right set of VMware vCenter/ESXi credentials.

When the scan runs, the Explorer authenticates with every VMware ESXi or vCenter host it finds that the credentials are configured to trust. runZero imports data about the VMware VMs automatically and merges it with what scanning found.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the VMware integration unable to connect?

1. Check whether the VMware integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
3. If the VMware integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue
4. If VMware is on-premises, verify that the integration task runs from an Explorer with access to the VMware host.

Wiz

runZero imports data from the Wiz API to sync your cloud assets, software, and vulnerabilities from Wiz, for better visibility of your cloud assets and security posture.

Getting started with Wiz

To set up the Wiz integration:

1. Create a Service Account in Wiz with permissions to read graph resources, read reports, and create reports.
2. Add the Wiz credential to runZero.
3. Choose whether to run the integration as a scan probe or a connector task.
4. Activate the integration to pull your data into runZero.

Requirements

- Administrator access to the Wiz portal.

Step 1: Create a Service Account in Wiz

1. Sign in to Wiz with an Administrator account.
2. Go to **Settings** > **Access Management** > **Service Accounts** > **Add Service Account**.
3. Enter a descriptive name in the **Name** field.
4. For **Type**, select *Custom Integration (GraphQL API)*.
5. Under **Projects**, select the projects the Service Account can access, or leave it blank to allow all projects.
6. For **API Scopes**, enable `read:resources`, `read:reports`, and `create:reports`.
7. Click **Add Service Account** and copy the **Client ID** and **Client Secret**.
8. Go to **User Settings** > **Tenant** and copy the **API Endpoint URL**, which has the format `https://api.{{region}}.app.wiz.io/.`

Step 2: Add the Wiz credential to runZero

1. Go to the Credentials page in runZero and name the credential, for example `Wiz`.
2. Choose **Wiz Client Secret** from the list of credential types.
3. Using the service account you created on the settings page in the Wiz portal, fill in the fields:
    - In **Wiz Client ID**, enter the client ID of your Wiz service account.
    - In **Wiz Client Secret**, enter the client secret of your Wiz service account.
    - In **Wiz Auth URL**, enter the URL used to authenticate the Wiz service account.
    - In **Wiz API URL**, enter the API Endpoint URL for the Wiz API.
4. To share this credential with other organizations, select *Make this a global credential*. Otherwise, you can grant access per organization.
5. Save the credential.

Step 3: Choose how to configure the Wiz integration

You can run the Wiz integration as a scan probe or a connector task. A scan probe gathers integration data during a scan task. A connector task runs on its own, from the cloud or from

one of your Explorers, and performs only the integration sync.

Step 4: Set up and activate the integration to sync data

Step 4a: Configure the Wiz integration as a connector task

A connection needs a schedule and a site. The schedule sets when the sync runs, and the site is where runZero creates any new Wiz-only assets.

1. Activate a connection to Wiz. You can also reach every third-party connection from the integrations page, your inventory, or the tasks page.
2. Choose the credentials you added earlier. If they aren't listed, check that they have access to your current organization.
3. Set the severity and risk levels to import (optional).
4. Set the **Fingerprint only** toggle to *Yes* to use vulnerability records for fingerprint analysis without storing them in your runZero vulnerability inventory (optional).
5. Enter a name for the task, like `Wiz Sync` (optional).
6. Choose the Explorer that runs this connector task (optional).
7. Choose the site for your assets. runZero stores all newly discovered assets there.
8. Enter a description for the task (optional).
9. To leave out assets runZero hasn't scanned, switch the **Exclude unknown assets** toggle to *Yes*. The import includes them by default.
10. To leave out assets that haven't been assessed for vulnerabilities, switch the **Include unscanned assets** toggle to *No*.
11. Schedule the sync to run once or on a recurring schedule, starting on the date and time you set.
12. Activate the connection. The sync runs on your schedule, and the Scheduled tasks page shows when the next one will run.

Step 4b: Configure the Wiz integration as a scan probe

Run the Wiz integration as a scan probe to have the runZero Explorer pull your Wiz assets into the runZero Console.

In a new or existing scan configuration:

- Set the *WIZ* option to *Yes* in the *Probes and SNMP* tab and change any default options you need.
- Optionally, set the severity and risk levels for ingested vulnerability results.
- Set the correct *Wiz* credential to *Yes* in the *Credentials* tab.

Step 5: View Wiz assets, software, and vulnerabilities

After a successful sync, your Wiz assets appear in your inventory with a Wiz icon in the **Source** column.

The integration also gathers the software and vulnerabilities Wiz detected. Go to **Inventory** > **Software** or **Inventory** > **Vulnerabilities** to view them.

To filter by Wiz assets, run this query:

- View all Wiz assets:

```
source:Wiz
```

Open an asset to see the attributes runZero gathered from Wiz.

Troubleshooting

If you're having trouble with the integration, start with these questions and answers.

Why is the Wiz integration unable to connect?

1. Check whether the Wiz integration is returning any data.
    - Query the inventory rather than the task details to review all the data available from this integration.
    - In some cases, an integration's configuration limits how much data reaches the runZero console.
2. Some integrations require specific actions that are easy to overlook. If you miss a step during setup, the integration may not work correctly. Review this page and follow the steps exactly.
    - Double-check the API Scopes on the Wiz service account. A valid account that lacks the required permissions fails the import.
3. If the Wiz integration can't connect, check the task log for errors. Common errors include:
    - 500: server error, unable to connect to the endpoint
    - 404: hitting an unknown endpoint on the server
    - 403: not authorized, likely a credential issue

Outbound integrations

Using runZero data to enrich other tools

Beyond pulling in data from your other IT and security tools, the runZero platform has egress integrations with several platforms. Through product APIs and export/import, runZero can add asset context to those tools. If your tool isn't listed, a custom integration script can export runZero data to another system.

These integrations send runZero data to other platforms:

IT service management

- Atlassian Insight & Jira Service Management
- Jira (issue tracking)
- ServiceNow CMDB

Detection and investigation

- Panther
- Splunk Search
- Sumo Logic
- Tines
- Thinkst Canary

Vulnerabilities and risk

- SecurityGate.io

Atlassian Insight & Jira Service Management

> This page covers the asset-enrichment integration that imports runZero asset data into Jira Service Management's CMDB. To push runZero issues to Jira as tickets and sync changes back, use the Jira integration instead.

All runZero editions integrate with Jira Service Management through an import in Atlassian Insight, which loads runZero asset data into the CMDB.

Step 1: Export runZero asset data

Export data with the **Export** button in the runZero inventory, or with the Export API.

These sample Export API commands include the common export fields and omit the tags field. Replace the token `ETxxx...` with your account's export token from the Inventory export API

page.

For a CSV export, use this command (one line):

```
curl -o "assets.csv" -H "Authorization: Bearer ETxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  "https://console.runzero.com:443/api/v1.0/export/org/assets.csv"
```

For a JSON export, use this command (one line):

```
curl -o "assets.json" -H "Authorization: Bearer ETxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  "https://console.runzero.com:443/api/v1.0/export/org/assets.csv"
```

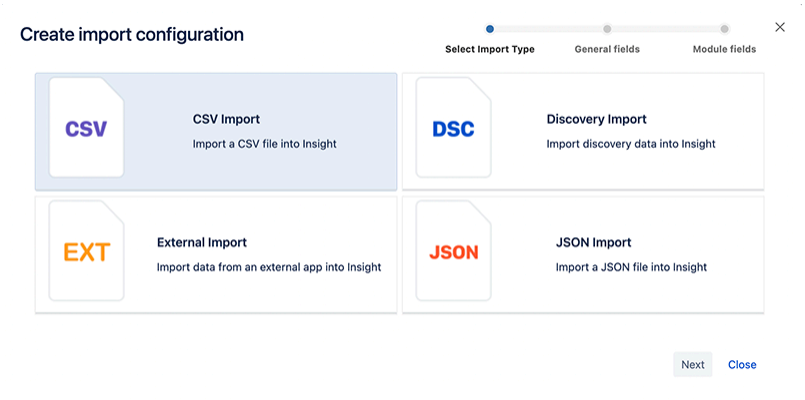
⚠

Atlassian Insight rejects any attribute longer than 255 characters and reports errors when an imported file contains longer fields. If you see those errors, you may need to trim or omit the fields.

Step 2: Import data into Atlassian Insight

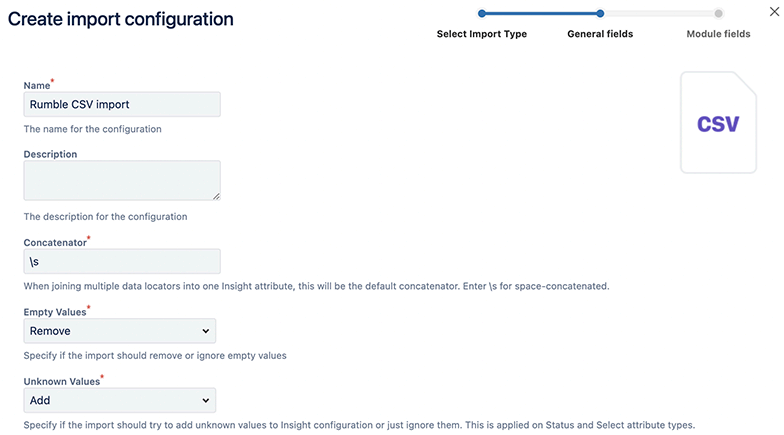
These steps need Jira Service Management Premium or Enterprise with Atlassian Insight installed.

1. In Jira Service Management, go to the Insight menu and click the **+** sign to create an object schema.
2. Name the object schema. This example calls it `Assets`.
3. Click **...** next to the object schema and choose **Configuration**.
4. Go to the **Import** tab.
5. Click **Create Configuration**.



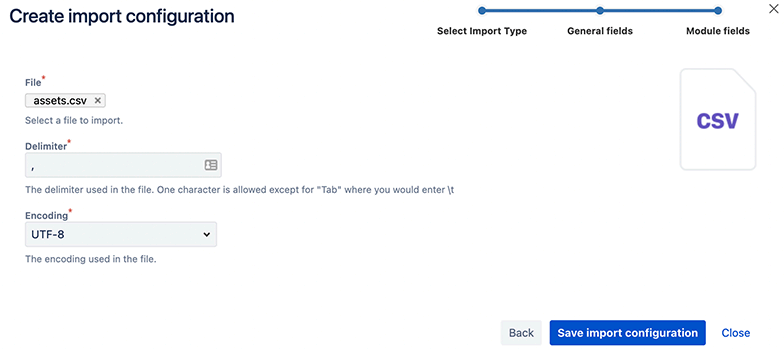
6. Choose CSV or JSON to match your export, and click **Next**.
7. Enter a name for the import.

8. In the **Concatenator** field, enter `\s`.



9. Scroll down and click **Next**.

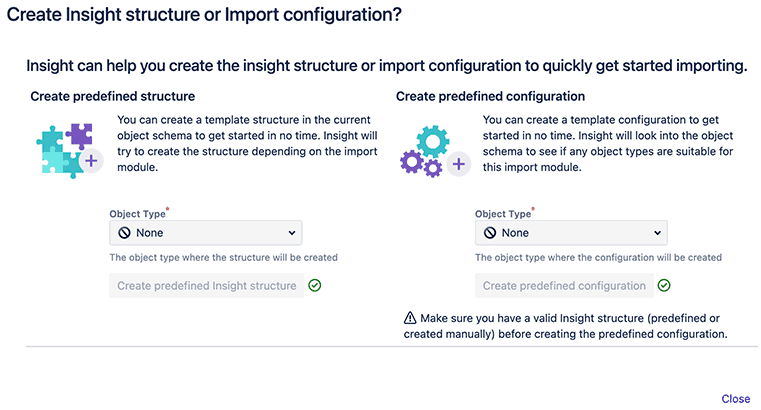
10. Choose the file you exported from runZero.



11. Click **Save Import Configuration**.

12. Click **Create predefined Insight structure** and wait for the task to finish.

13. Click **Create predefined configuration** and wait for the task to finish.



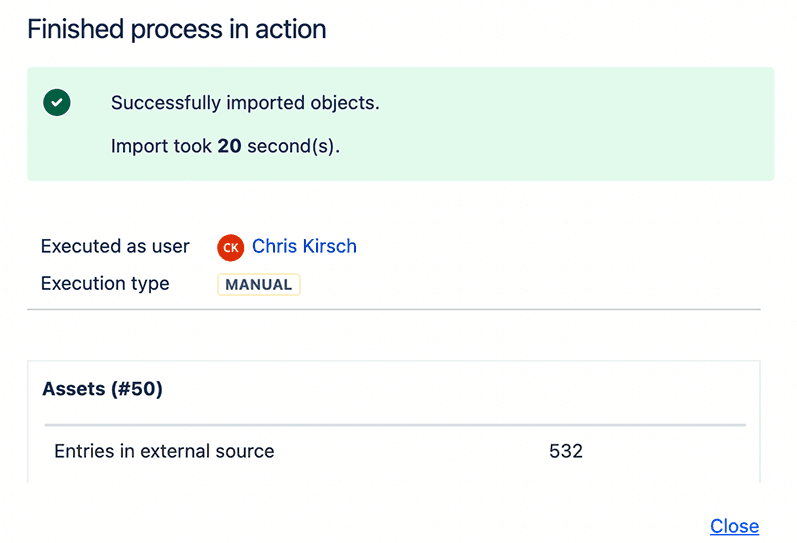
14. Close the window. The page now lists your configured import. Click the settings cog and select **Execute Import**.



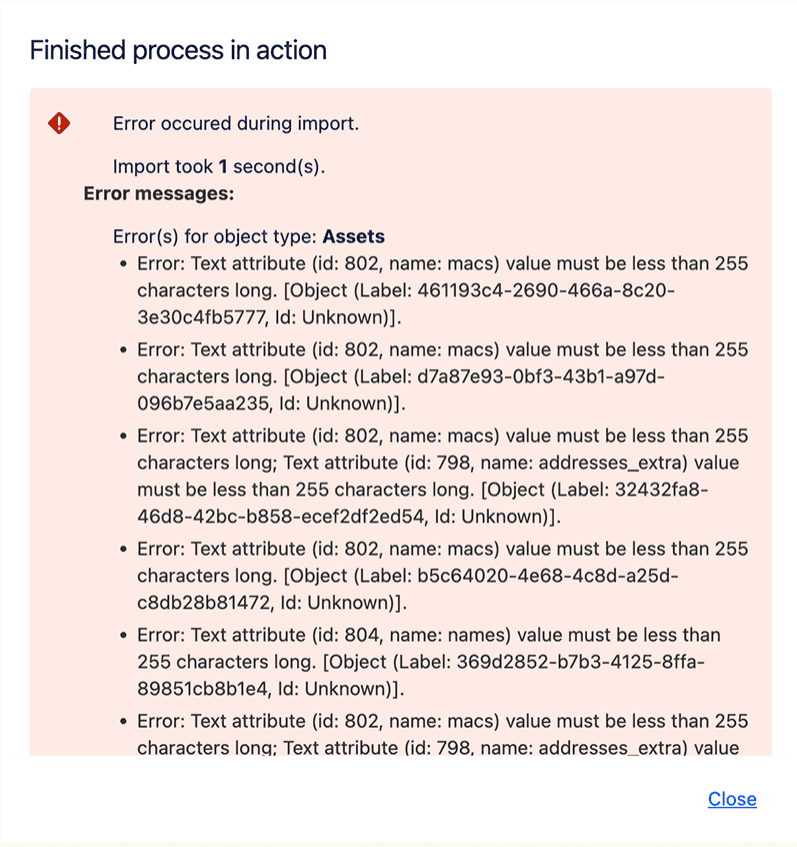
15. Click **Import**.



16. When the import finishes successfully, close the window.

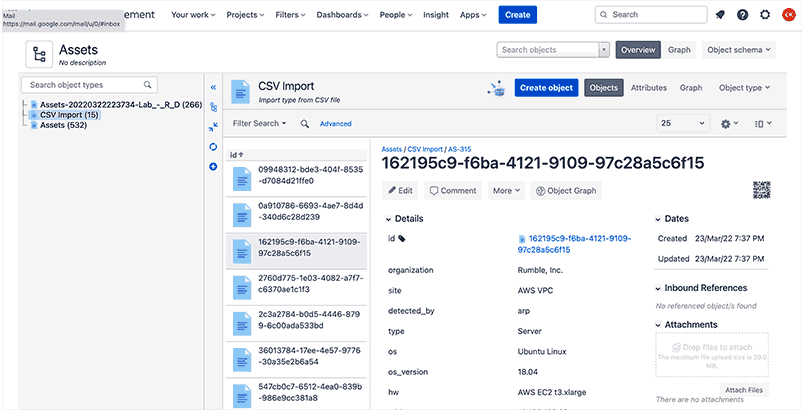


If you see the following error, one of your fields is longer than 255 characters. Trim the field to 255 characters or delete it so the import can succeed.



Step 3: View your imported assets

1. In Jira Service Management, click Insights.
2. Click your object schema.
3. Click an asset to view its details.



Jira

The Jira integration connects runZero to a Jira Cloud project for issue-tracking and service-desk operations. Once it is configured, you can push a runZero issue to Jira as a ticket and keep the runZero issue timeline in sync with changes made in Jira.

> To import runZero asset data into Jira Service Management's CMDB, see Atlassian Insight & Jira Service Management instead. That asset-enrichment integration is independent of the issue-tracking sync described here.

How the integration works

- runZero authenticates to Jira Cloud as a regular Atlassian user with HTTP basic auth (email + API token).
- When you create a Jira ticket from a runZero issue, runZero calls the Jira REST API (v3) to create the issue in the project configured on the credential, copies the relevant fields across, and stores the resulting Jira key and link on the runZero issue.
- A configurable Jira webhook posts changelog events back to runZero, and each change becomes a timeline entry on the matching runZero issue. If you configure state mapping, status and due-date changes also update the runZero issue directly.
- When you change a runZero issue state and a state mapping is configured, runZero calls the Jira Transitions API to move the linked ticket to the corresponding Jira status.
- When a linked runZero issue has an owner that runZero can match to an external Jira user, runZero syncs that owner with the Jira assignee. Changes to the Jira assignee sync back to the runZero issue.
- runZero initiates all traffic to Jira, and all traffic from Jira is signed and verified with HMAC-SHA256. runZero reads and writes no other Jira data in your environment.

Prerequisites

Before you start, you need:

- An Atlassian Cloud account that can create issues in the target Jira project.
- A Jira project to write tickets to, and that project's **Project key** (for example `SCRUM`, `SEC`, or `OPS`).
- A Jira API token. Create one at https://id.atlassian.com/manage-profile/security/api-tokens and copy it somewhere safe. You cannot view it again after you close the dialog.
- A randomly generated **webhook secret** if you plan to enable two-way sync. Any reasonably long random string will do; runZero uses it to verify inbound webhook requests.
- The Jira integration entitlement on your runZero account.

Creating the Jira credential

The Jira integration uses a runZero credential of type **Jira API Token**. You can create one ahead of time from the Credentials page, or directly from the **Create Jira ticket** dialog the first time you push an issue.

To create the credential ahead of time:

1. Go to Credentials in the runZero Console.
2. Click **New credential** and choose **Jira API Token**.
3. Fill in the form:
    - **URL**: the base URL of your Jira Cloud site, for example `https://yourcompany.atlassian.net`. Do not include a trailing path.
    - **Email**: the email address of the Atlassian user the API token belongs to. This account must be able to create and transition issues in the target project.
    - **API token**: the API token you created on the Atlassian side.
    - **Webhook secret**: a random string that authenticates inbound Jira webhooks. Save the value somewhere you can paste it into Jira later.
    - **Jira Project Key**: the key of the Jira project where runZero creates tickets (for example `SEC` or `OPS`). This field is required; ticket creation fails if you leave it empty.
    - **Map Issue States to Jira Values**: optional. A table mapping each runZero issue state to the corresponding Jira status name. See State mapping and bidirectional sync for details.
4. Save the credential. runZero validates the URL, email, and token with an authenticated call to Jira and rejects the credential if the call fails. The validation result appears next to the credential.

After you save the credential, open it for editing and copy the read-only **Jira webhook URL** field. You need that URL when configuring the Jira webhook.

You can reuse one credential for as many Jira tickets as you like. If you change the API token in Atlassian, update the credential in runZero or future calls will fail.

Creating a Jira ticket from an issue

1. Open an issue from the Issues page.
2. In the sidebar:

- **Credential**: select the Jira credential to use.
- **Attach snapshot to ticket**: optional. When enabled, runZero uploads a CSV of the issue's instance snapshot as an attachment on the new Jira ticket.
- Click **Create ticket**.

runZero then:

- Creates the Jira issue with the project key from the credential, the summary, the description, and the runZero issue's due date if one is set.
- Uploads the snapshot CSV as a Jira attachment, if you enabled it. A failed CSV upload does not abort ticket creation; the ticket is still linked.
- Saves the Jira issue key (for example `SEC-42`), a browse link (`{your-jira-url}/browse/SEC-42`), the credential, and the Jira base URL on the runZero issue.
- Appends a `Created Jira ticket SEC-42 for issue "..."` entry to the runZero issue timeline.

Once linked, the issue sidebar shows the Jira key and a button that opens the ticket in Jira.

A runZero issue links to at most one Jira ticket at a time. To relink, unlink the existing ticket first.

Unlinking a ticket

From a runZero issue with a linked ticket, use the unlink action in the sidebar to clear the link. Unlinking changes only the runZero side: the Jira ticket is neither deleted nor modified. After unlinking, you can create a new ticket from the same issue.

State mapping and bidirectional sync

The **Map Issue States to Jira Values** table on the Jira credential turns on two-way sync of state and due date between runZero issues and Jira tickets.

To configure state mapping, open the credential and add one row per runZero state you want to sync:

- **Issue State** (key): one of the four runZero issue states, `open`, `in progress`, `closed`, or `resolved`.
- **Jira Value** (value): the exact name of the corresponding Jira status in your project workflow, for example `To Do`, `In Progress`, or `Done`. Matching is case-insensitive for inbound webhook events.

You can map all four states or only some of them. runZero ignores unmapped states in both directions.

Outbound sync (runZero → Jira)

When you change a runZero issue state, or when runZero automatically transitions an issue to `resolved`, runZero looks up the new state in the credential's state mapping. If that state maps to a Jira status name, runZero calls the Jira Transitions API to move the linked ticket to the target status.

If the Jira workflow has no available transition from the ticket's current status to the target status, the transition silently fails. runZero applies the state change on its side whether or not the Jira transition succeeds.

Inbound sync (Jira → runZero)

When Jira sends a webhook with a status change, runZero reverses the state mapping to find the matching runZero state and updates the issue. Due date changes on the Jira ticket also flow back to the runZero issue.

Issue owner synchronization

runZero syncs issue owners with Jira assignees for linked tickets. When you push an issue to Jira, runZero looks for an external Jira user that matches the runZero owner. For the runZero issue owner, you can enter either the email address or the username of the Jira user. If runZero finds a matching Jira user, it assigns that user to the Jira ticket.

When the Jira assignee changes, the Jira webhook updates the owner of the linked runZero issue. For Jira sync, runZero accepts an owner only if it matches an external Jira user.

Issue ownership uses the built-in `Issue owner` type and supports one owner per issue. runZero does not sync custom ownership types. See Managing Issue ownership and Issue ownership.

Note: The casing of the Jira status needs to match the casing set in the Jira workflow

Example mapping:

runZero state	Jira status
`Open`	`To Do`
`In progress`	`In Progress`
`Closed`	`Closed`
`Resolved`	`Done`

Configuring the Jira webhook

To turn on inbound sync from Jira to runZero, configure a Jira webhook for the project (or the whole site):

1. In Jira, go to **System → WebHooks** and click **Create a WebHook** (administrator access required).
2. **Name**: anything descriptive, for example `runZero issue sync`.
3. **URL**: open your runZero Jira credential and copy the read-only **Jira webhook URL** field. It follows the pattern `https://<your-runzero-console>/api/webhooks/jira/${issue.key}`. The literal text `${issue.key}` is a Jira URL variable; Jira substitutes the actual issue key for each event.
4. **Secret**: paste the same webhook secret you saved on the runZero credential. Jira signs each request with HMAC-SHA256 and sends the signature in the `X-Hub-Signature` header.
5. **Events**: enable at least the **issue updated** event. You can add more events; runZero records every changelog item it receives.
6. (Optional) **JQL filter**: restrict the webhook to the project used with this credential, for example `project = SEC`. This keeps out unneeded events from other projects.
7. Save the webhook.

When an event fires, runZero:

1. Looks up the matching runZero issue by Jira issue key + Jira base URL (so multiple Jira sites do not collide).
2. Validates the HMAC-SHA256 signature against the credential's webhook secret using a constant-time comparison.
3. If state mapping is configured, applies any status or due-date changes to the runZero issue.
4. Appends one timeline line per changelog item, in this form:

```
Jira update (SEC-42): status changed from "In Progress" to "Done"; assignee set to "alice@example.com"
```

If a webhook arrives without a valid signature, for an unknown Jira key, or for a deleted runZero issue, runZero silently returns `200 OK` without writing anything, so the endpoint does not leak which keys are linked.

What gets sent to Jira

When runZero creates a ticket, it sends:

- The runZero issue **name** as the Jira **summary**.
- The runZero issue **description** as the Jira **description**.
- The runZero **due date**, if one is set, as the Jira **due date**.
- The **project key** from the credential, which routes the ticket to the right project.
- Optionally, a CSV file with one row per (vulnerability, asset) instance from the issue snapshot, uploaded as a Jira attachment.

After ticket creation, ongoing sync depends on your state mapping configuration:

- **runZero → Jira**: when a runZero issue state changes and a mapping is configured, runZero pushes the transition to Jira through the Transitions API.
- **Jira → runZero**: status and due date changes arrive through the Jira webhook and update the runZero issue (status changes require state mapping).

After the initial ticket creation, runZero does not push comments, description edits, or other field changes to Jira.

Troubleshooting

- **Credential verification fails when saving.** Check that the URL has no trailing path, the email matches the Atlassian account that owns the API token, and the token has not been revoked. The credential page shows the specific HTTP status returned by Jira.
- **Create Jira ticket returns an error.** The runZero issue must have a non-empty name and description. The selected credential must have a **Jira Project Key** configured; if the field is empty, the request fails with a *project_key is required* error. The project key must exist in Jira, and the API user must be allowed to create issues in it. The error message echoes the HTTP status returned by Jira.
- **Snapshot CSV is missing from the Jira ticket.** Check that the API user has permission to add attachments to the project. Snapshot upload is best-effort; the ticket is still created if the upload fails.
- **Jira changes do not appear in the runZero timeline.**
  - Confirm the webhook URL contains the literal `${issue.key}` variable.
  - Confirm the webhook secret in Jira matches the value stored on the runZero credential.
  - Confirm Jira is sending the `X-Hub-Signature` header (some proxies strip it).
  - Confirm the runZero issue's stored Jira base URL matches the Jira site sending the webhook. If you migrate Jira sites, unlink and recreate the ticket.
- **State changes are not being pushed to Jira.** Confirm the credential has a state mapping entry for the runZero state that changed, and that the Jira status name matches the exact name in your Jira workflow. The API user must also have permission to transition issues in the project. If no available Jira workflow transition leads from the ticket's current status to the target status, the push silently fails.
- **Jira status changes are not updating the runZero issue state.** Confirm the Jira status name in the webhook event matches the value in the credential's state mapping (matching is case-insensitive). Confirm the webhook secret is correct and that the `X-Hub-Signature` header is reaching runZero.

Panther

You can import runZero data into your Panther instance and use it in Panther logging and alerting.

Requirements

- A Panther account with the required permissions
- An AWS S3 bucket
- Exported .jsonl files from runZero, uploaded to your AWS S3 bucket

Step 1: Adding a custom schema

1. Go to **Configure > Schemas** and select **Create New**.

2. Add a name.
3. Upload a sample log, and Panther parses the runZero output schema from it.

Step 2: Adding a custom log source

1. Go to **Configure > Log Sources** and select **Create New**.
2. Complete the **Basic Information** section.
3. Choose to configure S3 prefixes and schemas now, and select the custom schema you created.
4. Configure the IAM role:
    - Choose **Using the AWS Console UI**.
    - Click **Launch Console UI**.
    - Review the stack in AWS, check the box to approve it, and deploy the stack.
    - When the deployment completes, open the **Resources** tab and select the new **LogProcessingRole**.
    - Copy the ARN from that role into the field on the Panther console.
5. Optionally, configure an alarm that fires if logs are not processed.

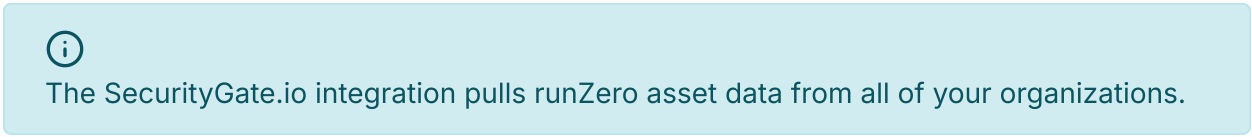
From then on, Panther automatically ingests and processes any .jsonl files added to that AWS S3 bucket.

SecurityGate.io

All runZero editions integrate with SecurityGate.io to bring runZero asset visibility into your risk assessment program. Setup takes a few steps in your SecurityGate.io console.

Requirements

- A runZero API key to configure the SecurityGate.io integration.



Integrate runZero with SecurityGate.io

1. Sign in to your SecurityGate.io console.
2. Go to **My Account** under the **Hello** dropdown menu.
3. Click **Integration Manager**.
4. Select **Add New Integration**.
5. Choose *Rumble* from the **Integration Partner** dropdown menu.
6. Provide the **Account API Key**, **Server URL**, and **API Version**.
7. Click **Test Connection**. If the test succeeds, click **OK** to save the configuration.

Viewing runZero data in SecurityGate.io

After you enable the integration in SecurityGate.io, runZero data is available on the Asset Inventory page.

ServiceNow Service Graph

The Service Graph connector for runZero brings runZero assets into your ServiceNow CMDB as CIs and, if you choose, periodically updates those CIs with fresh information from runZero scans.

The Service Graph Connector fetches and transforms data with ServiceNow IntegrationHub ETL and passes it through the Identification and Reconciliation Engine (IRE), so you can fine-tune specific fields and CI class mappings from the ServiceNow console. You can also set a runZero search query that decides which assets the connector brings in.

Important notes

- runZero does not charge for the connector. However, the CI resources the connector creates in CMDB increase your ServiceNow Subscription Unit consumption. CIs created by Service Graph Connectors should be charged at a lower rate than CIs created other ways; consult ServiceNow to confirm this and to get current pricing.
- If you self-host runZero, you need a MID Server so ServiceNow can connect to your runZero console. If your console uses a self-signed TLS certificate, or one signed by your own internal CA, configure the MID server to trust that CA or certificate. For details, see ServiceNow Knowledge Base article KB0863673.
- The Service Graph Connector is packaged and available from the ServiceNow Application Store, but fitting a connector into your existing ServiceNow configuration is unlikely to be a one-click operation. We strongly advise having a ServiceNow Consulting and Implementation partner available to help.
- For detailed installation instructions with screenshots, see the Scoped Application Installation and Configuration Guide on the Service Graph Connector for runZero page under Supporting Links and Docs.

Prerequisites

To use the Service Graph connector for runZero, you need:

- A Platform license for runZero.
- A ServiceNow ITOM license including ITOM Discovery and ITOM Visibility.
- Integration Commons for CMDB installed.

Connector installation checks the ServiceNow dependencies for you.

Installation

Start by finding the Service Graph connector for runZero in the ServiceNow Application Store and installing it into your ServiceNow instance through the usual process.

The connector adds a **Service Graph connector for runZero** entry to the ServiceNow main menu, with entries for **Setup**, **Data Sources**, **Scheduled Imports**, **Support**, and **System Import Sets**. You need the **admin** role in ServiceNow to configure the connector.

Setup

The **Setup** menu has a **Guided Setup** process that walks you through the connector in three stages:

1. Configure the connection
2. Configure the mappings (optional)
3. Configure the scheduled import (optional)

Step 1: Configure the connection

Configuring the connection has two parts: entering a runZero API key as credentials, and setting the connection options.

Entering the API key

The connector accepts a runZero **Organization API** or **Export API** key. To get either one, open the **Organizations** page in runZero and click the organization whose assets you want in ServiceNow CMDB. Organization tokens begin with `OT`, and export tokens begin with `ET`.

In the **API Key Credentials** form, paste the runZero token into the **API Key** field and click **Update**.

Configuring connection options

If you use runZero cloud, the default **HTTP connection** options should be correct.

You can add a **Search** string to the connection's **Attributes** at the bottom of the form. The connector passes this search to runZero to pick the set of assets to bring into CMDB. Write it in runZero search query format. We strongly encourage testing it in your runZero asset inventory first.

Leave the **Connection** alias and **Base path** fields as they are. (The connection alias groups the connection settings and the API key credentials for the connector to use.)

If you self-host runZero, the **connection options** form is where you pick the MID server and give the URL of your runZero console. Put your console's fully qualified hostname in the **Host** field. The hostname should match the hostname in the console's TLS certificate so the MID server trusts it, even when the MID server already trusts the certificate. Check the **Use MID server** box and select the MID server to use.

Step 2: Configure the mappings (optional)

In the second step, you can customize how runZero data maps to CI classes with the IntegrationHub ETL Transform Map Assistant.

Skip this stage if you have no special data mapping requirements.

Step 3: Configure the scheduled import (optional)

The third step sets a schedule for data import. Test the connector setup with a one-off import first. Once everything is configured correctly, you can create recurring imports.

Scheduled imports are regular ServiceNow tasks. Skip this step if you don't want a scheduled import yet.

Notes on data mapping

The ServiceNow CMDB and runZero object data models differ. Some runZero data fits no standard CMDB attribute, and in other cases runZero lacks data that CMDB expects. This list shows how the connector maps runZero attributes to ServiceNow CMDB attributes:

- **IP addresses**: ServiceNow CIs have an attribute that takes a single IP address. The connector places the first IP address in the CI attribute IP Address and creates the remaining addresses as `IP Address` CIs owned by the device CI. Each `IP Address` CI gets a `Description` that says whether runZero has scanned it. By default, ServiceNow does not show IP addresses owned by a device CI. It only shows IP addresses owned by a network adapter CI that the device CI owns. This is a known limitation of ServiceNow's default CMDB forms. runZero doesn't associate IP addresses with network adapters because it generally doesn't know the necessary information. In that situation, the CMDB connector development guide says to associate the IP addresses with the device CI.
- **Names**: ServiceNow CIs have a single `Name` attribute, while runZero assets can have any number of names, with runZero's best guesses at the front of the list. The connector places the first name in the CI `Name` attribute. The first name that looks like a fully qualified domain name goes in the `Fully qualified domain name` attribute, and the connector uses it to compute a DNS Domain for CIs descended from Network Gear. The full list of names goes in the `Description` attribute for reference.
- **Serial numbers**: ServiceNow CIs have an attribute that takes a single serial number. The connector puts the first serial number in the `Serial Number` attribute, then runs a Serial Number Lookup operation to add the complete list of serial numbers with their types.
- **Tags**: runZero assets can have any number of tags, either single tags such as `tag_name` or tags with values in the format `tag=value`. The connector places them in the `Key Value` class and associates that class with the `Hardware` class.
- **MAC addresses/Network Adapters**: As with IP addresses, ServiceNow has both a single-value `MAC Address` attribute and an option to look up named MAC addresses to create Network Adapter objects. A runZero asset can have any number of MAC addresses, but because runZero runs no agent software on the scanned systems, it cannot know the interface name that goes with a given MAC address. The connector imports the first MAC address as the `MAC Address` attribute and the entire set, through

lookup, as `Network Adapter` objects. It names each one from the MAC address and IP address (e.g. `00:00:00:00:00:00-127.0.0.1`), or from the MAC address and hostname when no IP address is available (e.g. `00:00:00:00:00:00-hostname`). If neither an IP address nor a hostname is available, the MAC address alone is the network adapter name. In some cases, runZero may not know an asset's MAC address(es); the connector then adds a zero MAC address (`00:00:00:00:00:00`) to satisfy ServiceNow requirements for some CI classes.

- **Sites**: runZero sites represent distinct networks, which may or may not correspond to physical sites. The connector imports them as CMDB `Network Site` objects.
- **Organizations**: Network hardware CIs have no obvious attribute for the runZero organization, so the connector adds the organization name to the `Description` of CIs.
- **SNMP data**: The runZero `snmp.sysDesc` attribute, if present, also goes in the `Description`.

Splunk Search

runZero integrates with Splunk through a dedicated Splunk Addon that is compatible with Splunk 7, Splunk 8, and Splunk Cloud. The add-on pulls new or updated hosts into a Splunk index, where you can analyze, visualize, and monitor them.

The add-on uses the Splunk API from the runZero Network Discovery platform. It syncs assets into Splunk with multiple inputs, global API key management, and an optional search filter for each input. For example, you can track new assets as one input and SMBv1-enabled assets as another.

To set up the add-on, you need an Export API or Organization API key, which you can generate from your Organization page in the runZero Console. For an overview of the runZero APIs, see the API documentation.

Get the runZero add-on for Splunk

1. Sign in to Splunk.
2. Go to **Find More Apps**.
3. Search for `runZero Network Discovery`.
4. Install the add-on for runZero.
5. Splunk prompts you to sign in again. After you log back in, the add-on is installed and you can open the runZero Asset Sync app. Splunk might also prompt you to restart your server.

Asset sync modes

The add-on has two asset sync modes: New Assets Only and All Updated Assets. You can export the assets newly discovered or updated since the last poll, in a sync-friendly format for Splunk. When you pull data into Splunk, you can use the same Asset Sync API capabilities, such as search filters, fields, and time-based checkpoints.

With data flowing into Splunk, you can create Splunk inputs with filters that sync only the assets with a certain protocol, discovery date, or open service.

Sumo Logic

All runZero editions integrate with Sumo Logic so you can see and visualize your asset data in Sumo Logic. Setup takes a few steps in your Sumo Logic console, and you can configure the integration for either of two purposes:

- Complete asset visibility
- Targeted alerting and visualization

Requirements

- A Sumo Logic account
- A runZero account and API key.

Sumo Logic asset export

runZero integrates with Sumo Logic so you can export your asset inventory into Sumo Logic and use it in the SIEM.

Integrating runZero with Sumo Logic

You can connect Sumo Logic and runZero in one of three ways, each with its own configuration steps.

Option A: Local script

1. Create a Sumo Logic HTTP source.
2. Configure your host to run the provided script.

Option B: AWS Lambda function

1. Create a Sumo Logic HTTP source.
2. Configure the AWS Lambda function to run the provided script.

Option C: Sumo Logic script source

1. Install a Sumo Logic collector.
2. Create a Sumo Logic script source.

Once data is flowing, you can start using it in Sumo Logic.

Option A: Local script

Step 1: Create a Sumo Logic HTTP source

1. Sign in to Sumo Logic and go to **Manage Data** > **Collection**.
2. Click **Add Collector**, then select **Hosted Collector**.
    - Provide a name, such as `runZero Collector`, and click **Save**.

3. If prompted to add a data source, click **OK**. Otherwise, find your Collector in the list and click **Add Source**.
4. Select the **HTTP Logs and Metrics** source.
    - Provide a name, such as `runZero Assets`, then click **Save**.
5. Copy the URL Sumo Logic provides; you need it in step 2.

Step 2: Configure your host to run the provided script

1. Pick the host that will run the script.

2. Check that the host has Python3 and Pipenv installed.

3. Save the script below on that host.

```
#!/usr/bin/env python3
import json
import requests
import os

# RUNZERO CONF
RUNZERO_EXPORT_TOKEN = os.environ["RUNZERO_EXPORT_TOKEN"]
HEADERS = {"Authorization": f"Bearer {RUNZERO_EXPORT_TOKEN}"}
BASE_URL = "https://console.runZero.com/api/v1.0"

# SUMO LOGIC CONF
HTTP_ENDPOINT = os.environ["SUMO_HTTP_ENDPOINT"]


def main():
    url = BASE_URL + "/export/org/assets.json"
    assets = requests.get(url, headers=HEADERS)
    batchsize = 500
    if len(assets.json()) > 0 and assets.status_code == 200:
        for i in range(0, len(assets.json()), batchsize):
            batch = assets.json()[i:i+batchsize]
            f = open("upload.txt", "w")
            f.truncate(0)
            for a in batch:
                json.dump(a, f)
                f.write("\n")
            f.close()
            r = open("upload.txt")
            requests.post(HTTP_ENDPOINT, data=r.read())
            r.close()
    else:
        print(f"No assets found - status code from runZero API: {assets.status_code}")

if __name__ == "__main__":
    main()
```

4.

Set your environment variables with these commands:

- `export RUNZERO_EXPORT_TOKEN=XXX`: your runZero export API token, from an organization detail page in your runZero console. Select the organization you want to export data from, then click **Edit organization** to view the export API token.
- `export SUMO_HTTP_ENDPOINT=XXX`: the Sumo Logic HTTP source URL you copied in step 1.

5. Create a virtual environment for the script by running `pipenv --python /path/to/python3`.
6. Install the `requests` library in the virtual environment for the API calls:
   - `pipenv shell`
   - `pip install requests`
7. Test the script by running it from the virtual environment.
   - Start with the location from the `pipenv` output.
   - Append `/bin/python3` to use Python in the virtual environment.
   - Use the full path to the script.

```
my-server:~/ $ /home/user/.local/share/virtualenvs/runZero-scripts-mVQtFLDO/bin/python3 \
  /home/user/scripts/script.py
```

8. Configure a crontab task to run at the cadence you want.
   - On the hour: `0 * * * * RUNZERO_EXPORT_TOKEN=XXX SUMO_HTTP_ENDPOINT=XXX /path/to/virtual/env/python3 /path/to/script.py`
   - Daily at midnight: `0 0 * * * RUNZERO_EXPORT_TOKEN=XXX SUMO_HTTP_ENDPOINT=XXX /path/to/virtual/env/python3 /path/to/script.py`
   - Weekly at midnight on Monday: `0 0 * * 1 RUNZERO_EXPORT_TOKEN=XXX SUMO_HTTP_ENDPOINT=XXX /path/to/virtual/env/python3 /path/to/script.py`

Option B: AWS Lambda function

Step 1: Create a Sumo Logic HTTP source

1. Sign in to Sumo Logic and go to **Manage Data** > **Collection**.
2. Click **Add Collector**, then select **Hosted Collector**.
   - Provide a name, such as `runZero Collector`, and click **Save**.
3. If prompted to add a data source, click **OK**. Otherwise, find your Collector in the list and click **Add Source**.
4. Select the **HTTP Logs and Metrics** source.
   - Provide a name, such as `runZero Assets`, then click **Save**.
5. Copy the URL Sumo Logic provides; you need it in step 2.

Step 2: Configure the AWS Lambda function to run the provided script

1. In your AWS Console, open the Lambda page.
2. Click **Create a function**.

3. Give your function a **name**.
4. Select *Python 3.9* as the **runtime**.
5. Leave everything else at its default. Click **Create function** to move to the next page.
6. Click **Add Trigger** to set up a cron job.
7. Select *EventBridge* to set up a schedule.
8. Use an existing rule or select **Create new rule**.
    - Give it a name and set **Rule type** to `Schedule expression`.
    - Use one of these expressions, or write your own for the cadence you want:
        - Daily: `rate(1 day)`
        - Every 12 hours: `rate(12 hours)`
        - Every 3 hours: `rate(3 hours)`
    - Click **Add** to return to the main Lambda configuration page.
9. Under **Configuration**, select *Environment variables*.
10. Enter these two environment variables:
    - `RUNZERO_EXPORT_TOKEN`: your export API token, from an organization detail page in your runZero console. Select the organization you want to export data from, then click **Edit organization** to view the export API token.
    - `SUMO_HTTP_ENDPOINT`: the URL you copied in step 1.
11. Click *Save* to return to the main Lambda configuration page.
12. Click the *Code* tab and replace the default code with this script.

```
import json
import urllib3
import os

# RUNZERO CONF
RUNZERO_EXPORT_TOKEN = os.environ["RUNZERO_EXPORT_TOKEN"]
HEADERS = {"Authorization": f"Bearer {RUNZERO_EXPORT_TOKEN}"}
BASE_URL = "https://console.runZero.com/api/v1.0"

# SUMO LOGIC CONF
HTTP_ENDPOINT = os.environ["SUMO_HTTP_ENDPOINT"]


def lambda_handler(event, context):
    http = urllib3.PoolManager()
    url = BASE_URL + "/export/org/assets.json"
    response = http.request("GET", url, headers=HEADERS)
    data = response.data
    assets = json.loads(data)
    batchsize = 500
```

```
for i in range(0, len(assets.json()), batchsize):
            batch = assets.json()[i : i + batchsize]
            f = open("upload.txt", "w")
            f.truncate(0)
            for a in batch:
                json.dump(a, f)
                f.write("\n")
            f.close()
            r = open("upload.txt")
            http.request("POST", HTTP_ENDPOINT, data=r.read())
            r.close()
    else:
        print(f"No assets found - status code from runZero API: {assets.status_code}")
```

13. Click **Deploy** to update the code.
14. Click **Test** to verify the code works.

The function now posts your asset data export to Sumo Logic at the cadence you configured.

Option C: Sumo Logic script source

Step 1: Install a Sumo Logic collector

Follow the Sumo Logic documentation to install a collector.

Step 2: Create a Sumo Logic script source

Sumo Logic also documents script sources. Once your collector is installed, set up the script source:

1. Go to the Collection page in Sumo Logic.
2. Find your collector and click **Add** > **Add Source**.
3. Select `Script` as the **source type**.
4. Enter a `Name` and `Source Category`.
5. Select a `Frequency`.
6. Select **Command** type `/usr/bin/python`.
7. Paste this script into the **Script** field.

```
#!/usr/bin/python
import json
import requests
import os

# RUNZERO CONF
RUNZERO_EXPORT_TOKEN = os.environ['RUNZERO_EXPORT_TOKEN']
HEADERS = {'Authorization': 'Bearer ' + RUNZERO_EXPORT_TOKEN}
```

```
BASE_URL = 'https://console.runZero.com/api/v1.0'


def main():
    url = BASE_URL + '/export/org/assets.json'
    assets = requests.get(url, headers=HEADERS)
    if assets.status_code == 200:
        for a in assets.json():
            print(json.dumps(a))
    else:
        print(f"No assets found - status code from runZero API: {assets.status_code}")


if __name__ == '__main__':
    main()
```

8. Click **Save** to start the source.

Working with the asset data in Sumo Logic

Once your asset data is in Sumo Logic, you can use it like any other log source. These sample searches are a starting point for scheduled searches and dashboards.

Search distinct assets

```
_sourceCategory="runzero"
| json field=_raw "id"
| count_distinct(id) as distinct_assets
```

Search assets with more than 3 services running

```
_sourceCategory="runzero"
| json field=_raw "addresses_extra"
| json field=_raw "addresses"
| json field=_raw "id"
| concat("https://console.runzero.com/inventory/", id) as runzero_link
| json field=_raw "service_count"
| where service_count > 3
| count addresses, addresses_extra, service_count,  runzero_link
```

Determine counts of different operating systems

```
_sourceCategory="runzero"
| json field=_raw "os"
| where !isEmpty(os)
| json field=_raw "id"
| count os, id
| count os
```

Sumo Logic alerting

runZero integrates with Sumo Logic so you can visualize your asset data and track your progress on reducing risk in your asset inventory over time.

Connecting Sumo Logic and runZero requires:

1. Creating a Sumo Logic HTTP source
2. Creating a runZero alert template
3. Creating a rule in runZero
4. Handling runZero data in Sumo Logic
5. Creating a Sumo Logic dashboard (optional)

Step 1: Create a Sumo Logic HTTP source

1. Sign in to Sumo Logic and go to **Manage Data** > **Collection**.
2. Click **Add Collector**, select **Hosted Collector**, provide a name such as `runZero Collector`, and click **save**.
3. If prompted to add a data source, click **OK**. Otherwise, find your Collector in the list and click **Add Source**.
4. Select the **HTTP Logs and Metrics** source, provide a name such as `runZero Alerts`, and click **save**.
5. Copy the URL Sumo Logic provides; you need it in step 2.

Step 2: Create a runZero alert template

1. Create an alert template in runZero with these details:
    - **Name**: a name for the template
    - **Template type**: JSON
    - **Subject line for message**: leave empty
    - **Body of message**: this JSON example includes the rule name and the search URL in the alert message body

      ```
      {"rule_name":"{{rule.name}}","search_url":"{{search.url}}","found": "{{search.found}}",
      "assets_new": "{{scan.assets_new}}"}
      ```

2. Create an alert channel in runZero with these details:
    - **Name**: a name for the alert channel
    - **Channel type**: Webhook
    - **Webhook URL**: the webhook URL you copied from Sumo Logic

Step 3: Create a rule in runZero

With the alert template and channel created, pick the triggers to alert on. Common examples:

- Asset query results: a query in runZero matches after a scan completes
- New assets found: a scan completes with new assets
- Agent offline: your runZero Explorer stops checking in to the console
- Task failed: a task fails for any reason

The rest of the steps use **asset query results** as the example. Review example queries for ideas, and see Using the rules engine for more on rules and alerts.

1. Create a new alert rule.
2. Select `asset-query-results` and click **Configure rule**.
3. Enter values for the rule:
    - **Name**: the name of the rule.
    - **Conditions**: optional parameters; the alert triggers when all conditions match.
        - **Query**: the query the assets must match.
        - **Number of matches**: the numeric comparison to apply to the **value**.
        - **Value**: the match threshold that triggers the rule.
        - **Limit to organization**: limits the alert to a specific organization.
        - **Limit to site**: limits the alert to a specific site.
        - **Action**: `Notify`
            - **Notification channel**: the alert channel you created in step 2.
            - **Notification template**: the alert template you created in step 2.

Step 4: Handle runZero data in Sumo Logic

Display the raw runZero data

```
_source="runZero Alerts" and _collector="runZero"
```

Show alerts matching the runZero rule name

```
_source="runZero Alerts" and _collector="runZero"
| json field=_raw "found" nodrop
| json field=_raw "rule_name" nodrop
| where rule_name = "<RULE NAME>"
```

Graph the data matching the runZero rule name

```
_source="runZero Alerts" and _collector="runZero"
| json field=_raw "assets_new" nodrop
| toLong(assets_new)
| json field=_raw "found" nodrop
| json field=_raw "search_url" nodrop
| json field=_raw "rule_name" nodrop
| where rule_name = "<RULE NAME>"
| timeslice 1m
| sum(found) by _timeslice
| order by _timeslice
```

Step 5: Create a dashboard in Sumo Logic (optional)

Now that you can search the data and graph it in Sumo Logic, build a dashboard. First create four rules in runZero, then import the sample Sumo Logic dashboard below.

Create the runZero rules

Create each of these rules the same way as in step 3 using the form inputs listed here.

Assets running a TLS service

- **Rule type**: asset-query-results
- **Name**: `Assets running a TLS service - Sumo`
- **Query**: `alive:t protocol:tls`
- **Number of matches**: is greater than 0
- **Notification channel**: alert channel created in Step 2
- **Notification template**: alert template created in Step 2

Multihomed assets - Sumo

- **Rule type**: asset-query-results
- **Name**: `Multihomed assets - Sumo`
- **Query**: `alive:t AND multi_home:t`
- **Number of matches**: is greater than 0
- **Notification channel**: alert channel created in Step 2
- **Notification template**: alert template created in Step 2

Assets with OpenSSL - Sumo

- **Rule type**: asset-query-results
- **Name**: `Assets with OpenSSL - Sumo`
- **Query**: `alive:t product:openssl`
- **Number of matches**: is greater than 0
- **Notification channel**: alert channel created in Step 2
- **Notification template**: alert template created in Step 2

New assets to Sumo Logic

- **Rule type**: new-assets-found
- **Name**: `New assets to Sumo Logic`
- **Number of matches**: is greater than 0
- **Notification channel**: alert channel created in Step 2
- **Notification template**: alert template created in Step 2

Sample Sumo Logic dashboard

1. Go to the library in Sumo Logic.
2. Click the **options** button on the folder you want to import into, then click `Import`.
    - **Name**: runZero Alert Metrics
    - **JSON**: copy the sample below
3. Click `Import`. The dashboard appears under that folder.

```
{
    "type": "DashboardV2SyncDefinition",
```

```
"name": "runZero Asset Metrics",
    "description": "",
    "title": "runZero Asset Metrics",
    "theme": "Dark",
    "topologyLabelMap": {
        "data": {}
    },
    "refreshInterval": 0,
    "timeRange": {
        "type": "BeginBoundedTimeRange",
        "from": {
            "type": "RelativeTimeRangeBoundary",
            "relativeTime": "-3d"
        },
        "to": null
    },
    "layout": {
        "layoutType": "Grid",
        "layoutStructures": [
            {
                "key": "panelPANE-AC8FB3DCBD32DA48",
                "structure": "{\"height\":6,\"width\":12,\"x\":0,\"y\":0}"
            },
            {
                "key": "panel3D084A3284252A4E",
                "structure": "{\"height\":6,\"width\":12,\"x\":12,\"y\":0}"
            },
            {
                "key": "panelPANE-4389DBF794B13B44",
                "structure": "{\"height\":6,\"width\":12,\"x\":0,\"y\":6}"
            },
            {
                "key": "panelPANE-FBE08549B2123A4A",
                "structure": "{\"height\":6,\"width\":12,\"x\":12,\"y\":6}"
            }
        ]
    },
    "panels": [
        {
            "id": null,
            "key": "panelPANE-AC8FB3DCBD32DA48",
            "title": "New assets found",
            "visualSettings": "{\"general\":{\"mode\":\"timeSeries\",\"type\":\"line\",\"displayType
\":\"default\",\"markerSize\":5,\"lineDashType\":\"solid\",\"markerType\":\"none\",\"lineThickness\":
1},\"title\":{\"fontSize\":14},\"axes\":{\"axisX\":{\"titleFontSize\":12,\"labelFontSize\":12},\"axisY
\":{\"titleFontSize\":12,\"labelFontSize\":12,\"logarithmic\":false}},\"legend\":{\"enabled\":true,\"v
erticalAlign\":\"bottom\",\"fontSize\":12,\"maxHeight\":50,\"showAsTable\":false,\"wrap\":true},\"colo
r\":{\"family\":\"Categorical Default\"},\"series\":{},\"overrides\":[]}",
            "keepVisualSettingsConsistentWithParent": true,
            "panelType": "SumoSearchPanel",
            "queries": [
                {
                    "transient": false,
```

```
d=_raw \"assets_new\" nodrop\n| json field=_raw \"found\" nodrop\n| json field=_raw \"search_url\" nod
rop\n| json field=_raw \"rule_name\" nodrop\n| where rule_name = \"New Assets to Sumo Logic\"\n| times
lice 1m\n| sum(assets_new) by _timeslice\n| order by _timeslice",
                    "queryType": "Logs",
                    "queryKey": "A",
                    "metricsQueryMode": null,
                    "metricsQueryData": null,
                    "tracesQueryData": null,
                    "spansQueryData": null,
                    "parseMode": "Auto",
                    "timeSource": "Message",
                    "outputCardinalityLimit": 1000
                }
            ],
            "description": "",
            "timeRange": null,
            "coloringRules": null,
            "linkedDashboards": []
        },
        {
            "id": null,
            "key": "panel3D084A3284252A4E",
            "title": "Multihomed assets found",
            "visualSettings": "{\"general\":{\"mode\":\"timeSeries\",\"type\":\"line\",\"displayType
\":\"default\",\"markerSize\":5,\"lineDashType\":\"solid\",\"markerType\":\"none\",\"lineThickness\":
1},\"title\":{\"fontSize\":14},\"axes\":{\"axisX\":{\"titleFontSize\":12,\"labelFontSize\":12},\"axisY
\":{\"titleFontSize\":12,\"labelFontSize\":12,\"logarithmic\":false}},\"legend\":{\"enabled\":true,\"v
erticalAlign\":\"bottom\",\"fontSize\":12,\"maxHeight\":50,\"showAsTable\":false,\"wrap\":true},\"colo
r\":{\"family\":\"Categorical Default\"},\"series\":{},\"overrides\":[]}",
            "keepVisualSettingsConsistentWithParent": true,
            "panelType": "SumoSearchPanel",
            "queries": [
                {
                    "transient": false,
                    "queryString": "_source=\"runZero Alerts\" and _collector=\"runZero\"\n| json fiel
d=_raw \"assets_new\" nodrop\n| toLong(assets_new)\n| json field=_raw \"found\" nodrop\n| json field=_
raw \"search_url\" nodrop\n| json field=_raw \"rule_name\" nodrop\n| where rule_name = \"Multihomed As
sets\" or rule_name = \"Multihomed Assets - Sumo\"\n| timeslice 1m\n| sum(found) by _timeslice\n| orde
r by _timeslice",
                    "queryType": "Logs",
                    "queryKey": "A",
                    "metricsQueryMode": null,
                    "metricsQueryData": null,
                    "tracesQueryData": null,
                    "spansQueryData": null,
                    "parseMode": "Auto",
                    "timeSource": "Message",
                    "outputCardinalityLimit": 1000
                }
            ],
            "description": "",
            "timeRange": null,
            "coloringRules": null,
```

```
},
        {
            "id": null,
            "key": "panelPANE-4389DBF794B13B44",
            "title": "Assets with a TLS service",
            "visualSettings": "{\"general\":{\"mode\":\"timeSeries\",\"type\":\"line\",\"displayType
\":\"default\",\"markerSize\":5,\"lineDashType\":\"solid\",\"markerType\":\"none\",\"lineThickness\":
1},\"title\":{\"fontSize\":14},\"axes\":{\"axisX\":{\"titleFontSize\":12,\"labelFontSize\":12},\"axisY
\":{\"titleFontSize\":12,\"labelFontSize\":12,\"logarithmic\":false}},\"legend\":{\"enabled\":true,\"v
erticalAlign\":\"bottom\",\"fontSize\":12,\"maxHeight\":50,\"showAsTable\":false,\"wrap\":true},\"colo
r\":{\"family\":\"Categorical Default\"},\"series\":{},\"overrides\":[]}",
            "keepVisualSettingsConsistentWithParent": true,
            "panelType": "SumoSearchPanel",
            "queries": [
                {
                    "transient": false,
                    "queryString": "_source=\"runZero Alerts\" and _collector=\"runZero\"\n| json fiel
d=_raw \"assets_new\" nodrop\n| toLong(assets_new)\n| json field=_raw \"found\" nodrop\n| json field=_
raw \"search_url\" nodrop\n| json field=_raw \"rule_name\" nodrop\n| where rule_name = \"Assets runnin
g a TLS service\" or rule_name = \"Assets running a TLS service - Sumo\"\n| timeslice 1m\n| sum(found)
by _timeslice\n| order by _timeslice",
                    "queryType": "Logs",
                    "queryKey": "A",
                    "metricsQueryMode": null,
                    "metricsQueryData": null,
                    "tracesQueryData": null,
                    "spansQueryData": null,
                    "parseMode": "Auto",
                    "timeSource": "Message",
                    "outputCardinalityLimit": 1000
                }
            ],
            "description": "",
            "timeRange": null,
            "coloringRules": null,
            "linkedDashboards": []
        },
        {
            "id": null,
            "key": "panelPANE-FBE08549B2123A4A",
            "title": "Assets running OpenSSL",
            "visualSettings": "{\"general\":{\"mode\":\"timeSeries\",\"type\":\"line\",\"displayType
\":\"default\",\"markerSize\":5,\"lineDashType\":\"solid\",\"markerType\":\"none\",\"lineThickness\":
1},\"title\":{\"fontSize\":14},\"axes\":{\"axisX\":{\"titleFontSize\":12,\"labelFontSize\":12},\"axisY
\":{\"titleFontSize\":12,\"labelFontSize\":12,\"logarithmic\":false}},\"legend\":{\"enabled\":true,\"v
erticalAlign\":\"bottom\",\"fontSize\":12,\"maxHeight\":50,\"showAsTable\":false,\"wrap\":true},\"colo
r\":{\"family\":\"Categorical Default\"},\"series\":{},\"overrides\":[]}",
            "keepVisualSettingsConsistentWithParent": true,
            "panelType": "SumoSearchPanel",
            "queries": [
                {
                    "transient": false,
                    "queryString": "_source=\"runZero Alerts\" and _collector=\"runZero\"\n| json fiel
```

```
raw \"search_url\" nodrop\n| json field=_raw \"rule_name\" nodrop\n| where rule_name = \"Assets with O
penSSL\" or rule_name = \"Assets with OpenSSL - Sumo\"\n| timeslice 1m\n| sum(found) by _timeslice\n|
order by _timeslice",
                    "queryType": "Logs",
                    "queryKey": "A",
                    "metricsQueryMode": null,
                    "metricsQueryData": null,
                    "tracesQueryData": null,
                    "spansQueryData": null,
                    "parseMode": "Auto",
                    "timeSource": "Message",
                    "outputCardinalityLimit": 1000
                }
            ],
            "description": "",
            "timeRange": null,
            "coloringRules": null,
            "linkedDashboards": []
        }
    ],
    "variables": [],
    "coloringRules": []
}
```

Tines

runZero integrates with Tines so you can automate workflows around your asset data. A video demo shows the end result of these instructions.

Requirements

- A Tines account
- runZero Export API and Organization API tokens

You can integrate runZero and Tines in two ways:

- Build a custom story in Tines step by step, or
- Import the **runZero sample story** and start from its outline.

Tines custom story

A Tines story is a collection of actions that work together toward a goal, like a playbook. The Tines Story Library has ready-made automated playbooks; when none matches your needs, these steps show you how to build a custom story.

Step 1: Creating a Tines story and adding runZero API credentials

1. Sign in to Tines and create a new story.
2. Use the + in the **credentials** section to add a **text** credential

.
    - **Name**: a name for the credential, for example `runzero_export_token`
    - **Value**: your runZero export API token, copied from the runZero organization page for the organization you want. Export API tokens start with `ET`.
3. Use the `+` in the **credentials** section to add another **text** credential.
    - **Name**: a name for the credential, for example `runzero_org_token`
    - **Value**: your runZero organization API token, copied from the runZero organization page for the organization you want. Organization API tokens start with `OT`.

Step 2: Creating a Tines webhook action

1. Add a **Webhook** action to your story.
2. Click the **Webhook** action and copy the webhook URL, which looks like `https://<tenant-name>.tines.com/webhook/<guid>`.

Step 3: Creating a runZero alert template

1. Create an alert template in runZero:
    - **Name**: a name for the template
    - **Template type**: JSON
    - **Subject line for message**: leave empty
    - **Body of message**: this JSON example includes the rule name and the search URL in the alert message body: `{"rule_name":"{{rule.name}}","search_url":"{{search.url}}"}`
2. Create an alert channel in runZero:
    - **Name**: a name for the alert channel
    - **Channel type**: webhook
    - **Webhook URL**: the webhook URL you copied from Tines

Step 4: Creating a rule in runZero

With the template ready, pick the triggers to alert on. Common examples:

- **Asset query results**: a query in runZero matches after a scan completes
- **New assets found**: a scan completes with new assets
- **Agent offline**: your runZero Explorer stops checking in to the console
- **Task failed**: a task fails for any reason

The rest of the steps use **asset query results**, with any asset that has an open Telnet port (`port:23`) as the example. You can see more example queries; for more on rules and alerts, see Using the rules engine.

1. Create an alert rule.
2. Select `asset-query-results` and click *Configure rule*.
3. Enter values for the rule:
    - **Name**: the name of the rule.
    - **Conditions**: optional parameters; the alert triggers when all conditions match.
        - **Query**: the query the assets must match, `port:23` in this example.
        - **Number of matches**: the numeric comparison to apply to the **value**.
        - **Value**: the match threshold that triggers the rule.
        - **Limit to organization**: limits the alert to a specific organization in runZero.

- **Limit to site**: limits the alert to a specific site in runZero.
- **Action**: notify
  - **Notification channel**: the alert channel you created in step 3.
  - **Notification template**: the alert template you created in step 3.

Step 5: Handling the data in Tines

1. The **Webhook** action you created in Step 2 is the entry point for the runZero alert in Tines.
2. Add an **Event Transform** to parse the search out of the URL in the runZero alert. If you connect the **Webhook** action to this one, some parameters populate automatically.
   - Mode: extract
   - Matchers:
     - Path: `path_to_alert_search`, for example `receive_alerts_from_runzero.body.search_url` if your **Webhook** action is named "RECEIVE alerts from runZero"
     - Regex: `\?.*`
     - Extract to: `search`
3. Add an Event Transform that takes the output of the previous step. If you connect the previous **Event Transform** action to this one, some parameters populate automatically.
   - Mode: explode
   - Matchers:
     - Path: `path_to_extraction_transform`, for example `get_search_from_url.search` if your previous **Event Transform** action is named "GET search from URL"
     - To: `individual_item`
4. Add an HTTP Request that calls the runZero Export API.
   - URL: the `path_to_explosion_transform` appended to the runZero API endpoint URL, for example `https://console.runZero.com/api/v1.0/export/org/assets.json<<get_search_from_regex.individual_item>>` if your previous **Event Transform** action is named "GET search from regex"
   - Content Type: JSON
   - Method: get
   - Use the **+ Option** button to add **Headers**:
     - Change `header` to `Authorization`
     - Change `value` to `Bearer CREDENTIAL.runzero_export_token`
5. Add an Event Transform that takes the runZero Export API output and loops through each value.
   - Mode: explode
   - Path: `path_to_HTTP_request`, for example `get_assets_from_runzero.body` if your **HTTP Request** action is named "GET assets from runZero"
   - To: `individual_item`

Step 6: Personalized automation in Tines

With the alert received, the search parsed, and the assets fetched from the runZero Export API, it's time to add your own flair. The sample story includes these two actions as examples, though you will likely want something more tailored.

1. Send an email to the destination of your choice for each asset. The sample JSON shows how to put the context you want in the Subject and Body of the email.

2. The HTTP Request action shows how to reach back into runZero and tag the assets after you gather other data.

runZero sample story

You can import the sample story below into Tines to get all of the actions from the steps above. Save its contents as a JSON file and upload it with the **Import** button in Tines. After importing, complete these steps.

Step 1: Update credentials in Tines

1. Replace the `runzero_export_token` credential:
    - **Name**: a name for the credential, such as `runzero_export_token`
    - **Type**: text
    - **Value**: your runZero export API token, copied from the runZero organization page for the organization you want. Export API tokens start with `ET`.
2. Replace the `runzero_org_token` credential:
    - **Name**: a name for the credential, such as `runzero_org_token`
    - **Type**: text
    - **Value**: your runZero organization API token, copied from the runZero organization page for the organization you want. Organization API tokens start with `OT`.

Step 2: Create runZero alert template

1. Create an alert template in runZero:
    - **Name**: a name for the template
    - **Template type**: JSON
    - **Subject line for message**: leave empty
    - **Body of message**: this JSON example includes the rule name and the search URL in the alert message body: `{"rule_name":"{{rule.name}}","search_url":"{{search.url}}"}`
2. Create an alert channel in runZero:
    - **Name**: a name for the alert channel
    - **Channel type**: webhook
    - **Webhook URL**: the webhook URL you copied from Tines

Step 3: Create the rule in runZero

With the template ready, pick the triggers to alert on. Common examples:

- Asset query results: a query in runZero matches after a scan completes.
- New assets found: a scan completes with new assets.
- Agent offline: your runZero Explorer stops checking in to the console.
- Task failed: a task fails for any reason.

The rest of the steps use **asset query results**, with any asset that has an open Telnet port (`port:23`) as the example. You can see more example queries.

1. Create a new alert rule.
2. Select `asset-query-results` and click `Configure rule`.

3. Enter values for the rule:
    - **Name**: the name of the rule.
    - **Conditions**: optional parameters; the alert triggers when all conditions match.
        - **Query**: the query the assets must match, `port:23` in this example.
        - **Number of matches**: the numeric comparison to apply to the **value**.
        - **Value**: the match threshold that triggers the rule.
        - **Limit to organization**: limits the alert to a specific organization in runZero.
        - **Limit to site**: limits the alert to a specific site in runZero.
        - **Action**: notify
            - **Notification channel**: the alert channel you created in step 2.
            - **Notification template**: the alert template you created in step 2.

runZero sample story JSON


```
{
  "schema_version": 4,
  "standard_lib_version": 6,
  "name": "runZero Sample Story",
  "description": null,
  "guid": "7706b502e51b4c68f0dbe9721c88d665",
  "slug": "runzero_sample_story",
  "exported_at": "2022-10-27T22:22:37Z",
  "agents": [
    {
      "type": "Agents::WebhookAgent",
      "name": "RECEIVE alerts from runZero",
      "disabled": false,
      "guid": "ab1f5bf7be49c943d893ab993ade9421",
      "options": {
        "path": "da6e90c0b276bdda97e6bfa31ad50787",
        "secret": "5e290305e0a483bf843f1213f0f21dda",
        "verbs": "get,post"
      },
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null
    },
    {
      "type": "Agents::HTTPRequestAgent",
      "name": "GET assets from runZero",
      "disabled": false,
      "guid": "d54d8f3f5b0ea857308f45f61a224547",
      "options": {
        "url": "https://console.runZero.com/api/v1.0/export/org/assets.json<<get_search_from_regex.ind
```

```
"content_type": "application_json",
        "method": "get",
        "headers": {
          "Authorization": "Bearer <<CREDENTIAL.runzero_export_token>>"
        }
      },
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": []
    },
    {
      "type": "Agents::EventTransformationAgent",
      "name": "GET search from URL",
      "disabled": false,
      "guid": "d62ddd189bb9080d778d900f2292504d",
      "options": {
        "mode": "extract",
        "matchers": [
          {
            "path": "=receive_alerts_from_runzero.body.search_url",
            "regexp": "\\?.*",
            "to": "search"
          }
        ]
      },
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": null
    },
    {
      "type": "Agents::EventTransformationAgent",
      "name": "GET search from regex",
      "disabled": false,
      "guid": "41afc5c4d4c0fdb694d14e1eb380688a",
      "options": {
        "mode": "explode",
        "path": "=get_search_from_url.search",
```

```
},
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": null
    },
    {
      "type": "Agents::EventTransformationAgent",
      "name": "LOOP through assets for follow up actions",
      "disabled": false,
      "guid": "b9ada61162bdf006f09d264845ebd304",
      "options": {
        "mode": "explode",
        "path": "=get_assets_from_runzero.body",
        "to": "individual_item"
      },
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": null
    },
    {
      "type": "Agents::EmailAgent",
      "name": "SEND Email Action",
      "disabled": false,
      "guid": "0e9647b114f315d480f1f85c58e481d3",
      "options": {
        "recipients": "youremail@email.com",
        "reply_to": "youremail@email.com",
        "sender_name": "Your Name",
        "subject": "<<receive_alerts_from_runzero.body.rule_name>>",
        "body": "Alert: <<receive_alerts_from_runzero.body.rule_name>>\n\\<br />\n\\<br />\nLink to as
set: https://console.runzero.com/inventory/<<loop_through_assets_for_follow_up_actions.individual_ite
m.id>>\n\\<br />\n\\<br />\nAddresses: <<loop_through_assets_for_follow_up_actions.individual_item.add
resses>>\n\\<br />\n\\<br />\nServices:\n<<NEWLINE_TO_BR(NEAT_JSON(loop_through_assets_for_follow_up_a
ctions.individual_item.services))>>"
      },
      "reporting": {
        "time_saved_value": 0,
```

```
},
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": null
    },
    {
      "type": "Agents::HTTPRequestAgent",
      "name": "TAG assets in runZero",
      "disabled": false,
      "guid": "c3171fed91ec40d39572260295a63ae0",
      "options": {
        "url": "https://console.runZero.com/api/v1.0/org/assets/<<loop_through_assets_for_follow_up_ac
tions.individual_item.id>>/tags",
        "content_type": "application_json",
        "method": "patch",
        "headers": {
          "Authorization": "Bearer <<CREDENTIAL.runzero_org_token>>"
        },
        "payload": {
          "tags": "hello=from_tines"
        }
      },
      "reporting": {
        "time_saved_value": 0,
        "time_saved_unit": "minutes"
      },
      "monitoring": {
        "monitor_all_events": false,
        "monitor_failures": false,
        "monitor_no_events_emitted": null
      },
      "width": null,
      "schedule": []
    }
  ],
  "diagram_notes": [
    {
      "content": "Hello! Thanks for importing the runZero Sample Story. \n\nThis story is a basic exam
ple of what you can do with runZero and Tines. \n\nPrerequisites:\n1. Create a text [credential](http
s://www.tines.com/docs/credentials/text) called `runzero_export_token`. Set the value as your [runZero
export token](docs/leveraging-the-api.md), which can be obtained from the desired [runZero organizatio
n](https://console.runzero.com/organizations) page. Export tokens start with `ET`.\n2. Create a text
[credential](https://www.tines.com/docs/credentials/text) called `runzero_org_token`. Set the value as
your as your [organization API token](docs/leveraging-the-api.md), which can be obtained from the desi
red [runZero organization](https://console.runzero.com/organizations). Organization API tokens start w
ith `OT`. \n\nThe steps are as follows:\n\n1. **RECEIVE alerts** provides the webhook destination for
your runZero alerts \n\n2. **GET search from URL** parses out the search string from the URL provided
in the runZero alert \n\n3. **GET search from regex** takes the list provided in step 2 and allows you
to use the individual value \n\n4. **GET assets from runZero** uses the runZero Export API to get the
```

```
t of assets and sends each to the next steps individually \n\n6. **NOTE**: this is the step that you c
ould implement more custom logic in most cases. All of the initial runZero and Tines data transfer is
done, but it's your chance to customize this story to fit your use case. \n\n7. **SEND email action**
simply sends an email to the destination of your choice \n\n8. **TAG assets in runZero** adds a tag in
runZero to each asset showing an example of how you might reach back into runZero after doing other au
tomated activities ",
      "position": [
        165.0,
        180.0
      ],
      "guid": "0e4b9b331dc65316849f7b089bf7bde2",
      "width": 375
    }
  ],
  "links": [
    {
      "source": 0,
      "receiver": 2
    },
    {
      "source": 1,
      "receiver": 4
    },
    {
      "source": 2,
      "receiver": 3
    },
    {
      "source": 3,
      "receiver": 1
    },
    {
      "source": 4,
      "receiver": 6
    },
    {
      "source": 4,
      "receiver": 5
    }
  ],
  "diagram_layout": "{\"ab1f5bf7be49c943d893ab993ade9421\":[555,180],\"d54d8f3f5b0ea857308f45f61a22454
7\":[555,450],\"d62ddd189bb9080d778d900f2292504d\":[555,270],\"41afc5c4d4c0fdb694d14e1eb380688a\":[55
5,360],\"b9ada61162bdf006f09d264845ebd304\":[555,555],\"0e9647b114f315d480f1f85c58e481d3\":[555,69
0],\"c3171fed91ec40d39572260295a63ae0\":[780,690]}",
  "send_to_story_enabled": false,
  "entry_agent_guid": null,
  "exit_agent_guids": [],
  "exit_agent_guid": null,
  "keep_events_for": 604800,
  "reporting_status": true,
  "send_to_story_access": null,
  "send_to_stories": [],
  "form": null,
```

```
"forms": []
}
```

Thinkst Canary

All runZero editions integrate with Thinkst Canary for quick access from the Canary console to your asset data in the runZero Console. Setup is one Canary settings change.

Integrate runZero with Thinkst Canary

1. Sign in to your runZero console.
2. Sign in to your Canary console.
3. Go to **Global Settings** under the gear icon.
4. Click **Integrations**.
5. Toggle the runZero switch.

Accessing runZero data from Canary

After you enable the integration in Canary, runZero data is available from any incident. When viewing an incident, click the magnifying glass icon under `source IP` or `reverse IP lookup` to open a search in your runZero console. To refine that search, see Search query syntax.

Exposure management

runZero gives you visibility across your whole internal and external attack surface: IT, OT, IoT, mobile, and cloud. It finds unknown and unmanageable assets, reveals hard-to-find exposures, and targets the risks other approaches miss, with no agents, no authentication, and no appliances.

Unified asset inventory

runZero builds one asset inventory from active scanning, passive traffic sampling, and integrations, backed by deep fingerprinting and correlation. Assets are normalized, deduplicated, and tracked as they move across your environment.

Total attack surface management

runZero manages the attack surface of external, internal, cloud, IoT, and OT environments.

External & cloud

runZero hosts scan engines for external monitoring. A scan scope takes CIDRs, hostnames, and IPs, plus keywords that find targets for you:

- `domain:runzero.com`: finds every host associated with runZero.com, then scans both the IPv4 and IPv6 addresses of the results.
- `asn4:1233`: resolves every CIDR associated with ASN 1233 and scans them.
- `country4:us`: can scan an entire country, but more often goes in the Excludes field to limit external discovery to a particular region.
- `defaults`: expands to every CIDR and hostname registered in the associated site.

You can combine these keywords and use them in both Scope and Exclusions. When you take the `Initialize` action on the scan page, runZero previews the expanded targets and estimates the scan time from the address count and full scope.

Integrations also identify and monitor the external attack surface. Assets imported from cloud providers like AWS, Azure, and GCP arrive with their external IPs and hostnames.

Internal

runZero enumerates your internal attack surface with active scans, passive discovery, and integrations.

The runZero scanner's subnet sampling discovers massive internal ranges fast. It trickles packets into the most commonly used octets of each /24 and scans only the subnets where at least one reply came back. It also parses ICMP error messages to identify the network ranges used by the network equipment itself. Use the `RFC1918` report to find blind spots and launch scans to close the gaps.

Beyond scans, runZero imports PCAP files directly through the web interface and turns any deployed Explorer into a passive network sensor. Passive discovery uses up to one CPU core to parse all traffic passing through the interface, including broadcast, SPAN port data, and encapsulated traffic (VLAN, VXLAN, GRE, etc.).

Integrations show which internal endpoints are missing a particular security control. If you use CrowdStrike as your EDR, runZero can flag every Windows system missing the agent. The same approach finds gaps in vulnerability management scopes.

IoT & OT

runZero was built for fast, unauthenticated, safe scanning of every device in every environment. That gives you coverage of internet of things (IoT) devices and reliable

fingerprinting of operational technology (OT) equipment. runZero worked with the US Department of Energy to make sure active discovery is safe and accurate for OT environments.

If you can't deploy active scanning, you can still use passive discovery, PCAP imports, and integrations with OT and IoT management tools.

Inside-out

Inside Out Attack Surface Management (IOASM) compares the unique fingerprints of your internal assets with a global database of public endpoints. It's enabled by default on the runZero-hosted platform and available to self-hosted customers through additional configuration. IOASM can quickly tell you if an internal device is publicly exposed through an unexpected IP, including port forwards, VPNs, IPv6 tunnels, and more. This presentation goes deeper.

IOASM reports four distinct vulnerabilities based on heuristics and detection type:

- (TLS|SSH) Private Key is Public
- (TLS|SSH) Private Key is Widely Shared
- Potential External Access to Internal Asset (SSH, TLS)
- Potential External Access to Internal Asset (MAC Address)

The reported risk also varies with the heuristics. A public service attached to a Remote Desktop or Secure Shell system counts as higher risk than a public TLS key reused by a web application.

IOASM also uses the BadKeys compromised key database.

Vulnerability detection

runZero identifies, imports, and manages vulnerabilities across your total attack surface through query-based detection logic, active scans, and API-based integrations with endpoint management and vulnerability management platforms.

Findings give you one view of every detected security issue, misconfiguration, and other exposure across your assets. See Findings for more.

The Rapid Response program detects zero-hour exposures and notifies you in real time.

The Query Library lists every runZero-provided query. You can also create custom queries that report vulnerabilities on matching assets and services.

Scans find vulnerabilities with an embedded version of Nuclei, an open source vulnerability scanner, paired with curated templates and runZero's network discovery and fingerprinting engine. You pick vulnerability categories in the scan configuration, and runZero chooses specific templates dynamically by precisely matching assets and services. Out-of-band testing relies on runZero-hosted Interactsh services at `reflect-us.rumble.network` and `reflect-eu.rumble.network`.

runZero recalculates vulnerabilities, findings, and asset risk as part of task processing and shows the results in their respective product sections.

Risk prioritization

runZero assigns every asset a normalized risk score. Threat intelligence, vulnerability information, and exposure measurement all influence the score. You can set asset criticality through automated rules or by hand in the product interface, and assign assets to specific owners for remediation. Tags come in from API integrations and can be managed in the interface, including rule-based tagging. Exported assets carry the risk, criticality, and tag information set in the product interface.

Continuous monitoring

runZero continuously monitors your organization for changes to exposure at the asset and service level. Recurring active scans, background passive traffic sampling, and regular sync with your existing infrastructure catch and report new risks quickly. Alerts can be managed in-product, sent by email, or delivered by webhook to the platform of your choice. All asset data can be synced to external platforms, including popular SIEMs and data lakes.

Rapid responses

The runZero Rapid Response program detects emerging threats and notifies you immediately. Older entries move to standalone queries or templates.

Rapid responses omitted from PDF. See help.runzero.com for the full list.

Each Rapid Response includes a query that finds matching assets, a trigger that analyzes all inventories for exposure, and a blog post with the details of the issue, so you can mitigate exposures before compromise.

> As mitigations become available, runZero replaces a Rapid Response entry with more specific coverage.

Query library

Query library omitted from PDF. See help.runzero.com for the full list.

runZero ships a library of pre-built queries, categorized by use case and risk level. Use them to detect vulnerabilities, trigger alerts, and apply changes to assets such as tags and ownership. You can also configure custom queries to report vulnerabilities on matching assets and services. To write your own, start with the search query syntax.

Templates

This table lists the Nuclei vulnerability templates available for scans. The full set of tuned templates lives in our nuclei-templates repository.

Vulnerability templates omitted from PDF. See help.runzero.com for the full list.

Alongside query-based vulnerability reporting, runZero detects exposures natively with an embedded version of the open-source Nuclei vulnerability scanner and its YAML-based vulnerability check templates. To keep scans fast and network disruption low, runZero selects templates dynamically based on the scan's configured categories and precise asset and service fingerprinting.

Findings

Findings are groups of significant exposures with the same root cause.

You can promote one or more vulnerabilities from a finding into a runZero issue for tracking and remediation, and optionally push the issue to Jira as an external ticket.

Findings summary omitted from PDF. See help.runzero.com for the full list.

Findings come from three sources:

1. Query-based findings come from specific queries defined within the system (see list above).
2. Nuclei-generated findings come from scans with default credentials and vulnerability checks enabled, using Nuclei templates (see templates).
3. KEV (Known Exploited Vulnerabilities) findings trigger when a discovered vulnerability appears in the CISA Known Exploited Vulnerabilities (KEV) catalog or VulnCheck KEV.

For more on findings, see Understanding findings.

Issues

Issues group one or more vulnerabilities for tracking, prioritization, and remediation. An issue captures a point-in-time **snapshot** of every (vulnerability, asset) pair it contains, so the original context survives even if the asset or vulnerability later changes.

Use issues to:

- Roll several related vulnerability instances up into a single unit of work.
- Assign a risk level, state, and due date, and track progress over time.
- Record decisions and discussion in a per-issue timeline.
- Push the issue to Jira as an external ticket and keep the runZero timeline in sync through a Jira webhook.

What is an issue

Each issue has these fields:

- **Name** is a short label that identifies the issue.
- **Description** holds longer free-form context about the issue.
- **Remediation** holds guidance for the people doing the work.
- **Risk** is one of `none`, `low`, `medium`, `high`, or `critical`. The default is the highest risk found across the issue's vulnerabilities at creation time.
- **State** is one of `open`, `in progress`, `closed`, or `resolved`. Unless the state is `closed`, runZero sets it to `resolved` automatically once every instance is remediated (see Automatic remediation tracking below). A Jira webhook state mapping can also set the state.
- **Due date** is an optional target date for resolving the issue.
- **Owner** is the single user responsible for the issue. Issues use the built-in `Issue owner` ownership type and cannot take custom ownership types. See Issue ownership for details.
- **Instances** are the (vulnerability, asset) pairs captured in the snapshot.

Creating an issue

You can create issues from anywhere in the runZero Console that shows vulnerabilities or groups of vulnerabilities:

- The vulnerability groups inventory
- The findings inventory
- The vulnerabilities by asset inventory table
- The vulnerabilities card on an asset details page
- The vulnerabilities table on a findings details page
- The vulnerabilities table on a certificate details page
- A vulnerability details page

Select one or more vulnerabilities and click **Create issue**. runZero opens a draft issue where you can edit the name, description, solution, risk, and due date before saving. Saved issues

appear on the Issues page for the current organization.

At creation time, runZero stores a snapshot of each selected vulnerability and its asset on the issue, so the issue keeps showing the original instance details even if the underlying asset is later modified, deleted, or rescanned.

Working with an issue

The issue details page has these sections:

- **Header** shows the name plus quick-edit risk, state, and due date.
- **Summary** shows the description and remediation guidance.
- **Instances** is the table of vulnerabilities captured in the snapshot, with a slideout for vulnerability and snapshotted asset details. Instances whose vulnerability has since been remediated are flagged.
- **Comments** and **History** tabs hold the issue timeline (see below).
- **Sidebar** shows risk, state, due date, and the linked Jira ticket, if one exists.

Comments and history

The issue timeline records three kinds of entries:

- **User events** cover changes to the issue fields and comments a runZero user writes on the **Comments** tab.
- **System events** are entries runZero generates automatically (state changes, ticket creation, and similar actions).
- **Jira events** arrive through the Jira webhook, one timeline line per changelog item on the linked ticket.

The **Comments** tab shows comments only; the **History** tab shows everything in the timeline.

Issue ownership

An issue can have one owner. Issue ownership is separate from the ownership types used for assets and vulnerability records: every issue uses the built-in `Issue owner` type, and custom ownership types cannot be added to an issue. When an issue is linked to Jira, runZero can sync its owner with the Jira assignee if it finds a matching external user. See Issue owner synchronization.

Automatic remediation tracking

Whenever a scan, import, or integration import finishes for an organization, runZero recomputes the **remediated instances** counter for every issue in that organization except closed ones. An instance counts as remediated when the original vulnerability is no longer present on the original asset.

If the state is `closed`, runZero does not recompute the remediated instances and makes no automated changes to the state.

When every instance on an issue is remediated, runZero moves the issue to the `resolved` state.

If a `resolved` issue has one or more of its vulnerabilities detected on the original asset again, runZero moves it back to `in progress` to alert you to the regression.

Linking a Jira ticket

To push an issue to Jira as an external ticket, first create a Jira credential. The Issues page sidebar shows a **Create Jira ticket** action where you pick the Jira credential and, optionally, upload a CSV of the issue's instance snapshot as a Jira attachment. After the ticket is created, the runZero issue stores the Jira issue key, a deep link to the ticket, and the credential and Jira base URL used. The Jira integration page has the full details.

Permissions

- Viewing issues requires read access to the organization the issue belongs to.
- Creating, editing, and deleting issues requires write access to that organization.
- Issues don't appear in the **All organizations** view; switch into a specific organization to use the Issues page.

Data analysis

Reviewing results

Task details

After each discovery task completes, the task details page summarizes how many assets were updated. runZero correlates assets across IPs and data sources, so the numbers can differ from what IP-based matching alone would give.

The change summary on the task details page includes these statistics:

- Asset changes:
  - **Newly discovered assets** are devices found during the task whose fingerprints match no device seen before.
  - **Assets marked offline** are assets runZero had seen before on the scanned network that didn't respond on any of their IP addresses during this scan. The offline status is a flag on the asset and doesn't count as a change to it. A device may be marked offline because it was powered down or disconnected, or because of network problems.
  - **Assets back online** are assets that were marked offline at some point but responded to the runZero Explorer during this scan. The online status is a flag on the asset and doesn't count as a change to it.
  - **Assets changed** is the number of assets where some property other than online status changed, such as the device's IP addresses or hostname, or responses from new ports or protocols.
  - **Assets unchanged** is the number of assets seen exactly where runZero found them in the last scan, with no change to their responses.
  - **Assets ignored** is the number of times the Explorer got a response from probing an IP address that turned out to be bogus in some way. This typically happens when a web proxy, stateful firewall, or SIP gateway answers as if it were the asset at every address on a subnet.
  - **Total assets seen by task** is **Assets changed** plus **Assets unchanged**: the number of asset records that are now up to date.
- User changes:
  - **Newly discovered users** are users seen for the first time during the integration sync.
  - **Users changed** are users whose attributes changed during the integration sync.
  - **Users unchanged** are users that did not change during the integration sync.
  - **Users updated by task** is **Users changed** plus **Users unchanged**: how many user records are now up to date.
- Group changes:
  - **Newly discovered groups** are groups seen for the first time during the integration sync.
  - **Groups changed** are groups whose attributes changed during the integration sync.
  - **Groups unchanged** are groups that did not change during the integration sync.

- **Groups updated by task** is **Groups changed** plus **Groups unchanged**: how many group records are now up to date.

Dashboard & inventory views

The dashboard fills with results after the first scan completes. It shows trend data and insights that help you see how your inventory changes over time. Pick a time period and site for the trend data with the selectors at the top right of the dashboard page.

The main asset trends graph shows the number of assets in each of the four main states: live, offline, scanned, and unscanned. Beneath the graph, further breakdowns show the top 10 in each asset category: asset type, operating system, hardware, and tags.

The service trends graph shows how many services in total were found in your asset inventory, with breakdowns for ARP, ICMP, TCP, and UDP. Below it are the top 10 TCP ports, UDP ports, protocols, and products detected.

Click the menu button at the top right of a table and select “View more” for a more detailed inventory of that category.

Insights from queries

Queries and reports help you draw insights from your inventory. Start with the pre-built queries in the Query Library. Some of them come from runZero’s Rapid Response to emerging threats and are described on our blog.

The runZero query language searches and filters your asset inventory by asset field and value pairs. The search query syntax page documents the language. Once you know it, you can write your own queries.

To run a query automatically, open it and set “Automatically track query results on the dashboard”. The query then runs when scans complete, and the dashboard page notifies you of any resulting insights.

Sample Queries

Asset inventory

- Equipment that is likely 8+ years old: `alive:t mac_age:>8years`
- Assets with end-of-life OS: `os_eol:<now`
- Virtual machines: `has:virtual`
- Devices acting as a router: `router:true`
- Devices that may be bridging: `has_public:t and has_private:t`

Service inventory

- Protocol on a non-standard port example: `protocol:ssh not port:22`

- Publicly addressed assets running RDP or VNC: `has_public:t and (protocol:rdp or protocol:vnc)`
- Authenticated web services that are not encrypted: `(_asset.protocol:http AND not _asset.protocol:tls) AND ( html.inputs:"password:" OR last.html.inputs:"password:" OR has:http.head.wwwAuthenticate OR has:last.http.head.wwwAuthenticate )`
- Older TLS versions in use: `alive:t AND protocol:"=tls" AND ( tls.versionName:"=TLS 1.0" OR tls.versionName:"=TLS 1.1")`

Our blog describes more sample queries:

- Finding duplicate SSH host keys
- Identifying rogue remote access solutions
- Finding device serial numbers

Reports

After the dashboard and inventory, your next stop is the runZero Reports page.

Switch topology

This report uses SNMP information to map how the switches on your network are connected. Each switch shows its IP address, name, and the number of assets connected to it. If runZero detected MAC addresses that weren't found within the scan scope, a count of unmapped assets appears below the switch.

Click a switch for a pop-up with the number of identified and unmapped assets. From there, click through to view the unmapped assets in a table of unmapped MACs by switch port.

Double-click a switch to expand that part of the diagram and show the individual assets connected to it.

The switch topology report isn't always exact. It relies on which switch claims to have seen each MAC address, and that may not be the nearest access switch. The algorithm picks the switch port with the fewest shared MACs as the best match, but switch cache timeouts and how the switches were scanned can still produce an answer you don't expect.

Subnet utilization

The subnet utilization report lists the subnets scanned on your network and what percentage of each is in use. If you scanned 10.0.1.0/24 and found 25 assets, the report shows 10% of the subnet's available IP addresses in use.

Network bridges

The network bridges report finds devices that bridge multiple network segments, which helps you locate unintentional bridging between your internal networks and the Internet.

The report shows your internal networks in green and external networks in red, then the multihomed assets that bridge an internal network to an external one.

RFC 1918 coverage

The RFC 1918 coverage report shows how much of the private internal address space you have scanned for assets. It helps you find rogue assets, unscanned subnets, and secondary interfaces on scanned devices. The section on coverage reports has more.

Unmapped MACs

This report uses SNMP information to list MAC addresses runZero found evidence for but never encountered as asset addresses during the network scan. The addresses are grouped by the switch that reported them, along with the vendor, manufacture date, and switch port of the possible asset, to help you identify it.

Outliers

The outliers reports summarize how often different values occur in specific attributes of assets and services, sorted from most frequent to least.

The HTTP servers outliers report, for example, lists every HTTP server runZero encountered, starting with the most common.

Beyond the one-click outliers reports, you can produce an outliers report for any asset or service attribute.

- Switch topology shows how your assets are connected and finds “unmapped” MAC addresses (in red) that weren’t in your scan scope (the Unmapped MACs report summarizes them)
- Bridging visualizes which hosts may have both public and private connections
- RFC 1918 coverage can identify potential blind spots on your network, like missing (unscanned) subnets, rogue devices, and “hinted” IPs that are secondary interfaces on unscanned network ranges
- The “View all” button at the top right lists other reports for investigating outliers

Domain membership report

The domain membership report lists the Active Directory domains runZero encountered and how many assets are in each.

Analysis reports

Analysis reports are the more advanced reports. They may run as tasks rather than being generated on the fly.

The first analysis report is Compare Sites, which generates searchable reports of the differences between two sites.

The Outlier Overview Report analyzes assets across the organization and summarizes the most unusual values for attributes such as hardware type and SNMP enterprise ID.

The Specific Outlier Report breaks down the outlying values of an attribute you select.

The Organization Overview Report builds a high-level summary of the entire organization and can optionally include lists of the assets found.

Alerts

Beyond the reports and queries you run yourself, runZero can send automatic alerts to designated channels for post-scan inventory queries, asset changes, Explorer and scan issues, security operations, or API events.

Channels are internal notifications in the runZero web console, email, or webhooks that connect to services such as Slack or Mattermost. Alerts use the same query language as the sample queries above, so they’re a good way to automate notification for critical events.

Using dashboards

Dashboards are customizable visual views into your asset inventory. You can create them for different use cases such as compliance, vulnerability remediation, or asset visibility.

Visualization widgets show operational information, trends, insights, goals, sources, and most and least seen graphs. You can also build your own custom widget from queries to show exactly the data you want, as a trend line or a latest count.

Filter the dashboard data by site and time bucket with the dropdown buttons at the top of the dashboard.

In Edit Mode you can customize the arrangement and size of each widget, and the widget library lets you add or remove widgets.

Dashboard selection

The dashboard selector button at the top of the dashboard shows the current dashboard's name. Click it to open a menu of runZero-managed dashboards, any shared dashboards, and your personal customized dashboards, then click the name of the dashboard you want to view.

Creating dashboards

To create a personal dashboard, click the dashboard selector, then "Create new dashboard". runZero creates the dashboard and takes you straight into Edit Mode, where you can add, resize, and arrange widgets.

Customizing dashboards

To customize a personal dashboard, click "Edit" in the dashboard header. The dashboard enters "Edit Mode", where you can resize, rearrange, add, and remove widgets.

In Edit Mode, resize a widget by dragging its bottom right corner.

Rearrange widgets by dragging the widget heading area, where the mouse cursor turns into a "cross-hair" icon.

When the layout looks right, click "Save". The dashboard keeps that layout, so the information you want is there every time you visit.

To revert instead, click "Cancel" to discard your changes. The visibility, arrangement, and size of every widget return to how they were before you started editing.

Sharing dashboards

Share a dashboard with any number of organizations by clicking “Share”. In the dialog, select the organizations to share with and click “Done”.

When another user shares a dashboard with your organization, it appears under “Shared with me” in the dashboard selector.

Dashboards you share still appear under “My dashboards”. Only you can modify them, but other organization users may duplicate them.

To share a read-only snapshot of a dashboard with someone outside runZero, use external reporting instead. On shareable dashboards, the Share button creates an email-gated public link, optionally on a recurring schedule.

Duplicating dashboards

A runZero-managed dashboard, or one shared with your organization, can’t be modified directly, but you can click “Duplicate” to make a copy you can customize.

You can also duplicate any dashboard from the menu button at the top of the dashboard by selecting “Duplicate dashboard”. A dialog opens where you can name the new dashboard to tell it apart from the original.

Sharing settings from the original dashboard are not copied to the new one.

Default dashboards

Set any dashboard as your preferred dashboard by clicking the dashboard menu button and selecting “Set as my preferred dashboard”.

An organization admin can set the default dashboard for every user of the selected organization with the “Set dashboard as organization default” option. In the dialog, the “Override user-preferred dashboard” checkbox makes the organization default win over each user’s preferred dashboard. With that override in place, users see their own preferred dashboard only if the organization has no default dashboard set to override it.

Deleting dashboards

To delete a custom dashboard, click the menu button at the top of the dashboard, select “Delete dashboard”, and confirm in the dialog. Deletion cannot be undone.

Widget library

Use the dashboard widget library to add and remove widgets. Open it with the “Widgets” button at the top right of the dashboard.

The library lists every available widget. You can also create your own custom widget from queries.

Widgets already on the dashboard carry a checkmark icon. Click one to deselect it, and it leaves the dashboard when you click “Save”.

Click a widget that isn’t on the dashboard to select it, then save to add it.

Widget types

Operational information

- Live assets: number of assets currently alive based on the latest scans
- Active scans: number of scans currently in progress
- Explorers online: number of Explorers reporting as healthy
- Accounts: number of users with access to your console

Trends

- Asset trends: timeline of asset counts broken out by live, recent, and a couple of other states
- Service trends: timeline of network services seen in your environment

Insights

- Latest insights: automated queries that ran after your latest scan, with result counts

Most and least seen graphs

Asset views by different asset properties. You can toggle each graph between most and least seen by count.

- Asset types
- Operating systems
- Hardware
- RRT latency
- MAC vendors
- Newest MAC age
- TCP ports
- UDP ports
- Products
- Protocols
- Address count
- Extra address count
- Risk
- Criticality

- Tags

Goals

- Goals overview: summary of every goal pinned to the dashboard

Asset count and trend tables

The latest count and historical timeline for your assets by ownership coverage, source type, and custom integration.

- Asset ownership: view of all ownership types, including assets missing owners
- Asset sources: view of all sources of asset data from supported integrations
- Custom integrations: view of all custom sources of asset data from API integrations

Bookmarks

A customizable list of frequently visited pages, favorite reports, or external links. After adding the widget to your dashboard, manage the list from the menu button at the top of the widget.

To edit a bookmark, select “Edit bookmarks”, pick the bookmark from the list in the dialog, update the name or URL below it, and save. To remove one, click the trash can icon in the list.

To add a bookmark, select “Add bookmark”, enter a name and URL, and save. Bookmarks to external sites open in a new window; internal console URLs open in the current window.

When viewing a report or browsing the list of reports, click the bookmark ribbon icon to add that report to your bookmarks. Click the ribbon again to open the editing dialog and remove it.

Custom widgets

Click “Create widget” in the widget library to build a custom widget of one of these types:

- **Single match count** shows the most recent match count for a given query.
- **Single match trend** charts the history of a given query.
- **Multi-query match count** shows the most recent match count for 2 or more queries, displayed like the *Organization overview* widget.
- **Multi-query trend line** charts the history of 2 or more queries.
- **Custom text widget** shows custom Markdown-formatted text, for extra context or information on the dashboard.
- **Goal progress overview** shows a summary and current progress toward completion for a single goal.

For a single-query widget, you choose a name, a query, and a color for the results. For a multi-query widget, you choose a name for the widget itself and a name and color for each query.

Any custom widget that displays as a trend chart can be exported to JPG, PNG, or SVG from the menu button at the top right of the widget on your dashboard.

After creation, manage custom widget templates in the widget library with the icons at the bottom of each widget's preview image. When you update a custom widget, runZero asks whether to update the widget on the dashboard too. When you delete one, it asks whether to remove it from the dashboard too.

Drill down

Except in timeline-based charts, you can view any widget's distinct data points in detail by clicking the relevant table row links or buttons, or individual segments of donut, bar, or column charts.

Printing and exporting

Chart-based widgets export to SVG, PNG, and JPEG, and table-based widgets export to CSV. Use the menu button at the top of the widget and pick the format.

Print the entire dashboard and all of its widgets with the printer icon at the top of the dashboard. For best results, use a Chromium-based browser or Firefox with the 'Print backgrounds' option enabled.

Recalculate metrics

Dashboard widgets, query results, and saved-query counts come from metrics that runZero recomputes after every scan, integration sync, and scheduled metrics task. If you just imported data, edited tags (including `geo*` location tags or manual `category` overrides), or changed asset criticality and want the dashboard, World Map, and saved-query counts to reflect it right away, click the dashboard menu button and select **Recalculate metrics**. runZero queues a one-shot metrics task and refreshes the dashboard when it completes.

Themes

The runZero console ships with three themes: **Classic**, **Light**, and **Dark**. Choose yours on the **Account > Preferences** page. Theme selection is per-user and applies to every dashboard, report, inventory view, and the Network and World Maps.

Display mode

Theater (or kiosk) mode hides the navigation menus and shows only the dashboard content. To enter it, click the menu button at the top of the dashboard and select "Enter theater mode".

In theater mode, the *Fullscreen* button at the top of the page fills the screen with the dashboard. Press `Escape` or click *Exit fullscreen* to return to theater mode.

To leave theater mode, click the X icon.

AI Threads

The AI section of the console lets you ask questions about your inventory, build reports, export data, draw charts and diagrams, and write custom integration scripts in a conversation. It lives at AI threads in the console navigation and has three tabs: **Threads** (your conversations), **Artifacts** (the saved output), and **Reporting** (guided report templates).

An **AI thread** is a private conversation between one user and the AI: no other user can see your threads, ever. What you choose to share is the artifact (the report, table, script, or other output a thread produces), never the conversation itself.

Requirements

- The account must have the AI entitlement, and AI must be enabled in the account (or organization) AI configuration. When AI is turned off, the composer is locked and the page says so, and superusers get a link to the AI configuration.
- An organization must be selected in the organization picker. A thread records the organization scope that was active when it was created, and the AI's searches never reach beyond that scope. The My Organizations option works with AI threads, but anyone you share an artifact with must have access to the same organizations.
- Runs are also subject to the account's AI token budget. When a daily budget or token cap is exhausted, new runs are refused until it resets.

Threads

Starting a thread

A thread starts from any of these places:

- the **New thread** composer on the threads page;
- the **New AI thread** button in the top bar;
- the command palette (**Cmd/Ctrl+K**), which can start a thread carrying the context of the page you are on;
- **Ask AI** handoffs from inventory pages (asset details, vulnerability and software groups, and similar views), which start the thread scoped to that object. A **Context** chip on the composer shows the scope; remove it before the first question to ask about the whole organization;
- the **Reporting** tab, where picking a guided report and answering its scoping questions starts a thread that generates the report;
- generating a custom integration script, where each authoring run appears as a live turn on its own thread.

In the composer, **Enter** sends and **Shift+Enter** inserts a newline. A new thread is titled from your prompt, and after the first answer the AI replaces that with a short generated title. Renaming a thread yourself stops the automatic titling.

The thread list

Threads are grouped by recency (Today, Yesterday, Earlier this week, Earlier this month, Older) and sorted by last activity. Each row shows a running indicator when a turn is in flight, an icon for the thread's newest artifact, the relative time of the last activity, when the thread expires, and the organizations it spans. The search box matches thread titles, prompts, and responses; press `/` to focus it.

The list shows the threads whose organization scope overlaps your current organization selection. When you have threads outside that scope, a footer note says so and offers to switch to **My Organizations**.

Reading a thread

Every turn shows your prompt, a **work log** of the steps the AI took (searches it ran, data it read), and a summary line with the elapsed time, token usage, tool calls, and the artifacts it produced. The **Normal** view folds the work log away; **Verbose** expands every step. The answer itself is saved as an artifact and shown as a linked card, and the newest one opens automatically in the artifact panel beside the conversation.

Background execution

A run is independent of your browser: closing the tab, navigating away, or losing the network does not stop it. Reopening the thread re-attaches to the run and replays everything that streamed while you were away, and the command palette lists running AI jobs so you can jump back to them. Only **Stop** ends a run early, and it returns your prompt to the composer. Otherwise a chat turn runs up to 5 minutes and a report turn up to 10.

Renaming and deleting

Rename a thread with the pencil beside its title (titles are capped at 120 characters). Deleting a thread permanently removes the conversation and every **unshared** artifact it produced, along with their attached files; shared artifacts are kept. A thread with a running turn cannot be deleted until the run finishes or is stopped. You cannot share a thread; share its artifacts instead.

Organization scope

Only a thread's owner can continue it, and only with an organization selection that still contains every organization the thread was created against. Switching to a different organization, or losing access to one of the thread's organizations, blocks new turns until the selection (or access) is restored; the existing conversation stays readable.

Tools

Every turn works through a set of read-only tools scoped to the thread's organizations, and the work log records which ones were called. The same tools are exposed over MCP, so an external MCP client and an in-product AI thread can ask the same questions of the same data.

Inventory search. `query_assets` and `query_asset_counts` for assets, `query_services`, `query_certificates`, and `query_tasks` for their own inventories, and `query_inventory` for everything else: software, software groups, vulnerabilities, vulnerability groups, wireless, findings, events, directory users and groups, tasks, issues, goals, credentials, saved queries, and sites. `get_attributes` pulls selected host attributes for a batch of assets, and `list_organizations` lists what the thread can reach.

Writing queries. Answers use real search syntax. `get_field_definitions` returns the searchable fields for an entity or integration source with descriptions and example values; `get_query_syntax` returns the operator reference, worked examples, and that entity's fields; and `translate_query` turns a plain-language phrase into runZero syntax. Asking for "cisco devices with ssh exposed" yields `(os:cisco or hw:cisco) and protocol:ssh`. The AI tries to cite data using live queries into the inventory search.

Detection coverage. `search_vuln_coverage` and `search_scan_checks` search the vulnerability match rules and active scan checks runZero ships, so "can runZero detect this?" is answered from platform coverage rather than from your findings. `list_rapid_response` lists the Rapid Response feed with the query that finds affected assets.

Organization analytics. A set of snapshot tools covering the whole organization at once: overview distributions, subnets, outliers, exposed assets, datastores, security-control signals, critical systems, unmanaged assets, end-of-life assets, Explorers, recent tasks, metrics, and goals.

A turn spends at most 8 tool calls on an ordinary question. Asking for a report, a deep investigation, a comparison, a chart, or a trend raises the budget to 20. Guided reports are not capped this way; the 10-minute turn limit bounds them instead.

Artifacts

Everything a thread produces is saved as an **artifact**: the answer document itself, plus any tables, charts, diagrams, or scripts it generated along the way. Artifacts are the durable, shareable output of the AI. They start private and expire with their parent thread unless shared. Once shared, they stay active indefinitely and can be deleted manually as needed.

The **Artifacts** tab lists your private artifacts and any artifacts team members have shared to their organizations. Search them by title, kind, and author, or filter with **Created by me** and **Created by other people**. Selecting one shows the content and its provenance: who created it, when, from which organizations, with which model, its size, and when it expires. Threads and artifacts can both span multiple organizations, and every viewer of an artifact needs overlapping access to every organization whose data it includes.

Artifact kinds

Kind	Notes
Markdown response	The primary answer document of a chat turn.
Report	The output of a guided report, named for the report.
CSV / TSV / JSON	Data exports, rendered as read-only source and downloadable as files.
SVG graphic	Rendered graphics, embedded inline where the answer references them.
Mermaid diagram	Diagram source, rendered in place and downloadable as a `.mmd` file.
HTML / HTML application	Self-contained documents, rendered in a hardened sandbox: no scripts from the network, no outbound requests.
Graphviz diagram	DOT source, shown read-only and downloadable as a `.dot` file.
Shell script / PowerShell / Starlark script	Code, rendered read-only with syntax highlighting.

Large inline tables and code blocks inside a report are carved into **attached files** (downloadable resources listed with the artifact), so a big export does not bloat the document. A single turn can produce up to 100 artifacts, each up to 512 KiB.

Charts and diagrams

When an answer needs a chart (a distribution, a trend, a flow, a path, a timeline, or a comparison), the AI draws it with Mermaid. Mermaid charts are plain text and cover bar and line charts, pie, flowchart, sequence, state, class, entity-relationship, timeline, journey, gantt, quadrant, sankey, mindmap, kanban, radar, treemap, venn, block, packet, architecture, C4 context, requirement, git graph, and fishbone. Standalone SVGs are reserved for the rare graphic no Mermaid form fits.

A diagram that stands on its own (one you asked for, or one worth sharing separately) is saved as its own **Mermaid diagram** artifact. It still renders in place in the answer, and you can download it separately as a `.mmd` file. Diagrams also render in the Markdown artifact's HTML and PDF exports. Inside the body of an HTML report, the diagram is a link to the artifact rather than a picture, because the report sandbox runs no scripts; graphics that must appear inside an HTML report body are drawn as SVG instead.

Viewing and exporting

Every artifact has a **Full screen** view that fills the window; press **Esc** to return. HTML reports carry their own toolbar with **Print / PDF**, an **HTML** download, and **Open** (full screen in a new

tab). Prose artifacts export to Markdown, HTML, and PDF; file artifacts (data and scripts) download as their original file. The creator can rename any artifact (up to 160 characters).

Sharing and privacy

Every artifact starts **private** to its creator. Sharing one makes it visible to every user who can read **all** of the organizations it was created from; **Copy link** (available once shared) produces a link that opens it in the Artifacts tab. Unsharing returns it to private.

The creator can always delete their own artifact. A superuser, or an administrator of every organization in its scope, can also delete a shared artifact. Deletion is permanent, removes attached files, and removes the artifact for everyone.

Use in custom integration

A Starlark script artifact, typically the product of the AI script generator, has a **Use in custom integration** action that loads the script back into the integration it was written for, or into a new one. This requires the custom integration entitlements in addition to AI.

Reporting

The **Reporting** tab is a gallery of guided report templates, such as End-of-Life OS Analysis, Rogue & Unmanaged Assets, Critical Systems & Blast Radius, and Internet Exposure, each with optional scoping questions. **Start report thread** opens a new thread that generates the report, so the result arrives as an artifact you can review before opening or sharing it.

Limits

- At most 4 AI runs per user and 12 per account may run at once, and one per thread.
- A single request (a prompt and its context) is capped at 64 KiB.
- A chat turn runs up to 5 minutes; a guided report turn up to 10 minutes.
- A chat turn makes at most 8 tool calls, or 20 when the question asks for a report, deep investigation, comparison, chart, or trend.
- One turn can produce at most 100 artifacts, 512 KiB each and 2 MiB in total.

Retention

AI data has fixed retention windows, separate from the organization-level data retention settings:

- **Threads are removed 180 days after their last use.** Every update pushes the clock back 180 days, so an active thread never expires and an abandoned one is removed half a year after it was abandoned. Renaming a thread does not reset the clock. Thread rows show the countdown: `expires in N days` within 30 days, a date beyond that.
- **Unshared artifacts expire with their thread.** When a thread expires or is deleted, the unshared artifacts it produced are removed with it, attached files included.

- **Shared artifacts are kept indefinitely** while shared. Unsharing one restores the thread-derived expiry; deleting it removes it immediately.
- **Run records are temporary.** The job record and streamed event log behind a run are kept about 15 minutes while running and 24 hours after the run ends; the durable results are the thread and its artifacts.

Cleanup runs daily, and expired items disappear from the console at their expiry time. These windows are not configurable; you can delete individual threads and artifacts at any time.

Using the inventory

The inventory page is the heart of runZero Network Discovery and the key to understanding what is on your network. The inventory shows all assets in the Organization, and you can sort, filter, and export it to get specific views of your environment.

Understanding assets

In runZero, an asset is a unique network entity. An asset may have multiple IP addresses and MAC addresses, and those addresses may change as the environment changes. runZero tracks assets with several heuristics, including MAC address, IP address, hostnames, and fingerprint results for the operating system and running services.

In most cases, runZero can follow assets accurately over time in environments using DHCP, even across remote subnets. For external networks, a scan started with fully qualified hostnames consolidates assets by hostname, so cloud-based external systems with dynamic IP addresses stay tracked consistently.

Within an organization, assets are isolated by site, and each site can have address space that overlaps with other sites. When they overlap, sorting the Inventory view by the site column can help, as can filtering the Inventory by a specific site name.

The search field filters the inventory by the criteria you enter. The search query syntax documentation covers the details.

Beyond viewing assets, you can export data from the Inventory page, select assets, and set the comments field. The *Rescan* action rescans specific systems from the inventory, and *Remove Assets* and *Purge Assets* permanently remove data from the inventory view.

You can also share an inventory query as a read-only snapshot with people outside runZero through external reporting.

The *Reports* button is a shortcut to the main reports on the runZero reports page.

Supported asset types

IT

- ADSL Modem
- ADSL Router
- ATM
- ATM DSL Unit
- BMC
- Broadband Router
- Cable Modem
- Desktop
- Docking Station

- DOCSIS Cable Modem
- DSL Modem
- DSLAM
- DSU/CSU
- Ethernet Adapter
- Fax Server
- Firewall
- Frame Relay
- Hypervisor
- IDS
- IPS
- JTAG Adapter
- KVM
- Laptop
- Lights Out Management
- Load Balancer
- Mainframe
- Management Console
- Management Processor
- Media Gateway
- Media Server
- Multiplexer
- NAC
- NAS
- Network Appliance
- Network Management
- Network Scanner
- Onboard Administrator
- Optical Line Terminal
- PABX
- Paging Terminal
- PBX
- Power Device
- Powerline
- Print Server
- Printer
- Remote Terminal
- Router
- SD-WAN Appliance
- Security Appliance
- Security Gateway
- SIP Gateway
- SSL-VPN
- Storage
- Storage Appliance
- Tablet
- Tape Library
- Terminal Server
- Thin Client
- USB Server

- VoIP Gateway
- VoIP Router
- VoIP Server
- VoIP Switch
- VPN
- WAF
- WAN Accelerator
- WAP
- Web Proxy
- Wireless Controller
- Wireless Dock
- Wireless Ethernet Bridge
- Wireless LAN Controller
- WLAN Repeater

OT

- AC Drive Device
- BACnet Router
- Building Automation
- Data Logger
- DCS
- Device Server
- Engineering Workstation
- Ethernet IO Module
- Frequency Converter
- General Purpose Analog I/O
- General Purpose Discrete I/O
- High-Speed Counter
- Historian
- HMI
- HMI Controller
- IED
- Industrial Control
- Industrial Robot
- Motion Control
- Motion Controller
- Motor Drive
- PLC
- Position Controller
- Power Relay
- Process Controller
- Protocol Gateway
- PXI Switch Module
- Redundancy Module
- Relay Controller
- Remote Access Server
- RTU
- Safety Controller
- Safety PLC

- SCADA
- Servo Controller
- Servo Drive
- SIS
- Soft Starter
- Specialty I/O
- Traffic Control
- Variable Frequency Drive
- VFD

IoT

- 3D Printer
- Access Control
- Access Controller
- Air Quality Monitor
- Alarm Panel
- Amazon Device
- Audio Encoder
- Audio/Video Switch
- Automobile
- AV Appliance
- AV Controller
- AV Receiver
- Barcode Scanner
- Check Scanner
- Clock
- Copier
- Dishwasher
- Display Controller
- DVR
- Environment Control
- Exercise Equipment
- Game Console
- Handheld Blood Analyzer
- Handheld Scanner
- Home Appliance
- IoT
- IP Camera
- IP Phone
- IPTV
- IR Network Adapter
- Irrigation Control
- Lab Equipment
- Laboratory Instrument
- Light Bulb
- Light Switch
- Mattress
- Media Player
- Media Receiver

- Multifunction Device
- Musical Instrument
- Network Audio
- NVR
- PDU
- Point of Sale
- Power Meter
- Presenter
- Projector
- Radio
- Room Controller
- Scale
- Security System
- Sensor
- Set-Top Box
- SIP Device
- Smart TV
- Smart Watch
- Thermal Imager
- UPS
- Vacuum Cleaner
- Video Conferencing
- Video Decoder
- Video Encoder
- Voice Appliance
- Voice Assistant
- VoIP Intercom
- VR Headset
- Water Meter
- Whiteboard
- Wireless Presenter

Loading assets

The **Scan** and **Import** buttons load data into the inventory. runZero analyzes and merges the results and updates asset information as needed.

The **Scan** button has two options: Standard Scan and Full RFC 1918 Discovery. The second option sets up a fast scan of all private range IP addresses. Afterwards, the coverage reports show whether assets turned up in unexpected private address ranges.

The **Import** button has two options. With imported runZero scan data, you can scan networks that have no connectivity to the internet and still view the results in the runZero console. Importing is also useful for reprocessing old scan data, so the site compare feature can show how assets have changed over time.

Bulk asset update

With bulk asset update, you modify assets by exporting a CSV with the **Export** button, editing the data in a spreadsheet program or text editor, and importing the result back into runZero with the **Import** button. The import updates existing assets in the organization that have a matching `id` value.

Bulk asset update can change these fields:

- Type
- Operating system
- OS version
- Hardware
- Comments
- Tags
- Owner
- Names
- Domains

The `type`, `os`, `os_version`, and `hardware` fields accept only a single value. The `comments`, `tags`, `owner`, `names`, and `domains` fields each accept multiple values, written as a space-delimited list of `field=value` pairs. The `tags` field also accepts a plain space-delimited list of values without `tag=`.

Only changes to the `tags`, `comments`, and `owners` fields survive later scans. The latest scan data overwrites changes to the other supported fields.

Removing tags from assets

To remove a tag from one or more assets, select the assets in the inventory with the **Select** button in the toolbar, then click **Modify** and choose **Set asset tags**. In the tag editor, prefix the tag you want to remove with a minus sign (`-`). runZero removes the matching tag and leaves the asset's other tags in place.

```
-tag_name
-tag_name=value
```

For example, enter `-env` to remove the `env` tag, including tags such as `env=prod`. To remove only a specific tag value, enter `-env=prod`; other `env` tags remain in place.

This pattern also works in bulk updates that modify multiple assets at once. If you edit the tags field in a bulk asset update CSV before using **Import**, the same `-tag` syntax removes tags from the selected assets.

To remove all tags from a single asset, open the asset details page, click **Manage** in the top-right corner, and choose **Unmerge asset**. runZero deletes the current asset record and reprocesses it, which drops stale tags and keeps the valid tags from the most recent tasks.

Connecting to other systems

The **Connect** button connects runZero to other systems. Which integrations you can connect depends on your license level, but they may include cloud and virtualization platforms, endpoint protection solutions, identity and access management tools, and vulnerability and risk platforms. You can also configure these inbound integrations as scan probes if required.

Viewing services

The *Services* submenu on the Inventory page switches the table from an asset-focused view to a service-focused view, with one row for each service runZero detected on each asset.

Like the main asset view, the services view has a full search interface. You can filter services by protocol, port, and many other criteria, using the runZero search language.

> Note
> runZero does not enumerate Windows services through the Service Control Manager because active scanning is unauthenticated. In some cases, you may still see limited Windows service details from unauthenticated DCE/RPC Endpoint Mapper (EPM) data, which runZero uses primarily to identify WiFi and WLAN service availability.

Viewing screenshots

If the runZero Explorer has access to Google Chrome, it attempts to take screenshots of the web pages it finds while scanning your network. You can disable this in the scan options when you set up the scan.

The *Screenshots* submenu shows the screenshots for all of your assets, and each one links through to the asset record for full details.

Viewing software

The *Software* submenu on the inventory page flips the table from an asset-focused view to a software-focused view, with one row for each piece of software that runZero or a supported integration detected on each asset.

Like the main asset view, the software view has a full search interface. You can filter software by vendor, product, and many other criteria, using the runZero search language.

Viewing vulnerabilities

The *Vulnerabilities* submenu on the inventory page flips the table from an asset-focused view to a vulnerability-focused view, with one row for each vulnerability that a supported integration detected on each asset.

Like the main asset view, the vulnerability view has a full search interface. You can filter vulnerabilities by CVSS score, name, CVE, and many other criteria, using the runZero search language.

Viewing certificates

The *Certificates* submenu opens the certificates inventory: every encryption certificate the runZero Explorer encountered while scanning, in a view you can search and sort like the others.

Viewing wireless networks

If the machine running the runZero Explorer has a working WiFi adapter and the right system tools installed, the Explorer attempts to scan for nearby wireless networks. The *Wireless* submenu shows the results.

The required tools are:

- Windows: `netsh.exe` (part of modern Windows releases)
- macOS: Airport Utility
- Linux: `iwlist`, often available via the `wireless-tools` package.

Viewing users and groups

The users and groups inventory pages hold data imported from directory services such as Active Directory. Third-party integrations listed under *Directory services* in the *Integrate* drop-down menu populate the user directory and group directory.

Understanding assets

runZero treats assets as unique network entities from the perspective of the system running the Explorer. An asset may have multiple IP addresses, MAC addresses, and hostnames, and it may move around the network as those attributes change. runZero tries to follow each asset by correlating new scan data with the existing inventory across multiple attributes.

An asset always belongs to a single site. If multiple sites cover the same system, runZero treats it as a different asset in each site and correlates it only against assets within that site. With this separation, you can scan the same network from multiple perspectives and compare the results in a single view within the organization.

After each scan, runZero updates all assets within the corresponding site. A system that doesn't match an existing asset becomes a new asset. If the scan does not find an asset in the site, runZero marks it offline. When an asset fails to correlate because its fingerprint changed substantially (for example, a new network adapter was installed and the firewall was enabled), runZero marks the previous asset offline and creates a new asset to track the new configuration. This can leave some duplicates within a site, but they are usually marked as offline, and you can safely ignore them or remove them from the inventory by hand.

Asset fields

You can search and filter on the fields below with the asset inventory keywords.

Primary addresses

runZero reports at least one, and often several, primary IP addresses for an asset. These addresses can span multiple network interfaces, but an address only appears as a primary address if runZero has scanned it, which means it falls within the scan scope of one or more runZero scans.

Secondary addresses

runZero may report one or more secondary addresses, based on network response probes. These are IP addresses detected on the asset that were outside the scan scope. You need secondary address detection to identify systems bridging networks that should be isolated.

Hostnames

runZero may report one or more hostnames. Names can come from the initial DNS lookup (when the scan scope includes hostnames), from DNS PTR lookups during the scan, and from names advertised in network probe responses.

Operating system (OS)

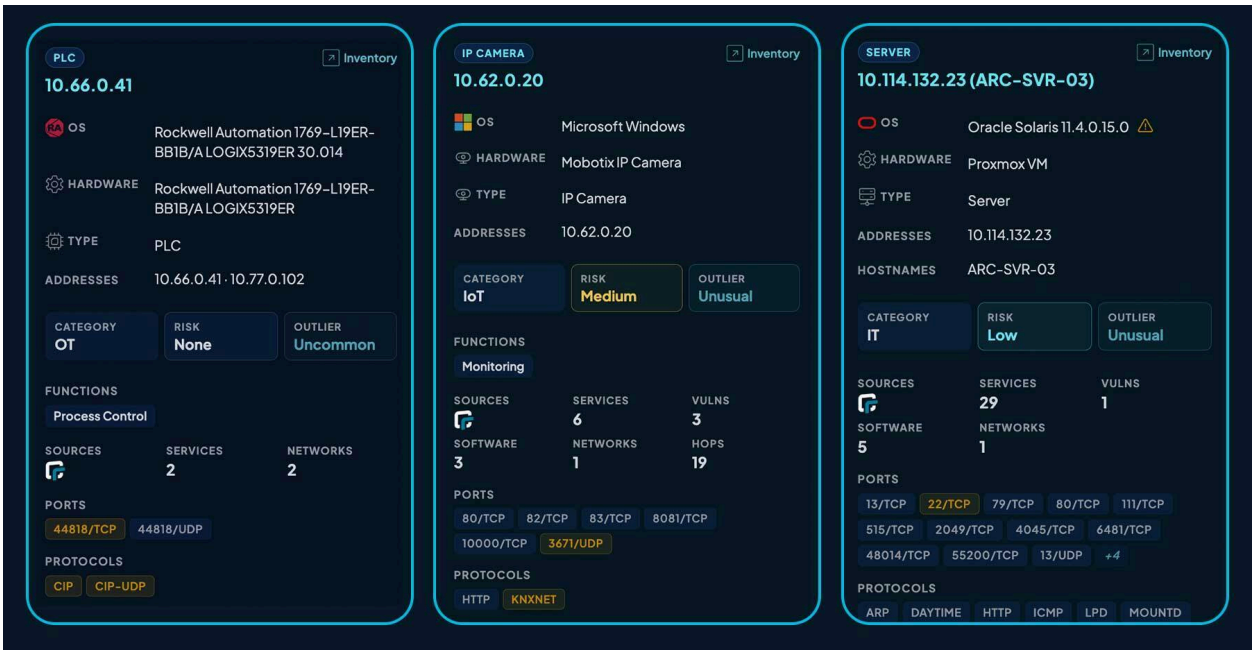
runZero attempts to fingerprint the operating system running on each asset and, failing that, to guess it. With limited information, this field may be empty.

Type

runZero attempts to determine the general device type by analyzing fingerprints and running services.

Category

runZero classifies every asset into a high-level asset category so you can separate traditional computing infrastructure from operational technology and connected devices at a glance. The category comes from the device type, vendor, and service protocols (for example, an asset speaking Modbus, CIP, or S7Comm is placed in `OT` automatically).



Value	Meaning
`IT`	Traditional information-technology assets: servers, workstations, laptops, network gear, storage, hypervisors.
`OT`	Operational-technology assets: PLCs, RTUs, IEDs, DCS, SCADA, HMIs, drives, protocol gateways, industrial robots.
`IoT`	Connected devices that are neither IT nor OT: IP cameras, building-automation sensors, printers, smart-TVs, medical devices, point-of-sale, access control.

Query with `category:OT`, `category:IT`, or `category:IoT`. To override the category, set the `category` asset attribute to one of these three values.

Functions

Beyond a category, OT and OT-adjacent assets carry one or more asset functions describing what the device does. Functions are multi-valued: a single device can carry several at once (for example, a substation IED is both `Process Control` and `Monitoring`).

Function	Typical assets
`Process Control`	PLCs, RTUs, IEDs, DCS, SCADA controllers, supervisory-control software.
`Safety`	Safety controllers, safety PLCs, Safety Instrumented Systems (SIS).
`Human Interface`	HMIs, operator panels, touchscreens, control-room displays.
`Engineering`	Engineering workstations and laptops used to program, configure, or maintain OT devices.
`Monitoring`	Sensors, meters, condition-monitoring units, tank monitors, environmental sensors used for observation only.
`Environmental`	Building-automation, HVAC, lighting, refrigeration, room and climate controllers.
`Communications`	Field-bus gateways, protocol translators, serial-to-IP adapters, modems used inside OT segments.
`Networking`	Switches, routers, firewalls, and data diodes that segment or interconnect OT zones.
`Remote Access`	Jump servers, VPN concentrators, and remote-access servers used to reach OT environments.
`Data Management`	Historians, data loggers, and other systems whose primary role is storing operational data.
`Power Management`	UPS units, PDUs, power meters, transfer switches, generators.
`Physical Security`	Access-control panels, door controllers, alarm panels, surveillance NVRs.
`Cyber Security`	OT-resident IDS/IPS, security gateways, and other dedicated cyber-security appliances.
`Management`	Management appliances and consoles used to administer OT devices and networks.

Query with `function:"Process Control"`, `function:Safety`, `has:functions`, and so on. To set function values by hand, edit the asset's `functions` attribute.

Hardware

runZero attempts to determine the physical (or virtual) hardware when enough information is present.

MAC addresses

runZero may be able to enumerate one or more MAC addresses from the asset. It pulls them from ARP when available, and also from several network services that can return MAC address information across routed segments.

Services

By default, runZero tries to detect approximately 100 TCP services plus several useful UDP services, on top of ARP and ICMP. The services field lists the most recently recorded services for the asset.

Round trip time (RTT)

runZero records how long certain probes take, which gives a rough sense of the latency between the Explorer and the asset.

Detected by

runZero records which probe identified an asset. For assets on remote subnets with firewalls in place, this field shows which service produced a response.

Alive status

runZero tracks whether an asset was found during the most recent scan that had its site in scope. An asset that was not found is marked as offline until a later scan detects it again.

First seen

runZero records the timestamp when it first identified an asset.

Last seen

runZero records the most recent timestamp when an asset responded to a probe during a scan.

Explorer

runZero tracks the most recent Explorer to detect an asset through active scanning or passive discovery. Third-party connections that run through an Explorer do not typically update this field, because the third party detected the asset, not the Explorer.

Outlier score

runZero computes an outlier score from 0 to 5 (inclusive) for every asset in your inventory. The score is a heuristic that aims to show how unusual the asset is compared to all of the others in the inventory.

To compute it, runZero examines the main properties of the asset and its services, works out which values are unusual (infrequent) across the organization, and counts how many unusual properties each asset has. The more unusual properties an asset has, the higher its outlier score.

Understanding findings

Findings condense vulnerabilities, misconfigurations, and best practices into one prioritized, curated, and aggregated list, so you can identify and remediate the most critical risk in your environment. runZero Findings are available from the *Findings* menu and from the Risk Management dashboard.

What are findings?

Findings surface the risks attackers are most likely to target, so security teams can focus remediation on risks with real operational impact.

Findings group similar vulnerabilities and misconfigurations to show the whole risk they represent. Each finding contains a description of the risk, remediation steps, risk rankings, and a list of every asset and entity the risk affects.

Finding categories

runZero places each finding into a category based on the type of risk it represents. Categories include:

- Internet Exposure: assets and services that may be unintentionally exposed to the internet
- Certificates: expired or soon-to-expire certificates, and widely shared private keys
- Vulnerability: actively exploited vulnerabilities, or critical vulnerabilities that runZero believes are critical to address
- End-of-Life: operating systems, hardware, and applications that have reached End-of-Life (EOL) or End-of-Service (EOS) and are no longer supported by the vendor
- Open Access: network services such as unauthenticated databases and sensitive applications that are accessible without authentication
- Compliance: assets and services that violate security best practices
- Best Practice: general best practices covering insecure authentication, service misconfiguration, and obsolete protocols
- Rapid Response: emerging and novel threats covered in detail by runZero Rapid Response blog posts

Findings list

The findings list is in the main navigation menu, directly below the Inventory item. You can search, sort, and export findings like other views in the console.

Finding details

Click a finding's name to open its details: an overview of the risks discovered, any related external resources, and remediation information to help you address the risk. Below the

finding summary is a list of every asset instance the finding applies to, so you can prioritize remediation quickly.

How are findings different from vulnerabilities?

A vulnerability is a specific security issue, usually tied closely to a specific CVE or security advisory. The runZero Explorer finds vulnerabilities, or you import them through one of the supported integrations. A single asset can have hundreds of thousands of vulnerabilities, and your whole environment even more.

A finding is a curated, aggregated, and prioritized list of the risks attackers are most likely to target. A single finding may group several similar vulnerabilities. Findings are not always tied to a CVE, and they may include misconfigurations, best practices, and other security issues that may not make sense as a vulnerability.

Asset risk and criticality

runZero helps you assign and evaluate risk and criticality levels for the assets in your inventory. With those levels set, you can spot the assets in your organization with the highest risk or criticality at a glance and prioritize risk mitigation or vulnerability remediation around them.

Defining risk and criticality

runZero assigns the **risk** level automatically, inferring it from the risk of the vulnerabilities or risky configurations on the asset; it defaults to `none`. A vulnerability's risk level may come from the vulnerability management solution its records were ingested from, or from the risk level assigned to a query vulnerability. You can override the risk level, and the override stays until the asset or vulnerability is deleted. For vulnerabilities ingested from integrations, deletion may happen when the source no longer reports the vulnerability on that asset.

You assign the **criticality** level manually; it defaults to `unset`. Criticality denotes how important an asset is to your organization. For example, you might give business-critical systems such as database and web servers a `critical` level and normal end-user systems a `medium` level.

Assigning asset risk and criticality

You can assign both asset risk and criticality from the asset inventory. Alert rules can also assign criticality.

Superusers, administrators, and users can add or modify asset risk and criticality levels, reset a risk assignment, and remove a criticality assignment from assets.

Risk and criticality in the asset inventory

To set risk and criticality through the asset inventory:

1. Select the assets you want to update, applying a query filter if needed.
2. Click *Modify asset risk* or *Modify asset criticality* to open the popup.
3. Select the level of risk or criticality to apply to the selected assets.
4. Click *Override risk* or *Set criticality* to apply your changes.

Applying criticality with rules

To apply asset criticality automatically after each scan, create an alert rule: go to **Alerts** > **Rules** and click *Create rule*.

1. Select the inventory query to use, such as the `asset-query-results` rule type, then click *Configure rule*.
2. Configure any settings you want.
3. Set the **Action** to *Modify asset*.

4. Select an option from the *Asset criticality* menu.
5. Save the rule.

When a scan completes, the rule adds the specified criticality level to every asset that matches it.

Asset risk report

The **Asset risk report** shows the risk and criticality levels across your asset inventory. To run it, go to **Reports** > **Asset risk report**, click the **Asset risk report** button, and configure these fields:

1. **Sites**: Select a site of assets to include in the report, or leave the default *All Sites*.
2. **Minimum risk**: Choose the lowest asset risk level to include.
3. **Minimum criticality**: Choose the lowest asset criticality to include.
4. **Top vulnerabilities per asset**: Set an integer between 0 and 20. With a value from 1 to 20, the report lists up to that many of the top vulnerabilities detected on each asset. runZero picks them by sorting each asset's vulnerabilities by risk rank, then risk score, then severity rank, then severity score.
5. Click **Create report** to generate the results.

The report is grouped by asset criticality level and then sorted by risk level. You can export the results as JSON Lines (.jsonl), a JSON document (.json), or CSV (.csv).

Certificates

runZero collects TLS certificates automatically during active scans and passive network monitoring. It stores them in the Certificates Inventory, which you can open from the *Inventory* menu, *Certificates* sub-menu. Most TLS certificates come from web servers, but runZero also collects them from SMTP, RDP, and other TLS-aware services.

Every certificate has a validity period. In the inventory view, the *Valid from* and *Valid until* values are color coded to show whether the certificate is not yet valid, has expired, or expires soon.

A certificate has an *issuer*, the authority that signed it. When the issuer is the same as the subject, runZero marks the certificate as self-signed.

Certificates rely on two kinds of algorithms:

- The public key algorithm, used for decryption and verification.
- The signature algorithm, used to produce a content hash that is signed using the public key algorithm.

The security of a public key algorithm depends on its key size, shown in bits, and how large a secure key must be varies by algorithm. An RSA key should be 2048 bits or larger to count as secure, while every ed25519 key is 256 bits, and that is considered secure.

Each certificate in the inventory view has a button to hide it from the inventory, which can be useful for embedded devices whose certificates you can't change and don't want to see. To see which certificates you've hidden, search for `hidden:true`. The certificate inventory keywords cover the other ways to search and filter certificates.

TLS certificates

TLS certificates have three sets of names for their hosts:

- The Subject, an X.509 distinguished name (DN) made up of one or more attributes, each with a value.
- The Subject Alternative Name lists, which hold the values web browsers check when connecting to a server. A certificate stores SANs in separate fields by type (DNS, IP, email, or URI).
- The Common Name, which web browsers once used but now ignore. It is typically in X.509 format.

The Names runZero shows for a TLS certificate combine the four kinds of SANs plus the Subject DN.

Each TLS certificate has a subject key ID that identifies the certificate's subject and an authority key ID that identifies the signing authority. These IDs link certificates into a chain: the authority key ID of each certificate is the subject key ID of the next certificate in the chain.

The issuer of a TLS certificate is included in the certificate as a DN.

TLS certificates have a flag that says whether they are intended for use as a certificate authority or (CA). The flag doesn't mean the certificate *is* a CA or that it is trusted, only that the flag is set. CA certificates are usually self-signed.

The values runZero shows for TLS certificate algorithms mostly come from RFC 5280 and should correspond closely to the values OpenSSL uses.

The self-signed value shows whether a certificate appears to be self-signed, based on its issuer, subject, and (if present) their IDs. A certificate flagged as apparently self-signed may still fail signature validation in some software, because different software applies stricter or looser checks.

The certificate details page has a button to download the certificate in PEM format. You can also download the entire chain of certificates known to runZero as a single PEM file.

Geolocation

runZero resolves a geographic location for every asset it has enough information about, and those locations drive the World Map, inventory search, and exports. This page covers the sources runZero uses for asset geolocation, how it interprets your location tags, and the order in which it applies the sources.

Where locations come from

runZero builds asset locations from five independent sources during each analysis pass. It records every location with its provenance, so you can trace each pin on the World Map back to the attribute, integration, or tag that produced it.

Source	Description
`attribute-coord`	Paired numeric latitude/longitude attributes from integrations or imports.
`attribute-ip`	Egress / public IP addresses resolved through MaxMind GeoLite2.
`attribute-text`	Discrete country / city attributes, plus free-form `snmp.sysLocation`-style strings.
`cloud-region`	AWS, Azure, GCP, OCI, IBM Cloud, and DigitalOcean region codes resolved to a region centroid.
`geolite-isp`	GeoIP lookup of the asset's primary scanned IP through MaxMind GeoLite2.
`manual`	User-supplied `geo*` asset tags.

An asset can carry locations from more than one source at a time. For example, a cloud workload may have a `cloud-region` location from its AWS region, an `attribute-ip` location from its public NAT address, and a `manual` location set by a tag. All three appear as separate pins on the World Map, each labelled with its source.

When multiple sources agree on the same coordinate, runZero de-duplicates the entry.

GeoIP lookups against MaxMind

runZero ships with the MaxMind GeoLite2 City and ASN databases and uses them to resolve any public IP address it observes for an asset into a latitude, longitude, country, subdivision (state/province), city, postal code, timezone, continent, ASN number, and ASN organization name.

GeoIP lookups reject private RFC1918 and RFC6598 addresses, so a private IP never produces a location.

Primary and hinted external IPs

When runZero scans an asset, it records the public IP it observed for that asset. It looks up the primary scanned address directly against GeoLite2, which produces a location with source `geolite-isp`.

When passive sources or integrations report additional public addresses for the same asset (for example, a secondary NAT address or an alternate egress on a multi-homed device), runZero looks those up too. It records the resulting locations alongside the primary, so an asset visible at two different egress points appears in both places on the World Map.

Egress and public-IP attributes from integrations

Many integrations report the public, external, or NAT IP address an asset was observed from. runZero ingests these and resolves each one through GeoLite2, which produces both a city-level location and an ASN annotation for the pin.

The egress IP pipeline reads these attributes:

Attribute	Set by
`externalIP`	CrowdStrike, SentinelOne
`publicIP`	Azure (VM/database/load balancer), Microsoft Defender for Endpoint, AWS
`publicIPs`	AWS
`natIP`	Google Cloud Platform
`wanIPAddresses`	Google Workspace
`discover.externalIP`	CrowdStrike

runZero also writes the full set of unique public IPs seen across these attributes back to the asset as an `egressIPs` attribute, so you can query the underlying values directly.

Cloud region codes

When an integration provides a cloud region code (for example, AWS `us-east-1`, Azure `eastus`, or GCP `europe-west4`), runZero maps the code to the published centroid of that region. This gives the asset a country, subdivision (where applicable), city, and continent without a GeoIP lookup.

The cloud region pipeline reads from attributes including `region`, `cloud.region`, `aws.region`, `azure.region`, `azure.location`, `gcp.region`, `gcp.zone`, `oci.region`, `availabilityZone`, `availability_zone`, `zone`, `placement.region`, and `placement.availabilityZone`. runZero ignores unknown values (for example, a generic `region=Northeast` tag).

Discrete country and city attributes

Some integrations report location as separate `country` (or `countryCode`) and `city` fields. runZero pairs them and resolves the combination to the matching GeoNames city centroid. If only a city is present, runZero uses the highest-population match worldwide.

The country fields runZero reads are: `country`, `countryCode`, `country_code`, `host.country`, `host.countryCode`, `host.location.country`, `host.location.countryCode`, `discover.country`, `discover.countryCode`, `custom.country`, and `site.country`.

The city fields runZero reads are: `city`, `host.city`, `host.location.city`, `discover.city`, `custom.city`, and `site.city`.

Free-form location strings

runZero parses free-form location strings from SNMP and integrations on a best-effort basis. The parser strips address noise (house numbers, suite/floor markers, PO boxes, postal codes, and street suffixes like `St`, `Ave`, and `Blvd`) and matches the remaining trailing tokens against the bundled GeoNames tables.

The free-form attributes runZero reads are: `location`, `host.location`, `site.location`, `custom.location`, `location.address`, `snmp.sysLocation`, and `shodan.snmp.location`.

The best-effort approach is deliberate: SNMP `sysLocation` has no standardized format, and many devices report a campus or rack identifier rather than an address. When the parser cannot anchor the string to a known city, region, or country, it records no location.

Manually setting an asset location

To pin an asset to a specific geographic location, set one of the `geo*` tags below. Each tag has a fixed shape, and its value tells runZero where the asset is. The location appears on the World Map, in exports, and in queries the next time runZero recalculates metrics for the asset.



Manual `geo*` tags do not take effect immediately. runZero picks them up the next time it recalculates metrics for the asset, which happens automatically during normal task processing (scans, integration syncs, and scheduled metrics tasks). To apply the change right away, go to **Dashboard** > **Recalculate metrics**.

The short version: set one tag, and use `/` to separate fields.

```
geoCity = Austin/TX/US
geoCity = Springfield/IL/US
geoLatLng = 30.2672/-97.7431
```

Available tags

Tag	Shape	Example
`geoLatLng`	`lat/lng`	`geoLatLng = 30.2672/-97.7431`
`geoContinent`	continent name or two-letter alias (`EU`, `NA`, ...)	`geoContinent = Europe`
`geoCountry`	country (see Country values)	`geoCountry = US`
`geoRegionCountry`	`region/country`	`geoRegionCountry = TX/US`
`geoRegion`	`region/country` or a single token (state, country, or continent)	`geoRegion = Texas`
`geoState`	alias of `geoRegion`	`geoState = TX/US`
`geoProvince`	alias of `geoRegion`	`geoProvince = Ontario`
`geoCity`	`city/country` or `city/region/country`	`geoCity = Austin/US`

You only need to set one tag per asset. If you set more than one, runZero uses whichever tags match their shape. runZero silently ignores tags whose values do not match their expected shape; it will not guess.

When to use which tag

- Use `geoLatLng` if you have exact coordinates from a CMDB, building floor plan, or GPS source.
- Use `geoContinent` when you only know the continent.
- Use `geoCountry` when you only know the country.
- Use `geoRegionCountry` when you know the country and the state/province/region but not the city.
- Use `geoRegion` (or its synonyms `geoState` / `geoProvince`) when you want runZero to accept any plausible region-ish value: a `region/country` pair, a unique state name (`Texas`, `Ontario`, `Bavaria`), a country code, or a continent.
- Use `geoCity` in every other case. The two-field form is enough for major cities; use the three-field form to disambiguate names like `Springfield`, `Portland`, or `Cambridge`.

Field separator

The separator between fields is always a forward slash (`/`). The characters `-`, `.`, `,`, and _ are not field separators, because they appear inside real city and region names: `Winston-Salem`, `St. Louis`, `Stratford-upon-Avon`, `St-Lawrence`.

```
geoCity = Austin/US                ✓  city/country
geoCity = Austin TX US             ✗  no separators
geoCity = Austin-TX-US             ✗  hyphen is part of names, not a separator
geoCity = Austin,TX,US             ✗  comma is part of names, not a separator
```

If you set a tag through the UI or API and the value contains spaces, runZero stores those spaces as underscores. The parser treats _ and a space the same inside a field, so `geoCity = San Francisco/CA/US` and `geoCity = San_Francisco/CA/US` parse identically.

Country values

Any of these forms work for the country field:

Form	Examples
ISO 3166-1 alpha-2	`US`, `DE`, `GB`, `JP`
ISO 3166-1 alpha-3	`USA`, `DEU`, `GBR`, `JPN`
English name	`United States`, `Germany`, `Japan`
Common alias	`UK` and `Britain` (both normalize to `GB`)

Country values are case-insensitive. When runZero recognizes an alias, it normalizes it to the ISO alpha-2 code shown on the map.

Region values

Region values are state, province, or admin region names. runZero recognizes both short codes and full names for the countries below; for other countries, runZero accepts the GeoNames admin1 code directly (often a numeric value like `02` or `16`).

Country	Examples of accepted region values
United States	`TX`, `Texas`, `CA`, `California`
Canada	`ON`, `Ontario`, `BC`, `British Columbia`, `Québec`
Australia	`NSW`, `New South Wales`, `VIC`, `Victoria`
United Kingdom	`England`, `Scotland`, `Wales`, `Northern Ireland`
Ireland	`Connacht`, `Leinster`, `Munster`, `Ulster`
New Zealand	`Auckland`, `Canterbury`, `Wellington`

If runZero does not recognize the region, the asset still resolves at the country level. The region tag adds information but never blocks resolution.

City matching and GeoNames

runZero matches cities and regions against the bundled GeoNames data. The matcher is fuzzy: `St. Louis`, `St Louis`, `St-Louis`, `Saint Louis`, and `saint_louis` all resolve to the same record. The map and exports show the canonical GeoNames spelling, so your tag values do not need perfect formatting to look right.

When a city name is ambiguous (`Springfield`, `Cambridge`, `Portland`), use the three-field form to disambiguate:

```
geoCity = Springfield/IL/US      ✓ disambiguated
geoCity = Springfield/MO/US      ✓ different city
geoCity = Springfield/US         ⚠ resolves to whichever Springfield is largest
```

Examples

Pinning a corporate office:

```
geoCity = Austin/TX/US
```

A device with exact coordinates:

```
geoLatLng = 30.2672/-97.7431
```

A subnet known to be in Germany, city unknown:

```
geoCountry = DE
```

A device in a known region, city unknown:

```
geoRegionCountry = Bayern/DE
```

What happens if the value is wrong?

Each tag has a fixed shape and a strict matcher. When the value does not fit the shape, or names a place that runZero's bundled GeoNames data does not recognize, the tag produces no manual location and no manual pin appears on the map for that asset.

This is intentional. To debug:

1. Confirm the field separator is `/`, not `-`, `.`, or `,`.
2. Confirm the country value resolves (try the ISO alpha-2 code).
3. For `geoCity`, try the three-field form to disambiguate the city.
4. As a last resort, use `geoLatLng` with explicit coordinates.

Searching by location in the asset inventory

The asset inventory supports a `geo` keyword (with synonyms `geo_within` and `geo_radius`) that matches assets whose geolocation falls inside a circular area:

```
geo:<lat>,<lon>[,<radius>]
```

- `<lat>` and `<lon>` are decimal degrees (`30.2672,-97.7431`).
- `<radius>` is optional and defaults to 50 km. Supply a number followed by `km`, `mi`, or `m` (for example `25km`, `15mi`, `5000m`). runZero treats plain numbers as kilometres.
- runZero clamps the radius between 1 metre and 1,000 km.

Examples:

```
geo:30.2672,-97.7431              # within 50 km of downtown Austin
geo:30.2672,-97.7431,25km         # within 25 km
geo:52.5200,13.4050,100mi         # within 100 miles of Berlin
geo_within:51.5074,-0.1278,5km    # synonym
```

The match is true if any location source on the asset falls inside the circle, so a workload with both a cloud region centroid and a public IP egress in the same metro area matches the same `geo:` query.

The same `geo:` keyword is available on the services inventory.

When locations are refreshed

runZero recalculates asset locations on every analysis pass. The new locations from each source replace the ones recorded before, so stale data (for example, a `cloud-region` attribute that an integration stopped reporting) disappears automatically on the next pass.

If you change a `geo*` tag value, the new location appears on the next analysis pass and not before.

Managing ownership

runZero helps you track ownership: you configure different types of owners and assign them to runZero assets and vulnerability records. You can also track ownership coverage as a goal.

Ownership types

Superusers manage the available ownership types on the **Account** > **Ownership types** page and can configure custom types there. Common examples include **Security owner**, **IT owner**, and **Compliance owner**.

Each ownership type has three fields:

- **Name**: the name of the ownership type.
- **Reference**: whether the ownership type should be correlated with the user inventory, group inventory, or neither.
- **Visibility**: whether the ownership type is visible through the asset inventory and asset details pages.

When the default `Asset Owner` ownership type is visible, runZero populates it automatically with ownership data gleaned from your configured integrations. A superuser can rename this type.

Drag the ownership types into your preferred order to prioritize them. The inventory and asset details pages display the types in that order, and only types marked `visible` appear.

Issue ownership

Issues use a separate, built-in `Issue owner` ownership type. Each issue can have one owner, and you cannot assign custom ownership types to issues. Issue ownership and Jira synchronization cover assigning an issue owner and syncing it with Jira.

Default asset owner

Integrations populate the default asset ownership type. The table below is the prioritized mapping of which attribute from each integration sets that value. If an asset has more than one of these attributes, the first match in the table wins.

Integration Source	Device/Object Type	Attribute Name	Reference Type
Google Workspace	Mobile	`owner`	User
LDAP	Computer	`managedBy`	User
LDAP	Computer	`manager`	User
Google Workspace	ChromeOS	`orgUnitPath`	Group
Defender 365	Device	`rbacGroupName`	Group

SentinelOne	Device	`groupName`	Group
Google Workspace	ChromeOS	`recentUsers.names`	User
Google Workspace	Mobile	`email.names`	User
Google Workspace	Endpoint	`email.names`	User
Miradore	Device	`user.name`	User
Intune	Device	`userDisplayName`	User
CrowdStrike	Falcon Device	`email`	User
Google Workspace	ChromeOS	`annotatedUser`	User
Google Workspace	ChromeOS	`recentUsers`	User
Google Workspace	Mobile	`email`	User
Google Workspace	Endpoint	`email`	User
Miradore	Device	`user.email`	User
Intune	Device	`emailAddress`	User
Intune	Device	`userPrincipalName`	User
SentinelOne	Device	`lastLoggedInUserName`	User

Assigning owners to assets and vulnerabilities

Once you have created custom ownership types, you can assign owners through the inventory or with an alert rule.

Superusers, administrators, and users can add, modify, and remove owners on assets and vulnerability records. Annotators can add owner values but cannot modify or remove them.

Ownership in the inventory

To assign owners through the asset or vulnerability inventory:

1. Select the assets or vulnerability records you want to update, applying a query filter if needed.
2. Click *Manage asset ownership* or *Manage vulnerability ownership* to open the ownership popup. Unowned vulnerability records on an asset inherit the ownership values applied to that asset. Vulnerability records that already have owners do not inherit the asset's ownership value.
3. Click *Add ownership type* and choose the type or types of owner to apply to the selected assets or vulnerability records.
4. Add the owner value to the field.
5. Click *Save* to apply your changes.

Applying owners with rules

To apply ownership values automatically after each scan, create an alert rule: go to **Alerts > Rules** and click *Create rule*.

1. Select the inventory query to use, such as the `asset-query-results` rule type, then click *Configure rule*.
2. Configure any settings you want.
3. Set the **Action** to *Modify asset*.
4. Enter a value in the *Set [ownership type]* field for each ownership type you want to apply. Unowned vulnerability records on an asset inherit the ownership values applied to that asset. Vulnerability records that already have owners do not inherit the asset's ownership value.
5. Save the rule.

When a scan completes, the rule adds the specified owner type and value to every asset that matches it.

Managing tasks

The Tasks overview page is where you view and manage discovery scans and other background actions. The *Active* and *Completed* task sections show standard tasks, such as scans and imports, with their current progress and summarized results. You can search or filter tasks by different attributes.

Task status values

Tasks can have the following status values:

- **New**: The task is created and waiting for a scanner or connector to pick it up.
- **Active**: A scan task is in progress and the Explorer is scanning the network.
- **Scanned**: The network scan part of a scan task has completed.
- **Connecting**: A connector task is connecting to the remote system and downloading data.
- **Connected**: A connector task has finished downloading and the data is waiting to be processed.
- **Processing, queued**: runZero has collected the data from a scan or connector task and queued it for processing.
- **Processing**: Task data is being processed.
- **Processed**: runZero has processed the task data and updated its records.
- **Stopping**: The task was asked to stop and has not yet stopped.
- **Stopped**: The task stopped successfully.
- **Canceled**: An error occurred and the task could not continue.
- **Paused**: A repeating task is paused.

Tabs

The tabs on the Tasks overview page filter the task list, and each tab has a search bar, so you can search your complete task history.

- **Active** shows all tasks currently scanning or connecting
- **Processing** shows all tasks currently processing
- **Scheduled** shows all non-recurring tasks that are scheduled to run
- **Failed** includes all tasks that could not finish because of an error or cancellation
- **Completed** lists all completed tasks in chronological order by when they finished processing
- **Recurring** shows all recurring tasks

Task details

Click a task to open a details card at the top of the page. The concentric circles show the task's progress: the outer circle tracks the scan or connection and the inner circle tracks processing. The card also includes relevant timestamps and user creation data, plus the site,

Explorer, and hosted zone settings. Completed tasks add a change summary, and recurring tasks add a list of past runs.

Scheduled tasks

Scheduled tasks are one-off actions that start at a future time, and recurring tasks repeat on a schedule. Recurring scan tasks generate a new standard task on each iteration of their schedule. Both scheduled and recurring scans launch only if their Explorer is online and no other scan tasks are running.

From the Tasks page you can remove *Scheduled* and *Recurring* tasks and stop *Active* tasks. Stopping an active *Scan* task may take a few moments, because the status does not update until the Explorer confirms that the scan was terminated.

System tasks

runZero also runs system tasks that you may notice while searching your tasks. Some examples:

- **Report-based**: a few reports create tasks when they start, including:
  - Organization overview report
  - Site and organization comparison
  - External assets report
- **Outlier calculation**: generates the outlier score shown on the asset details page and appears with the name `Outlier calculation`
- **Query**: every query pinned to the Dashboard as an Insight appears with the name Query

Dismissing failed tasks

A failed task is one that could not finish because of an error or cancellation. To remove it from the runZero Console, open the task details, click the “Dismiss” button, and confirm. Dismissing a failed task archives it, so it is no longer accessible.

Reprocessing tasks

The Reprocess Tasks tool on the tasks overview deletes the existing assets in the current organization and repopulates asset data from previous scan, import, and integration tasks using runZero’s latest algorithms. This can fix duplicated or incorrectly merged assets. To reprocess tasks:

1. On the Tasks Overview page, open the hamburger menu at the top right of the table and select “Reprocess tasks”.
2. Review the warnings in the modal that opens.
3. Enter the number of days to reprocess (the default is 7).
4. Enter the confirmation text.
5. Click “Continue”.

Deleting the organization's assets and queueing the tasks created within the specified number of days takes some time. When queueing finishes, runZero displays the number of tasks to be reprocessed, and you can monitor their progress in the console as normal.

Reprocessing deletes all assets in the organization, and with them any tags, risk, criticality, comments, or other manually applied changes. During reprocessing, asset information may be incomplete and alert rules may be triggered. If your organization has a very large number of assets, the request may time out before asset deletion completes; if it does, you can retry the request. The entire process may take several hours.

Tracking goal progress

With runZero goals, you create goals for your security initiatives and monitor progress toward achieving them. Every goal type is built on the runZero query language, and goal tracking supports all types of inventory queries.

runZero has two types of goals:

- A **Saved query** goal is based on the results of a user-defined or runZero system query against all inventory.
- A **Baseline** goal is based on the results of a user-defined query against a subset of inventory, which another user-defined query specifies.

Goal creation

Users whose default role is user or greater can create goals from the Goals page in the console. Users with viewer-level or greater access can view the goals page and see the goals that apply to organizations they have access to.

To create a new goal, click the *New goal* button on the *Goals* page in the console.

Creating saved query goals

1. On the *Goal type* tab, select the **Saved query** type.
2. On the *Build goal* tab, complete the fields you want:
    - **Name**: A name for your goal.
    - **Description**: A description of the goal.
    - **Notes**: Any notes about this goal.
    - **Organization access**: By default, the goal is created for the currently selected organization, but you may select additional organizations. Any user with User role access to every organization associated with the goal can edit it, and any user with Viewer role access to at least one of those organizations can read it.
    - **Query**: Select the query that tracks progress for this goal from the table. If needed, use the search field to find it.
    - **Threshold**: Choose whether progress is calculated as *Greater than or equal* or *Less than or equal*, and as a percent or a fixed number, then set the threshold value.
    - **Target date**: An optional target date for achieving this goal.
3. To preview the goal's current state, click *Preview*.
4. Click *Save* when you're ready, or keep making changes.

Creating baseline goals

1. On the *Goal type* tab, select the **Baseline** type.
2. On the *Build goal* tab, complete the fields you want:
    - **Name**: A name for your goal.

    - **Description**: A description of the goal.
    - **Notes**: Any notes about this goal.
    - **Organization access**: By default, the goal is created for the currently selected organization, but you may select additional organizations. Any user with User role access to every organization associated with the goal can edit it, and any user with Viewer role access to at least one of those organizations can read it.
    - **Inventory dataset**: The inventory type that the target and baseline queries search.
    - **Target query**: The query that defines what the goal targets. Its count becomes the numerator of the fraction that represents the goal's state, which runZero compares to the threshold to determine progress.
    - **Target label**: A label for the target query in the goal breakdown. Without one, runZero uses a default.
    - **Baseline query**: The query that defines the goal's scope. Its count becomes the denominator of the fraction that represents the goal's state, which runZero compares to the threshold to determine progress.
    - **Baseline label**: A label for the baseline query in the goal breakdown. Without one, runZero uses a default.
    - **Threshold**: Choose whether progress is calculated as *Greater than or equal* or *Less than or equal*, and as a percent or a fixed number, then set the threshold value.
    - **Target date**: An optional target date for achieving this goal.
3. To preview the goal's current state, click *Preview*.
4. Click *Save* when you're ready, or keep making changes.

Goal progress calculation

After you create a goal, runZero calculates its progress once a day and whenever tasks complete. The historical line chart may therefore show a different number of metric calculations from day to day, depending on how many tasks completed on a given day.

The *overall status* in the *Goal details* card on a goal details page reflects progress toward completion across all the configured organizations you have access to, so a user with more or less per-organization access may see a different status.

When a goal applies to multiple organizations, runZero calculates progress per organization and extrapolates it across all applicable organizations.

Goal events and notifications

The events system includes these goal events:

- `goal-created`: fires when a goal is created and saved.
- `goal-updated`: fires when a goal is modified and saved.
- `goal-completed`: fires when a goal's progress crosses the completion threshold.
- `goal-lapsed`: fires when the progress of a previously completed goal falls below the completion threshold.
- `goal-removed`: fires when a goal is deleted.

Pair these events with channels and rules to get notified when a goal's status changes. After you create a notification channel, create a rule for the event you want and select the channel that should receive the notification.

Understanding suppression

You can hide findings, vulnerabilities, and vulnerability groups from the platform by suppressing them. Suppression can be useful for filtering out false positives or hiding vulnerabilities that don't apply to your environment. runZero keeps suppressed objects rather than deleting them, along with metadata about when, why, and by whom they were suppressed.

How suppression works

Suppression behaves differently for each type of object. A suppressed vulnerability instance disappears from the vulnerabilities by asset inventory. A suppressed finding disappears from the findings list, and runZero also suppresses every vulnerability associated with it. A suppressed vulnerability group disappears from the vulnerability groups inventory, and runZero also suppresses every vulnerability associated with the group.

Once suppressed, a vulnerability, finding, or vulnerability group no longer counts toward an associated asset's risk score or toward dashboard metrics.

When future scans find new vulnerability instances associated with a suppressed vulnerability group or finding, runZero suppresses those new vulnerabilities automatically.

If you suppress every vulnerability instance associated with a finding or vulnerability group without suppressing the finding or group itself, it stays visible in the platform with a count of zero associated vulnerabilities until future scans detect new instances.

Suppressing and unsuppressing objects

You can suppress or unsuppress vulnerabilities, findings, and vulnerability groups from several places in the runZero platform. To suppress an individual finding or vulnerability, use the "Suppress" button in the header of its detail page. For bulk actions, use a data grid view such as the vulnerability inventory by asset, the vulnerability groups inventory, or the findings list.

When you suppress an object, a modal asks for a reason and an optional comment. Select the reason that fits best from the dropdown menu, or select "Other" and give your own reason in the comment box. Click "Suppress" to confirm. On success, the modal closes and the object is marked as suppressed.

A suppressed finding or vulnerability shows a banner at the top of its detail page. To unsuppress it, click "Remove suppression" and confirm.

In a data grid view, select rows with their checkboxes, click "Modify", and choose "Suppress" or "Remove suppression" from the dropdown menu to act on many objects at once. Each row also has action buttons for suppressing or unsuppressing that row alone.

Searching for suppressed objects

In the vulnerability inventory by asset, the vulnerability groups inventory, or the findings list, the boolean search term `suppressed` filters for suppressed or unsuppressed items. For example, `suppressed:true` finds all suppressed items, and `suppressed:any` shows suppressed and unsuppressed items together.

Each data grid view also has a "Suppression" quick filter button whose menu switches the view between all items regardless of suppression state, only suppressed items, and only unsuppressed items.

Dashboard widget

The "Suppression overview" dashboard widget shows a count of suppressed findings, vulnerabilities, and vulnerability groups at a glance. Add it to any custom dashboard from the widget library modal.

Suppression events

Suppression actions trigger runZero system events, which can drive rules in the rules engine. The suppression events are:

- `vulnerabilities-suppressed`: fires when a vulnerability is suppressed.
- `vulnerabilities-unsuppressed`: fires when a vulnerability's suppression is removed.
- `findings-suppressed`: fires when a finding is suppressed.
- `findings-unsuppressed`: fires when a finding's suppression is removed.
- `vulnerability-groups-suppressed`: fires when a vulnerability group is suppressed.
- `vulnerability-groups-unsuppressed`: fires when a vulnerability group's suppression is removed.

Understanding network segmentation

runZero multi-homed asset detection

Network segmentation is a critical security control for many businesses, and verifying that segmentation works can be hard, especially across large and complex environments. Common checks, such as reviewing firewall rules and spot testing from individual systems, only go so far, and thorough testing, such as running full network scans from every segment to every segment, can be time intensive and is difficult to justify on a regular basis.

For businesses subject to the PCI DSS requirements, validating cardholder data environment (CDE) segmentation is part of the security audit process. The PCI guidance on scoping and segmentation describes a common CDE administration model.

runZero's network bridge detection is opportunistic and far from perfect, but it may point to places where segmentation is broken and cut down on surprises in your next security audit.

Using the bridge report

The bridge report shows external networks in red and internal networks in green. Instead of a typical network map, the view shows possible paths through the network by way of multi-homed assets. To keep the graph readable, it omits assets where runZero detected only a single IP address.

Zoom in to see asset and subnet details. Click a bridged node once to highlight the networks it connects to, and a second time to open the asset page. Click a network once to highlight its connections to bridged nodes, and a second time to run a CIDR-based inventory search.

Bridge detection helps you validate network segmentation and confirm that an attacker can't reach a sensitive network from an untrusted network or asset. Examples include laptops plugged into the internal corporate network while also connected to a guest wireless segment, and systems on an untrusted network, such as a coffee shop's wireless network, that also have an active VPN connection to the corporate network.

runZero detects network bridges by looking for extra IP addresses in responses to common network probes (NetBIOS, SNMP, MDNS, UPnP, and others), and it reports bridges only when at least one asset has multiple IP addresses. Typical hardening steps, such as desktop firewalls and disabled network services, usually keep runZero from detecting multi-homed assets.

Using the asset route pathing report

Network misconfigurations and multi-homed machines can undermine network segmentation. On the runZero Platform, the asset route pathing report lets you visualize potential network paths between any two assets in an organization.

The report graphs multiple potential paths by combining IPv4 and IPv6 traceroute data with subnet analysis of detected multi-homed assets. It needs no access to the hosts or network equipment. This method finds unexpected paths between assets that existing security controls or reviews may overlook.

With a view of potential paths, you can verify whether a low-trust asset, such as a machine on a wireless guest network, can reach a high-value target, such as a database server within a cardholder data environment (CDE). The report exposes potential network segmentation violations and openings for an attacker to move laterally from one segment to another.

For an interactive Layer-2 and Layer-3 topology of every asset in the current scope, open the Network Map.

Protocol gateways

A protocol gateway is a device that bridges an IP network to a separate, sometimes non-IP fieldbus, backplane, or serial bus. Examples include a Modbus/TCP-to-Modbus-RTU gateway, an EtherNet/IP CIP rack with backplane modules, a BACnet/IP router fronting an MS/TP segment, and a HART-IP multiplexer in front of a loop of HART field instruments. From the IP side, only the gateway is reachable. The PLCs, RTUs, IEDs, meters, sensors, and CNC modules behind it are invisible to a typical port scanner.

runZero unmasks those downstream devices. When a scan identifies a supported gateway, the scanner sends protocol-native, read-only enumeration queries to the gateway and reports each downstream device as its own sub-asset in the inventory. Sub-assets link back to the gateway, so you can see, search, and report on the full field-level topology with no credentials, agents, or active control commands.

How it works

1. The standard service probe identifies a supported industrial protocol on the gateway's IP address and port (for example, Modbus/TCP on TCP/502, CIP on TCP/44818, BACnet/IP on UDP/47808, HART-IP on TCP/5094).
2. The probe sends protocol-specific identification requests to each address space the gateway exposes: Modbus unit IDs, CIP backplane slots, BACnet device instances and routed networks, KNX individual addresses, HART sub-device indices, S7 racks/slots, and so on. All requests are read-only.
3. runZero reports each enumerated device as a sub-asset with a stable foreign ID of the form `<protocol>/<port>/<address>`, plus whatever identity the protocol exposes: vendor, model, firmware, serial, name, location.
4. Each sub-asset carries `parent.id`, `parent.addresses`, and protocol-specific parent attributes that point back to the gateway asset. The console uses these to show downstream devices on the gateway's detail page and to draw edges from the gateway to each sub-asset on the Network Map.
5. Because the gateway is the only path to its sub-assets, the Network Map treats it as a protocol-gateway pivot and ranks it at the top of the choke-point panel.

Sub-assets are full assets in the inventory: they appear in search, in queries, in exports, and in dashboards. runZero matches and merges them with assets discovered through other sources (for example, a PLC enumerated through CIP and also seen directly via SNMP), so the same physical device appears once.

Supported protocols

The protocols below produce sub-assets when enumeration is enabled. The `gateway_kind` column gives the value to use with the `gateway_kind:` and `child_kind:` search keywords in both the asset inventory and the Network Map.

Protocol	Default ports	`gateway_kind`	What runZero enumerates
BACnet/IP	UDP 47808 (+ range)	`bacnet`	Device instances and routed networks reachable through the BACnet router, including BBMD / FDT entries.
EtherNet/IP CIP	TCP/UDP 44818	`cip`	Modules in the controller backplane (CPU, scanner, communications, I/O, drives) via List Identity.
Modbus/TCP	TCP 502	`modbus`	Connected unit IDs reachable through the gateway, with vendor / product / firmware where exposed.
KNXnet/IP	UDP 3671	`knxnet`	KNX devices behind the IP interface, keyed by KNX individual address.
Siemens S7Comm	TCP 102	`s7comm`	Backplane rack/slot modules (CPU, CP, IM, signal modules) for S7-300/400/1200/1500.
DNP3	TCP 20000	`dnp3`	Outstation addresses behind a DNP3 master / data concentrator.
PROFINET	UDP 34962-34964	`profinet`	Slot/subslot module list reported by PROFINET I/O devices.
EtherCAT	UDP 34980	`ethercat`	Slaves discovered on the EtherCAT segment behind the master.
Omron FINS	UDP 9600	`omronfins`	CPU modules and connected units reachable through the FINS gateway.
Mitsubishi MELSEC-Q	TCP 5006-5007	`melsecq`	CPU modules on the MELSEC backplane, including CPU model and type code.
HART-IP	TCP 5094	`hartip`	HART field instruments behind the multiplexer (Cmd 84 walk of sub-device indices).
IEC 60870-5-104	TCP 2404	`iec104`	Outstations addressed through the IEC-104 master gateway.
IEC 61850 MMS	TCP 102	`mms`	Logical devices and IEDs reported through the substation MMS gateway.
Beckhoff ADS	TCP 48898	`ads`	ADS sub-devices behind the Beckhoff TwinCAT runtime.
IEEE C37.118 Synchrophasor	TCP 4712	`c37118`	PMUs reported by the Phasor Data Concentrator.

When probed, other industrial protocols return downstream-device summaries as gateway attributes, but runZero currently does not promote those entries to standalone sub-assets. These include Emerson BSAP/IP, ANSI C12.22, Allen-Bradley CSPv4 / PCCC, Diagnostics over

IP (DoIP), Fanuc FOCAS, SEMI HSMS / SECS-GEM, M-Bus over TCP, and OPC UA. See Supported protocols for the full list of probes and what each one returns.

Enabling enumeration

Sub-asset enumeration runs as part of the standard scan when the relevant protocol is enabled and the matching probe is selected in the scan configuration. Most OT probes are off by default to keep scans of IT-only networks lean. See the Scanning OT networks playbook for the recommended enable-list and probe options.

When you enable a gateway protocol, the scanner runs both the standard fingerprint probe and the downstream enumeration step. There's no separate "enumerate sub-assets" toggle; enabling the protocol turns on both.

Finding gateways and sub-assets

Use the asset inventory search to pivot between gateways and the devices behind them. The topology keywords below come straight from the `edges` table runZero builds during enumeration, so the same queries work in **Inventory > Assets**, in any report that has an asset search box, and in the Network Map search bar.

Goal	Query
All protocol gateways	`is:gateway`
Layer-2 switches	`is:switch`
Modbus/TCP gateways	`gateway_kind:modbus`
EtherNet/IP CIP gateways	`gateway_kind:cip`
BACnet/IP routers	`gateway_kind:bacnet`
Siemens S7Comm racks	`gateway_kind:s7comm`
All sub-assets discovered behind any protocol gateway	`is:child`
Sub-assets behind a specific gateway protocol	`child_kind:modbus` (or `bacnet`, `cip`, `s7comm`, `dnp3`, `profinet`, `ethercat`, `omronfins`, `melsecq`, `hartip`, `c37118`, `knxnet`)
Gateways fronting more than 10 downstream devices	`is:gateway AND downlink_count:>10`
Public-facing protocol gateways	`is:gateway AND has_public:t`
OT assets within 2 hops of a public-facing asset	`category:OT AND hops_from_public:<=2`
Devices in the **OT** category	`category:OT`

The topology keywords accept the OT protocol names from the `gateway_kind` column in the Supported protocols table. They aren't limited to OT: `is:switch` and the existing `has:uplink` /

`has:downlink` / `has:switch` keywords cover layer-2 switch relationships.

Each sub-asset's detail page shows the parent gateway under **Source data > Parent**, and the gateway's detail page lists every sub-asset enumerated through it.

Where sub-assets show up

- Sub-assets are regular assets in the inventory. They appear in **Inventory > Assets**, in saved queries, and in CSV exports.
- A sub-asset's detail page links back to its parent gateway, and the gateway's page lists all sub-assets discovered through it.
- On the Network Map, gateways render as protocol-gateway nodes with edges out to each sub-asset on their backplane. The **Proto Gateways** category in the Asset Index counts BACnet, EtherNet/IP, Modbus, KNXnet/IP, C37.118, and the other supported gateway kinds.
- The Network Map always promotes protocol gateways into its Choke Points panel, with a +1000 Pivot Risk Score boost and a count of the assets they front. If the gateway is also EOL or carries known vulnerabilities, runZero applies an additional +200 boost.
- runZero places sub-assets in the **OT** Category and assigns them OT Functions (typically `Process Control`, with `Safety`, `Monitoring`, `Communications`, or `Human Interface` added when the protocol exposes enough detail).

Safety

All gateway enumeration uses read-only protocol requests: identification reads, list-identity, who-is, sub-device indexing. The scanner never sends control or write commands, never issues operator overrides, and never attempts to authenticate. The standard scan rate and per-protocol pacing throttle the probes.

The runZero scan engine has been independently tested against a range of OT and industrial environments, including the U.S. Department of Energy's National Renewable Energy Laboratory (NREL) CECA evaluation, which validated that active scanning is safe for operational technology. See the Scanning OT networks playbook for additional safety guidance, recommended pacing, and per-protocol options.

See also

- Network Map: visualize gateways, sub-assets, and the choke points they create.
- Scanning OT networks: how to enable OT probes and tune them for production environments.
- Supported protocols: the full list of protocols runZero probes, including which ones return backplane / sub-device summaries.
- Understanding assets: categories, functions, and how runZero classifies devices.

Understanding fingerprints

runZero automatically builds **fingerprints** for every asset it discovers. A fingerprint describes how runZero identified a device, service, or operating system from the evidence it collected, and its attributes show *what was matched* and *where it came from*.

runZero relies on fingerprints to normalize your inventory, so it can correlate and deduplicate assets accurately across discovery sources.

How fingerprints work

During discovery, runZero gathers clues such as TCP banners, mDNS names, SMB negotiation data, and TLS certificates, then classifies each clue into one or more fingerprint categories.

Fingerprints are stored in the asset record under keys that follow this pattern:

```
fp.<category>.<attribute>
```

For example:

- `fp.os.source` identifies which source provided the operating system fingerprint
- `fp.os.match` shows the string or probe result matched to identify the OS
- `fp.hw.vendor` lists the hardware vendor name

These values are visible in exports, the API, and internal datasets.

Refingerprinting and submitting improvements

If a fingerprint looks incorrect or incomplete, you can:

- **Refingerprint the asset** to re-run fingerprinting for that asset with the latest detection logic. This can update the OS, hardware, type, category, and function fields.
- **Refingerprint in bulk** by selecting one or more assets in the **Asset Inventory** and choosing **Modify > Refingerprint assets**, which re-runs fingerprinting against every selected asset with the latest detection logic.
- **Submit a fingerprint improvement** to give feedback that helps improve fingerprinting accuracy. When submitting an improvement, you can specify:
    - **Type** (e.g., *Server*, *Router*)
    - **Operating System** (e.g., *Windows 10*, *Ubuntu 22.04*)
    - **Hardware** (e.g., *VMware VM*, *Synology NAS*)
    - Optional **comments** for additional context.

These improvements help refine fingerprinting logic in future releases and improve identification of similar devices.

Fingerprint categories

Each fingerprint category covers one area of asset identification. Common categories include:

Category	Description	Example sources
`os`	Vendor, operating system, and version detected	TCP banners, SMB, mDNS
`hw`	Hardware and vendor identification	SNMP, MAC lookup, integrations
`type`	Asset type classification	runZero service probe

Common fingerprint attributes

This table lists the fingerprint attributes seen in exports and the API.

Attribute	Meaning	Sample Values
`fp.build`	Internal fingerprinting engine build version	`v4.0.251103.0-2fd02762da549a3e87ffb5b0649e1621011872dd`
`fp.hw.certainty`	Confidence level of hardware fingerprint	`0.8`, `1.0`
`fp.hw.cpe23`	CPE 2.3 identifier for the hardware	`cpe:2.3:h:dell:optiplex_7080:-:*:*:*:*:*:*:*`
`fp.hw.device`	Detected hardware device type	`Laptop`, `Server`, `Switch`
`fp.hw.match`	Raw hardware identification string	`Dell Inc. OptiPlex 7080`, `HP ProLiant DL360`
`fp.hw.product`	Hardware product name	`OptiPlex 7080`, `ProLiant DL360`, `MacBookPro18,3`
`fp.hw.serialNumber`	Serial number extracted from hardware probe	`ABC123XYZ`, `VMware-564d4a4a`
`fp.hw.source`	Source that provided the hardware fingerprint	`SNMP`, `Intune`, `CrowdStrike`, `runZero`
`fp.hw.vendor`	Vendor identified from hardware data	`Dell`, `HP`, `Apple`, `Lenovo`
`fp.hw.version`	Hardware or firmware version string	`1.0.3`, `rev A00`
`fp.os.build`	OS build number parsed from source data	`22621`, `19045`, `22D68`

`fp.os.certainty`	Confidence level of OS fingerprint	`0.95`, `1.0`
`fp.os.cpe23`	CPE 2.3 identifier for the operating system	`cpe:2.3:o:microsoft:windows_10:19045:*:*:*:*:*:*:*`
`fp.os.device`	Detected OS device type	`Desktop`, `Server`, `Mobile`
`fp.os.edition`	OS edition or variant	`Pro`, `Enterprise`, `Home`
`fp.os.match`	String matched to infer the OS	`Windows 10.0`, `Ubuntu 22.04`, `macOS 14.1`
`fp.os.product`	OS product name	`Windows 10`, `Ubuntu`, `macOS`
`fp.os.source`	Source that provided OS identification	`runZero`, `MDNS`, `Intune`, `AzureAD`
`fp.os.vendor`	Vendor identified from OS data	`Microsoft`, `Canonical`, `Apple`
`fp.os.version`	OS version parsed from matched data	`10.0.19045`, `22.04`, `14.1`
`fp.type.source`	Source that provided type classification	`runZero`, `Integration`, `Hostname`, `AzureAD`

Using fingerprints in queries

During asset processing, runZero normalizes these `fp.x.y` values into the `os`, `os_version`, `hw`, and `type` attributes.

Some sample searches using the normalized attributes:

- Windows `os` assets

  ```
  os:windows
  ```

- Ubuntu assets running `os_version` less than 20.04

  ```
  os_version:<20.04 os:ubuntu
  ```

- Assets with Dell `hardware`

  ```
  hw:dell
  ```

- Assets with the `type` server

  ```
  type:server
  ```

Troubleshooting fingerprints

To track down why a fingerprint is wrong or incomplete:

1. Check which `fp.*.source` values contributed to the identification. Multiple integrations or probes may overlap.
2. Review recent discovery runs to confirm whether an updated probe result is available.
3. Consider enriching the asset with integration data sources (e.g., Intune, CrowdStrike) to improve fingerprint accuracy.

If you run a self-hosted instance or the standalone scanner, you can also write your own custom fingerprints.

Managing alerts

runZero can trigger automatic alerts for specific events through a combination of Channels and Rules.

runZero supports Internal, Email, Email runZero Users, and Webhook channel types.

Internal channels store events in the runZero Console's Alerts list. Internal alerts support explicit acknowledgement, and you can acknowledge and clear them in bulk from the Console.

Email channels deliver mail to one or more recipients. Each message contains a summary of the alert and a link to the specifics in the runZero Console. runZero sends these messages from its own infrastructure through the Sendgrid service.

Webhook channels tie runZero alerts into third-party platforms by posting to internet-reachable web services. The post request contains a standard text message for platforms like Slack and Mattermost, plus additional fields with the full alert details.

To trigger an alert on a channel, create a Rule. Rules define which events lead to an alert on which channels. The rule's name appears in the alert content, so name it for the type of event it monitors.

Some example event types you can build rules on:

- Scan completed
- New assets found
- Assets back online
- Assets now offline
- Assets changed

Scan completion and *assets changed* rules can be noisy, but they may be useful for keeping a running log of network changes. For typical monitoring, create a rule that triggers on *Assets now offline*, *Assets back online*, and *New assets found*, then alerts an email alias or a Slack channel. Combined with recurring scans, alert rules can be a simple way to track network changes over time.

You create these rules in the rules engine, the automation framework for monitoring, alerts, and workflow management.

Using the rules engine

The Rules Engine automates monitoring, alerts, and workflow management. Use it to customize alerts for the events that matter most to your organization and to automate repetitive tasks. A rule defines the action to take when a set of conditions is met. Rules can alert your team to changes in Explorers, assets, scans, organizations, and sites, and they can tag assets or modify asset fields based on the results of a query.

Some ways to put the Rules Engine to work:

- Alert your team when new policy violations are identified.
- Modify asset fields when assets match specific criteria.
- Bulk tag assets that match a specific query.
- Get a Slack notification when a query returns new results.
- Monitor when an Explorer goes offline in the runZero Console.
- Know when organizations, sites, and users change.

Key concepts

Rules can help you stay on top of events as they happen and give you better visibility across your network, assets, and runZero deployment. A rule has four parts: an event, organization access, conditions, and an action. When the event happens and the conditions are met, runZero performs the configured action.

Events

Each rule begins with an event, which triggers it. An event can be based on a query or on a system-defined event. runZero has a library of system-defined events; the runZero events catalog lists them all. Choosing an event shows the conditions and actions available for it.

Organization access

A rule can serve any number of organizations. Any user with User role access to every organization associated with the rule can edit it, and any user with Viewer role access to at least one organization can read it. By default, a rule triggers for the currently selected organization.

Organization access on templates and channels only controls who can view and edit them, not which rules can use them. You can set up a rule for an organization with a template or channel whose access does not include that organization, as long as you have at least Viewer role access to one or more organizations in the template or channel's organization access list.

Conditions

A condition narrows the scope of your rule: the action runs only when the condition is met. You see only the conditions that apply to the event you chose. Conditions usually specify sites, organizations, and asset attributes for the event.

Actions

An action is what the rule does when the event occurs and every condition is met: send a notification to a channel, or modify an asset. Notifications need a channel and a template. Asset modifications can edit fields such as the OS vendor, OS product, OS version, hardware vendor, hardware product, hardware version, asset tags, and asset type.

Channels

A channel is how runZero tells you that an event occurred. You can create as many channels as you need, depending on who you want to reach and how. For example, a Slack channel might serve one team and an email list another.

Like rules, channels can serve any number of organizations. Any user with User role access to all associated organizations can manage a channel. Any user with Viewer role access to one or more organizations can view the channel details but can't edit or create channels.

The message body uses default text from runZero and currently can't be customized.

Create a rule

To create a rule, choose an event, define the conditions, and choose the resulting action.

Step 1: Open the Rules Engine

- From the Alerts menu, select the Rules submenu.
- Click **Create Rule** to open the editor.

Step 2: Choose an event type and configure organization access

- Give the rule a descriptive name that tells you at a glance what it does.
- Choose the event you want as your trigger.
- Browse the list of predefined events. Use the left-hand categories to narrow the list, or the search field to filter by keyword.
- Choose 'asset-query-results' or 'service-query-results' to modify the fields of the resulting assets.
- After you've chosen an event and configured organization access, click **Next**.

Step 3: Define the conditions

- The conditions you can configure depend on the event you selected.
- If you selected an asset or service query event, provide a query for the rule. The query runs against the site after the scan completes. Assets with data from non-runZero sources must be recent (seen in the last 30 days) to fall within the search, and runZero-scanned assets must be live.
- You can also scope the rule to a specific site or Explorer and, for some events, set minimum asset counts or a task type.

Step 4: Choose an action, and optionally select a specific channel or template

- An action sends a notification to the channel of your choice or modifies assets. For example, you can send an email when orphaned devices are found.

Step 5: Turn on and save the rule

- To activate the rule immediately, select the “Enable this rule” checkbox. Otherwise, save the rule and turn it on later.
- Save the rule when you’re done.

Choosing event types

Scan and asset event types can be noisy, but they are useful for tracking network changes over time. To focus on the events that matter most, track assets that go offline, assets that come back online, and newly discovered assets.

Monitoring the status of rules

The rules submenu of the Alerts page lists all rules. For each rule, you can see:

- Whether the rule is enabled.
- The event that triggers processing of the rule.
- The organizations the rule applies to, if it is limited to specific organizations.
- When the rule was last triggered.
- Whether the rule resulted in a processed action.
- When the rule was created and the username that created it.

A status of “skipped” means the rule’s preconditions weren’t met the last time it was processed, so no action was taken. A status of “processed” means its preconditions were met and its action was processed.

If processing a rule or sending a notification fails, the rule’s action status is set to “error”, and the error message appears as a tooltip on that status.

To build one of these rules step by step, follow the Alerting on asset and service changes playbook.

Creating alert templates

With the Rules Engine, you define rules that alert you on specific events, such as changes to scans, assets, and Explorers. Custom templates standardize and format the alert messages those rules send, and let you add more context and data to each alert.

A template can output raw HTML, a runZero HTML template, JSON, or text for use in emails, internal notifications, or webhooks.

Like rules and channels, a template can serve any number of organizations, and any user with User role access to all of those organizations can manage it. A user with Viewer role access to one or more organizations can view template details only, and can’t edit or create templates.

Template building basics

You write alert messages in the Mustache templating language. With a little Mustache syntax, you can build custom HTML and JSON templates that pull in runZero data.

- For Slack notifications, build a rich JSON message with Slack's Block Kit and the interactive Block Kit Builder, then use Mustache to insert the data.
- For Microsoft Teams, build rich messages in the Adaptive Cards format and insert data the same way.

A template has two fields, subject and body, and both accept Mustache syntax.

Inserting data in a template

runZero passes a standard set of objects to the template engine; you name the fields you want to insert. Use the Mustache syntax `{{variable}}` to include alert values when a rule matches.

In HTML and JSON templates, values inserted with `{{ }}` are escaped according to the rules for that format. To insert a value unescaped, use triple curly braces: `{{{task.name}}}`.

In JSON templates, the special self variable `{{.}}` at the top level outputs all available variables and their values as JSON.

Special rules for JSON

JSON templates render single values and multiple values differently:

- A string or number renders as its raw value. Embedded quotes in a string are escaped, but the string isn't wrapped in quotes, so you can put several strings or numbers into one JSON string in your template.
- An object or array renders as its full JSON representation, so you can dump arrays and objects to JSON without looping through values. But if you wrap an array value in quotes to put it inside a JSON string, the result is invalid JSON.

Take this template running on an asset with multiple IP addresses:

```
{
"addresses_1": {{addresses}},
"addresses_2": "{{#addresses}}{{.}} {{/addresses}}",
"addresses_3": "{{addresses}}"
}
```

The result:

```
{
"addresses_1": ["10.0.1.4","10.1.7.5"],
"addresses_2": "10.0.1.4 10.1.7.5 ",
```

```
"addresses_3": "["10.0.1.4","10.1.7.5"]"
}
```

The `addresses_1` substitution works as you'd expect: `{{addresses}}` is an array, so it becomes a proper JSON array of the addresses.

The second substitution also works. The template loops through each address in the array and puts the values into a single string. The values aren't quoted, but any quotes inside them are escaped.

The third substitution fails: the addresses become a JSON array of strings, and the surrounding quotes make the result invalid JSON.

Using the runZero HTML template

The runZero HTML template option wraps your alert template body in an HTML template that runZero provides. It makes consistent rich-content notification emails easier to create, and includes a base set of CSS classes you can apply to your template's contents.

Class name	Description
`padding-sm`	Applies a small amount of padding in each direction.
`padding-md`	Applies a medium amount of padding in each direction.
`color-yellow`	Applies a yellow text color to the target text
`preamble`	Creates a "preamble" block intended to go below the header. `p` tags within this block have a slightly larger font size and line height.
`card`	Creates a block that has a thin border, slightly rounded corners, and a slightly lighter background color than the main background color.
`section`	Similar to the card block, but with a darker background and a small amount of bottom margin. `section` is intended to be used within `card`.
`preserve-newlines`	Preserves newline characters in the contained text. `preserve-newlines` is intended to be paired with `section`, but can be used anywhere.
`button`	Creates a button with styling similar to the standard button seen in the runZero console, with a light blue background, white text color, and slightly rounded corners. When paired with `size-lg`, the button is given a slightly larger font size and a minimum width.

These elements also have specific styles:

Tag name	Description
`h1`	Bold font-weight, white text color, a moderate amount of bottom margin, considerably larger font size, and center-aligned text.

`h2`	Bold font-weight, white text color, a moderate amount of bottom margin, and a moderately larger font size.
`h3`	Normal font-weight, white text color, a small amount of bottom margin.
`p`	8 pixel bottom margin
`hr`	Thin light blue top border, with a considerable amount of margin above and below.
`a`	Light blue text color.

This example for the `site-created` event uses that template type and these classes:

```
<div class="preamble padding-md">
  <h1>Site created</h1>
  <p>A new site was created in an organization</p>
</div>

<div class="card padding-md">
  <h3>{{site.name}} attributes</h3>
  <div class="section padding-sm">
    <p>Created at: {{event.created_at}}</p>
    <p>ID: {{site.id}}</p>
    <p>Organization: {{organization.name}}</p>
    <p>Created by {{event.source_type}}: {{event.source_name}}</p>
  </div>
  <a class="button size-lg" href="{{console}}/sites/{{site.id}}/?_oid={{organization.id}}">View site</a>
</div>
```

We try to keep templates of this type backwards compatible, but the available classes and predefined styling may change without warning.

Inserting a value

To insert a value, put the variable name in double curly brackets: `{{variablename}}`.

This example inserts the console address:

```
The runZero Console is at {{console}}.
```

Inserting a value from an object

To insert a value from a specific object, join the object name and field name with a dot: `{{object.fieldname}}`.

This example inserts the organization name:

```
The organization name is {{organization.name}}.
```

Inserting multiple values from an object

To insert several values from one object, use a section. Open it with `{{#objectname}}` and close it with a matching `{{/objectname}}`.

This example inserts the results of a scan, including the total assets and the number of assets changed:

```
Here are the results:
{{#scan}}
Scan found {{assets_total}} assets and changed {{assets_changed}} of them.
{{/scan}}
```

Fields with multiple values

If a section refers to a field with multiple values, the template engine loops through them and renders the section once for each value.

This example loops through every asset in `report.new` and outputs each one's `names` and `addresses` fields. If `report.new` is empty, the section between the tags isn't rendered.

```
{{#report.new}}
{{names}} {{addresses}}
{{/report.new}}
```

Inside a loop, you can still refer to values from the outer object. If a named value isn't found in the current loop object, the template engine checks the outer object. This can be useful for the `{{console}}` variable, which holds the root runZero console URL.

Using boolean values

Boolean values work with the `{{#field}}{{/field}}` tags. If the field is false, the section between the tags isn't rendered.

For example:

```
{{#query.truncated}}
(Additional results were found but not included in this report)
{{/query.truncated}}
```

Objects and fields reference

Build your template from these objects and fields to include runZero data and details in your alerts.

globals

Field	Contents	Example
`console`	The base URL of the runZero web console.	https://console.runzero.com

event

The `event` object has these fields:

Field	Contents	Example
`action`	The action which triggered the event	task-completed
`created_at`	When the event was created	2021-04-02 12:50:26 -0500
`id`	The UUID of the event	b4b871db-bdf1-4a42-b82d-18ae99972228
`source_name`	Name of the thing which caused the event	Weekly security scan
`source_type`	Type of thing which caused the event	task
`success`	Whether the event succeeded	true
`target_name`	Name of the object targeted by the event	Head Office
`target_type`	The type of the object targeted by the event	site

task

For events triggered by a task, the `task` object has these fields:

Field	Contents	Example
`created_at`	When the task was created	2021-04-02 12:50:26 -0500
`created_by`	The user who created the task	user@example.com
`description`	The description of the task	Weekly scan of main network
`error`	The text of any error message for the task	explorer unavailable after 4h
`id`	The UUID of the task	b4b871db-bdf1-4a42-b82d-18ae99972228
`name`	The name of the task	Weekly Scan
`start_time`	When the task started	2021-04-02 12:50:26 -0500
`status`	The status of the task	processed
`type`	The type of task	scan
`updated_at`	When the task was last updated	2021-04-02 12:50:26 -0500
`url`	A URL linking to the task details	

organization

The `organization` object has these fields:

Field	Contents	Example
`id`	The UUID of the organization	86f12ee1-f0f1-419a-8799-63fff555777a
`name`	The name of the organization	IT Dept.

site

The `site` object has these fields:

Field	Contents	Example
`id`	The UUID of the site	49f9323a-fea1-4afc-b490-2414c3aaaeee
`name`	The name of the site	Head Office

rule

The `rule` object has these fields:

Field	Contents	Example
`action`	The action the rule said to take	notify
`created_at`	When the rule was created	2021-03-08 12:43:59 -0600
`created_by`	The user who created the rule	user@example.com
`event`	The event triggering the rule	scan-completed
`id`	The UUID of the rule	b2269a2c-69a1-4652-bcdf-899938886c17
`name`	The name of the Rule	Alert on scan
`updated_at`	When the rule was last updated	2021-04-01 17:09:40 -0500

channel

The `channel` object has these fields:

Field	Contents	Example
`id`	The UUID of the channel	7c9f0d5a-0f8e-4b3b-8a2e-1d9b5f1c2c3d
`name`	The name of the channel	Notify auditors
`type`	The channel type	email

scan

For events triggered by a scan task, a `scan` object adds these fields:

Field	Contents	Example
explorer_id	The UUID of the runZero Explorer which carried out the scan	0fd44a62-d827-41c0-b26c-4837222d8888
assets_changed	The number of assets changed as a result of the scan	9
assets_ignored	The number of assets ignored by the scan	2
assets_new	The number of new assets detected by the scan	2
assets_offline	The number of assets which were previously online but now offline	1
assets_online	The number of assets previously offline but now online again	2
assets_total	The total number of assets for the site scanned (including offline)	11
assets_unchanged	The number of assets unchanged by the scan	1
assets_updated	The total number of assets up-to-date as a result of the scan (changed + unchanged)	15
duration	Duration of the scan in seconds	26
end_time	When the scan ended	2021-04-02 12:50:26 -0500
excludes	Any IP addresses excluded from the scan	10.0.1.123
id	The UUID of the scan task	894a112c-3fb9-4301-8da7-8ce7fffb4443
name	The name of the scan task	Weekly security scan
rate	The scan rate	1000
recv_bytes	How many bytes were received during the scan	45176
recv_error	How many receive errors were detected	1
recv_packets	How many data packets were received during the scan	555
scheduled_time	When the scan was scheduled to run	2021-04-02 12:48:00 -0500
sent_bytes	How many bytes were sent during the scan	44740
sent_error	How many send errors were detected	0
sent_packets	How many data packets were sent during the scan	577
start_time	When the scan began running	2021-04-02 12:49:55 -0500

`tags`	An array of tags associated with the scan task	
`targets`	The CIDR address ranges scanned	10.0.1.0/24
`type`	The type of operation	scan

explorer

For events triggered by a scan task, an `explorer` object adds these fields (runZero 2.1.8+):

Field	Contents	Example
`id`	The UUID of the runZero Explorer which carried out the scan	0fd44a62-d827-41c0-b26c-4837222d8888
`name`	The name of the Explorer which carried out the scan	MM34B-2
`internal_ip`	The internal IP address of the Explorer which carried out the scan	10.0.1.200

search

For events triggered by a search query rule, a `search` object adds these fields:

Field	Contents	Example
`url`	A link to perform the same search	https://console.runzero.com/inventory/?search=...
`found`	The number of matches found	3
`comparator`	The operation used to compare the number of matches	>=
`value`	The value the number of matches was compared against	1

report

For “scan completed” events, a `report` object holds these results from the scan:

Field	Contents
`truncated`	Whether the set of objects was truncated due to large numbers of assets
`changed`	An array of changed assets (see below)
`new`	An array of new assets (see below)
`offline`	An array of assets now offline (see below)
`online`	An array of assets now online (see below)

The template receives at most 25 objects for email notifications and 10 for Webhook notifications.

asset (from a scan report)

Each asset in a scan report has these fields:

Field	Contents	
`addresses`	The IP address(es) of the asset	10.0.1.123
`alive`	Whether the asset responded to probes	true
`created_at`	The timestamp when the asset record was created	2021-04-02 12:50:26 -0500
`criticality_rank`	The criticality rank (0-5)	1
`detected_by`	The method by which the asset was detected	arp
`domains`	Any domains the asset was found in	WORKGROUP
`first_seen`	The timestamp when the asset was first seen	2021-04-02 12:50:26 -0500
`hw`	A summary of the asset hardware	HP LaserJet Pro
`id`	The UUID of the asset	b38295fb-bef1-fa42-b82d-18ae99972228
`last_seen`	The timestamp when the asset was most recently seen	2021-04-02 12:50:26 -0500
`macs`	Any MAC addresses detected for the asset	00:56:55:00:91:04
`modified_risk_rank`	The risk rank after applying user overrides	2
`names`	Any names detected for the asset	LP538N
`os`	A summary of the asset's operating system	Linux
`outlier_score`	The normalized outlier score (0-5)	2
`risk_rank`	The normalized risk rank (0-4)	4
`service_count`	How many services the asset is running	4
`software_count`	How many software entries were identified on the asset	10

`type`	The type of asset	Router
`updated_at`	The timestamp when the asset record was last updated	2021-04-02 12:50:26 -0500
`vulnerability_count`	How many vulnerabilities were identified on the asset	4

query

For query events, the `query` variable provides this data:

Field	Contents
`count`	The number of rows matching the search query
`assets`	If it was an asset search, the array of assets matching the query
`services`	If it was a service search, the array of services matching the query
`wlans`	If it was a wireless network search, the array of wireless networks matching the query
`truncated`	Whether the set of assets was truncated due to the query returning a large number

A query passes at most 25 assets to the template for email notifications and 10 for Webhook notifications.

asset (from a query)

Assets returned from a query rule have these fields:

Field	Contents	
`addresses`	The IP address(es) of the asset	10.0.1.123
`alive`	Whether the asset responded to probes	true
`comments`	Any comments set in the asset record	
`created_at`	The timestamp when the asset record was created	2021-04-02 12:50:26 -0500
`detected_by`	The method by which the asset was detected	arp
`domains`	Any domains the asset was found in	WORKGROUP
`first_seen`	The timestamp when the asset was first seen	2021-04-02 12:50:26 -0500
`hw`	A summary of the asset hardware	ThinkPad X1

`id`	The UUID of the asset	b38295fb-bef1-fa42-b82d-18ae99972228
`last_seen`	The timestamp when the asset was most recently seen	2021-04-02 12:50:26 -0500
`macs`	The list of MAC addresses associated with the asset	[F4:F5:E8:89:92:31, 00:D0:2D:9F:47:77]
`names`	Any names detected for the asset	laptop.local
`organization`	The organization the asset belongs to	IT
`os`	A summary of the asset's operating system	Windows 10
`service_count`	How many services the asset is running	4
`site`	The site the asset was detected at	New York
`tags`	An array of tags set on the asset	
`type`	The type of asset	Thermostat
`updated_at`	The timestamp when the asset record was last updated	2021-04-02 12:50:26 -0500

service (from a query)

Services returned from a query rule have these fields, some taken from the associated asset:

Field	Contents	Example
`id`	The UUID of the service (not the asset)	b38efadb-61f1-f332-b92d-18ae99972228
`created_at`	When the service record was created	2021-04-02 12:50:26 -0500
`summary`	A summary of the service	ciscoSystems
`port`	The TCP/UDP port the service is on	53
`vhost`	The vhost of the service	ftp.example.com
`address`	The TCP/IP address of the service	192.168.33.44
`transport`	The transport	udp
`protocol`	The name of the protocol, if known	ssh
`organization`	The name of the organization the service's asset belongs to	HR
`site`	The name of the site the service's asset belongs to	Lab

`alive`	Whether the asset offering the service was alive	true
`last_seen`	When the asset was last seen	2021-04-02 12:50:26 -0500
`first_seen`	When the asset was first seen	2021-02-11 09:38:17 -0500
`type`	The asset type offering the service	Laptop
`os`	The OS offering the service	Linux
`hw`	The hardware offering the service	APC UPS
`addresses`	A list of other IP addresses associated with the asset offering the service	[192.168.0.2, 192.168.0.3]
`macs`	The list of MAC addresses associated with the asset	[F4:F5:E8:89:92:31, 00:D0:2D:9F:47:77]
`names`	Any names associated with the asset	[fw-3, fw-3a]
`tags`	Tags set on the asset	
`domains`	Any domain associated with the asset	LOCAL
`service_count`	A count of how many services the asset offers	4
`comments`	Any comments set on the asset	

certificates (from a query)

Certificates returned from a query rule have these fields:

Field	Contents	Example
`id`	The UUID of the certificate	ff6a8817-828a-43a3-b2cb-596dff5f0bd5
`type`	The type of certificate	x509
`self_signed`	Whether the certificate is self-signed	true
`names`	All names associated with the certificate	[“example.com”, “www.example.com”]
`serial`	The serial number of the certificate	0A014A1D4D0F861EC4C6EFD6E13A2F
`validity_start`	When the certificate becomes valid	2025-11-06 15:58:04 -0500
`validity_end`	When the certificate expires	2025-11-08 15:58:04 -0500
`validity_period`	The total validity period in seconds	172800

public_key_algorithm	The algorithm used for the public key	rsaEncryption
public_key_bits	The size of the public key in bits	2048
signature_algorithm	The algorithm used for the signature	sha256WithRSAEncryption
fp_bkhash	The BKHash fingerprint of the certificate	d2c7e8f9a1b2c3d4e5f6
fp_sha1	The SHA1 fingerprint of the certificate	3c482c71d874763e560c617129793dec76e91196
fp_sha256	The SHA256 fingerprint of the certificate	19vJPgdyh3BdGgTiSSavQsKx133lzFkJhu51n6fzG+k=
cn	The common name from the certificate subject	example.com
subject	The subject distinguished name	CN=example.com,O=Example Corp,C=US
issuer	The issuer distinguished name	CN=Example CA,O=Example Corp,C=US
is_ca	Whether the certificate is a certificate authority	false
key_usage	The key usage extension values	digitalSignature
ext_key_usage	The extended key usage extension values	serverAuthentication
san_dns_names	Subject Alternative Name DNS entries	[“example.com”, “www.example.com”]
san_ip_addresses	Subject Alternative Name IP entries	[“192.168.1.1”]

`san_email_addresses`	Subject Alternative Name email entries	[“admin@example.com”]
`san_uris`	Subject Alternative Name URI entries	[“spiffe://example.com/service”]
`public_key_insecure`	Whether the public key is considered insecure	false
`signature_insecure`	Whether the signature algorithm is considered insecure	false
`public_key_parameters`	Parameters for the public key, if any	map[rsa_exponent:65537 rsa_modulus:244...]
`name_constraints`	Name constraints extension, if present	map[]
`comments`	Any comments set on the certificate	“Issued for legacy system”
`tags`	Tags applied to the certificate	[“production”, “web”]
`risk_rank`	The calculated risk rank (-1..5)	2
`modified_risk_rank`	The modified risk rank if set	1
`risk`	Either modified risk rank if set or risk_rank if unset as a name	“info”
`vulnerability_count`	Number of vulnerabilities associated	3
`service_address`	The address of the associated service	192.168.1.10
`service_port`	The port of the associated service	443
`service_transport`	The transport protocol of the service	tcp
`asset_site_id`	The UUID of the associated site	49f9323a-fea1-4afc-b490-2414c3aaaeee
`asset_site_name`	The name of the associated site	“Head Office”
`asset_id`	The UUID(s) of associated assets	b38295fb-bef1-fa42-b82d-18ae99972228

wlan (wireless LAN, from a query)

Wireless LANs (wlans) returned from a query rule have these fields:

Field	Contents	Example
id	The UUID of the wireless LAN in runZero's database	f938934b-ae23-f112-b23d-18ae99972228
last_seen	When the wlan was last seen	2021-04-02 12:50:26 -0500
essid	The ESSID of the network	Free WiFi
bssid	The BSSID of the network	c4:41:1e:99:88:77
type	The type of wireless network	infrastructure
authentication	The authentication used to access the network	WPA2-PSK
encryption	The encryption used to protect data	AES
signal	The signal strength as a percentage	86
channels	The channel of the network	11
organization	The name of the organization the service's asset belongs to	HR
site	The name of the site the service's asset belongs to	Lab

findings_with_instances

Field	Contents	Example
finding_category	The category of the finding	Open Access
finding_code	The code of the finding	rz-finding-open-access-default-credentials
finding_id	The ID of the finding	b1f8a2e7-3c4d-4e6a-9c2f-1a5e7d8b9c0a
finding_name	The name of the finding	Service Accessible With Default Credentials
finding_risk_rank	The normalized risk rank of the finding (0-4)	3
finding_vulnerability_count	How many vulnerabilities are associated with the finding; the same value as new_vulnerability_count	8
new_vulnerability_count	How many vulnerabilities are associated with the finding after the task was run	8

`old_vulnerability_count`	How many vulnerabilities were associated with the finding the last time the task was run	5
`organization_id`	The ID of the organization	e2d3c4b5-6a7b-8c9d-0e1f-2a3b4c5d6e7f
`site_id`	The ID of the site	f3e4d5c6-b7a8-9c0d-1e2f-3a4b5c6d7e8f
`site_name`	The name of the site	Primary
`task_id`	The ID of the task	a4b5c6d7-e8f9-0a1b-2c3d-4e5f6a7b8c9d
`task_name`	The name of the task	Metrics
`task_start_time`	The start time of the task	2026-03-02 11:16:59 -0500
`task_type`	The type of the task	analysis

rapid_response_published

For rapid-response-published events, the `rapid_response_published` object contains `new`, `removed` and `updated` lists. Objects in those lists have these fields:

Field	Contents	Example
`id`	The UUID of the new rapid response query	f938934b-ae23-f112-b23d-18ae99972228
`name`	The name of the query	Rapid Response: Assets with multiple RCE exploits
`description`	Plaintext description of the query	Certain assets can be exploited
`solution`	Plaintext description of how to resolve the issue exposed by this query	Update the Chipland Exploitable NVR to v3.2 or newer
`query`	The query contents	`hw:="Chipland Exploitable NVR"`
`effective_query`	Query contents that are used to match assets. Includes any system predicates, such as `alive:t`.	`alive:t AND (hw:="Chipland Exploitable NVR")`
`finding_code`	The finding code for the query, if assigned	rz-finding-rapid-response-assets
`exploitable`	Whether or not the rapid response is known exploitable	true

`cve`	Newline-separated list of CVEs that may apply to this query	`CVE-2025-12345\nCVE-2025-12346`
`console_inventory_path`	The path to the console inventory page for the query	`/inventory`
`attributes`	Miscellaneous attributes, such as reference URLs and runZero blog links.	See example below table

Example attributes object:

```
{
  "links": {
    "rz_blog": [
      {
        "title": "How to find Chipland Exploitable NVR on your network",
        "url": "https://www.runzero.com/blog/"
      }
    ],
    "urls": [
      {
        "url": "https://example.local/1.html"
      },
      {
        "url": "https://example.local/2.html"
      }
    ]
  }
}
```

rapid_response_with_matches

For rapid-response-with-matches events, these fields are available:

Field	Contents	Example
`id`	The UUID of the rapid response query	f938934b-ae23-f112-b23d-18ae99972228
`query`	Attributes of the query. See `rapid_response_published` for the fields available.	See `rapid_response_published` fields
`old_count`	How many resources matched last time this query was run.	0
`new_count`	How many resources matched the most recent time this query was run.	2

Example: Alert when scan completes

Say you want an alert whenever a scan completes. A rule can email you the details, such as the number of new, online, offline, and modified assets. The steps below build those details

into a template.

Step 1. Create a template

Create a template from the Alerts page.

- Name the template something like `Email the team when a scan completes.`
- For the template type, choose `HTML`, since this template is for emails.
- For the subject line, enter something descriptive, like `runZero scan {{scan.name}} completed at {{scan.end_time}}`. The subject accepts Mustache syntax too.

Step 2. Create the body message

The email body reports how many new, online, offline, and modified assets there are, and lists details of the new assets discovered.

The body looks like this:

```
<h1>{{site.name}}</h1>

<h2>Scan Results</h2>
{{#scan}}
<ul>
<li>{{assets_new}} new assets</li>
<li>{{assets_online}} online assets</li>
<li>{{assets_offline}} offline assets</li>
<li>{{assets_changed}} modified assets</li>
</ul>
{{/scan}}

<h2>New assets</h2>
<ul>
{{#report.new}}
<li>{{names}} {{addresses}} {{os}}</li>
{{/report.new}}
{{^report.new}}
<li>No new assets were discovered.</li>
{{/report.new}}
</ul>

<p><a href="{{task.url}}">View the scan results</a></p>
```

The caret (^) marks an inverted section, which renders only when the list is empty or the value is empty or false.

Step 3. Save the template and create a rule

Once saved, the template is available to choose when you create a rule.

Sample JSON templates

A JSON array of new assets found by a scan

```
[
  {{#report.new}}
    {
      "addresses": "{{addresses}}",
      "alive": "{{alive}}",
      "detected_by": "{{detected_by}}",
      "domains": "{{domains}}",
      "first_seen": "{{first_seen}}",
      "hw": "{{hw}}",
      "names": "{{names}}",
      "os": "{{os}}",
      "type": "{{type}}"
    },
  {{/report.new}}
  null
]
```

The trailing `null` keeps the array valid JSON, because JSON doesn't allow trailing commas.

All available attributes

Output all available variables and their attributes as a JSON object:

```
{{.}}
```

Or build your own JSON with all attributes:

```
{
  "event": {
    "action": "{{event.action}}",
    "created_at": "{{event.created_at}}",
    "id": "{{event.id}}",
    "source_name": "{{event.source_name}}",
    "source_type": "{{event.source_type}}",
    "success": "{{event.success}}",
    "target_name": "{{event.target_name}}",
    "target_type": "{{event.target_type}}"
  },
  "organization": {
    "name": "{{organization.name}}",
    "id": "{{organization.id}}"
  },
  "site": {
    "name": "{{site.name}}",
    "id": "{{site.id}}"
  },
```

```
"truncated": "{{report.truncated}}",
    "changed": "{{report.changed}}",
    "new": "{{report.new}}",
    "offline": "{{report.offline}}",
    "online": "{{report.online}}"
  },
  "rule": {
    "action": "{{rule.action}}",
    "created_at": "{{rule.created_at}}",
    "created_by": "{{rule.created_by}}",
    "event": "{{rule.event}}",
    "id": "{{rule.id}}",
    "name": "{{rule.name}}",
    "updated_at": "{{rule.updated_at}}"
  },
  "scan": {
    "explorer_id": "{{scan.explorer_id}}",
    "assets_changed": "{{scan.assets_changed}}",
    "assets_ignored": "{{scan.assets_ignored}}",
    "assets_new": "{{scan.assets_new}}",
    "assets_offline": "{{assets_offline}}",
    "assets_online": "{{scan.assets_online}}",
    "assets_total": "{{scan.assets_total}}",
    "assets_unchanged": "{{scan.assets_unchanged}}",
    "assets_updated": "{{scan.assets_updated}}",
    "duration": "{{scan.duration}}",
    "end_time": "{{scan.end_time}}",
    "excludes": "{{scan.excludes}}",
    "id": "{{scan.id}}",
    "name": "{{scan.name}}",
    "rate": "{{scan.rate}}",
    "recv_bytes": "{{scan.recv_bytes}}",
    "recv_error": "{{scan.recv_error}}",
    "recv_packets": "{{scan.recv_packets}}",
    "scheduled_time": "{{scan.scheduled_time}}",
    "sent_bytes": "{{scan.sent_bytes}}",
    "sent_error": "{{scan.sent_error}}",
    "sent_packets": "{{scan.sent_packets}}",
    "start_time": "{{scan.start_time}}",
    "tags": "{{scan.tags}}",
    "targets": "{{scan.targets}}",
    "type": "{{scan.type}}"
  },
  "explorer": {
    "id": "{{explorer.id}}",
    "name": "{{explorer.name}}",
    "internal_ip": "{{explorer.internal_ip}}"
  },
  "search": {
    "url": "{{search.url}}",
    "found": "{{search.found}}",
    "comparator": "{{search.comparator}}",
    "value": "{{search.value}}"
```

```
"query": {
    "count": "{{query.count}}",
    "assets": "{{query.assets}}",
    "services": "{{query.services}}",
    "wlans": "{{query.wlans}}",
    "truncated": "{{query.truncated}}"
  }
}
```

Event attributes only

```
{
  "event": {{event}}
}
```

or

```
{
  "action": "{{event.action}}",
  "created_at": "{{event.created_at}}",
  "id": "{{event.id}}",
  "source_name": "{{sevent.ource_name}}",
  "source_type": "{{event.source_type}}",
  "success": "{{event.success}}",
  "target_name": "{{event.target_name}}",
  "target_type": "{{tevent.arget_type}}"
}
```

Organization attributes only

```
{
  "organization": {{organization}}
}
```

or

```
{
  "name": "{{organization.action}}",
  "id": "{{organization.id}}"
}
```

Site attributes only

```
{
  "site": {{site}}
}
```

or

```
{
  "name": "{{site.action}}",
  "id": "{{site.id}}"
}
```

Report attributes only

```
{
  "report": {{report}}
}
```

or

```
{
  "truncated": "{{report.truncated}}",
  "changed": "{{report.changed}}",
  "new": "{{report.new}}",
  "offline": "{{report.offline}}",
  "online": "{{report.online}}"
}
```

Rule attributes only

```
{
  "rule": {{rule}}
}
```

or

```
{
  "action": "{{rule.action}}",
  "created_at": "{{rule.created_at}}",
  "created_by": "{{rule.created_by}}",
  "event": "{{rule.event}}",
  "id": "{{rule.id}}",
  "name": "{{rule.name}}",
  "updated_at": "{{rule.updated_at}}"
}
```

Scan attributes only

```
{
  "scan": {{scan}}
}
```

or

```
{
  "explorer_id": "{{scan.explorer_id}}",
  "assets_changed": "{{scan.assets_changed}}",
  "assets_ignored": "{{scan.assets_ignored}}",
  "assets_new": "{{scan.assets_new}}",
  "assets_offline": "{{assets_offline}}",
  "assets_online": "{{scan.assets_online}}",
  "assets_total": "{{scan.assets_total}}",
  "assets_unchanged": "{{scan.assets_unchanged}}",
```

```
"duration": "{{scan.duration}}",
  "end_time": "{{scan.end_time}}",
  "excludes": "{{scan.excludes}}",
  "id": "{{scan.id}}",
  "name": "{{scan.name}}",
  "rate": "{{scan.rate}}",
  "recv_bytes": "{{scan.recv_bytes}}",
  "recv_error": "{{scan.recv_error}}",
  "recv_packets": "{{scan.recv_packets}}",
  "scheduled_time": "{{scan.scheduled_time}}",
  "sent_bytes": "{{scan.sent_bytes}}",
  "sent_error": "{{scan.sent_error}}",
  "sent_packets": "{{scan.sent_packets}}",
  "start_time": "{{scan.start_time}}",
  "tags": "{{scan.tags}}",
  "targets": "{{scan.targets}}",
  "type": "{{scan.type}}"
}
```

Explorer attributes only

```
{
  "explorer": {{explorer}}
}
```

or

```
{
  "id": "{{explorer.id}}",
  "name": "{{explorer.name}}",
  "internal_ip": "{{explorer.internal_ip}}"
}
```

Search attributes only

```
{
  "search": {{search}}
}
```

or

```
{
  "url": "{{search.url}}",
  "found": "{{search.found}}",
  "comparator": "{{search.comparator}}",
  "value": "{{search.value}}"
}
```

Query attributes only

```
{
  "query": {{query}}
}
```

or

```
{
  "count": "{{query.count}}",
  "assets": "{{query.assets}}",
  "services": "{{query.services}}",
  "wlans": "{{query.wlans}}",
  "truncated": "{{query.truncated}}"
}
```

Asset, service, or wireless query alert to SIEM or SOAR

```
{
  "organization": {
    "name": "{{organization.name}}",
    "id": "{{organization.id}}"
  },
  "site": {
    "name": "{{site.name}}",
    "id": "{{site.id}}"
  },
  "rule": {
    "action": "{{rule.action}}",
    "created_at": "{{rule.created_at}}",
    "created_by": "{{rule.created_by}}",
    "event": "{{rule.event}}",
    "id": "{{rule.id}}",
    "name": "{{rule.name}}",
    "updated_at": "{{rule.updated_at}}"
  },
  "search": {
    "url": "{{search.url}}",
    "found": "{{search.found}}",
    "comparator": "{{search.comparator}}",
    "value": "{{search.value}}"
  }
}
```

Asset, service, or wireless query alert to Slack

```
{
  "blocks": [
    {
      "type": "section",
      "text": {
        "type": "mrkdwn",
        "text": ":red_circle: *runZero Alert* - {{rule.name}}"
      }
    },
    {
      "type": "divider"
```

```
{
      "type": "section",
      "text": {
        "type": "mrkdwn",
        "text": "*Rule information*\n\n_Name_: {{rule.name}}\n_Type_: {{rule.event}}\n_Link_: https://
console.runzero.com/alerts/rules/{{rule.id}}"
      }
    },
    {
      "type": "section",
      "text": {
        "type": "mrkdwn",
        "text": "*Match information*\n\n_Organization_: {{organization.name}}\n_Site_: {{site.name}}\n
_Match count_: {{search.found}}\n_Search_: {{search.value}}\n_Link_: {{search.url}}"
      }
    }
  ]
}
```

Asset, service, or wireless query alert to Microsoft Teams

```
{
  "type": "message",
  "attachments": [
    {
      "contentType": "application/vnd.microsoft.card.adaptive",
      "contentUrl": null,
      "content": {
        "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
        "type": "AdaptiveCard",
        "version": "1.5",
        "body": [
          {
            "type": "TextBlock",
            "text": "runZero Alert - {{rule.name}}"
          },
          {
            "type": "TextBlock",
            "text": "**Rule information**\n\n_Name_: {{rule.name}}\n\n_Type_: {{rule.event}}\n\n_Link
_: [Rule](https://console.runzero.com/alerts/rules/{{rule.id}})",
            "wrap": true
          },
          {
            "type": "TextBlock",
            "text": "**Match information**\n\n_Organization_: {{organization.name}}\n\n_Site_: {{site.
name}}\n\n_Match count_: {{search.found}}\n\n_Search_: {{search.value}}\n\n_Link_: [Search]({{search.u
rl}})",
            "wrap": true,
            "spacing": "Medium"
          }
        ]
      }
```

```
]
}
```

Data type accepted by each channel

When you create a rule, the channel you select must accept the template's data type. For example, a Slack notification needs a plain text or JSON template.

Each channel accepts these data types:

Channel	Data type
Email	Plain text, HTML, JSON
Webhook	Plain text, JSON
Internal notification	Plain text

Managing templates

Go to Alerts > Templates to create, edit, and delete templates. Before deleting a template, make sure no rule is using it. If you delete a template that a rule still uses, the rule reverts to the default template. To put a template to work, follow the Alerting on asset and service changes playbook.

runZero events

runZero records system events for administrative actions, alert rule processing, scans, Explorers, API activity, and more. Use an event action below as the trigger for an alert rule, or search for it in the audit log.

Event catalog omitted from PDF. See help.runzero.com for the full list.

Querying your data

runZero gives you many ways to query your data, and queries fall into two groups:

- Filters or parameters you type into the search bars on pages across the console, or
- System and custom queries, whose match metrics runZero calculates as tasks complete.

Both use the same query language.

Filtering and searching data

The inventory pages are likely where you'll use these queries most, but many other pages have the same kind of search bar for filtering results. To build a query, start with these pages:

- Syntax for compound queries
- Query examples
- Keywords for the inventory pages
- Keywords for other parts of the console

System and custom queries

The Queries page lists all your saved queries: system queries published by runZero and custom queries created by you or your team.

Standard query attributes

Every query has these standard attributes:

- **Name**: A name for the saved query.
- **Description**: A description of the saved query.
- **Search query**: The query or parameters to match against.
- **Search live assets only**: When toggled to *Yes* (default), runZero adds `alive:t` to the search query so it includes only assets marked alive.
- **Type**: The inventory the query searches.
- **Category**: The category the query belongs to.
- **Severity**: A severity level for the query.
- **Automatically track query results on the dashboard**: When toggled to *Yes*, a dashboard component shows the query and its match count.

Vulnerability record attributes

System and custom queries can also create vulnerability records and attach them to matching assets. Some system queries have this enabled by default. To have a query create vulnerability records, switch **Apply a vulnerability record to matching assets** to *Yes*, then complete these fields:

- **Vulnerability ID**: Choose a unique ID to track this vulnerability within runZero.
- **CVEs**: A list of CVEs relevant to this vulnerability record (optional).
- **Solution**: How this vulnerability could be remediated on assets (optional).
- **Risk**: The risk level to associate with the vulnerability. This affects the asset risk.
- **Exploitable**: Whether an exploit is available for the vulnerability.
- **CVSS v3 base score**: The CVSS v3 base score (0.00 to 10.00).
- **CVSS v3 temporal score**: The CVSS v3 temporal score (0.00 to 10.00).
- **CVSS v2 base score**: The CVSS v2 base score (0.00 to 10.00).
- **CVSS v2 temporal score**: The CVSS v2 temporal score (0.00 to 10.00).
- **CPE 2.3 identifier**: A Common Platform Enumeration identifier in URI format (v2.3) to associate with the reported vulnerability (optional).

Creating and editing queries

Users with the default role of administrator or higher can create custom queries from the Queries page by clicking **New query**. You can edit and copy custom queries after creating them. System queries can't be edited directly, but you can copy one and change the copy.

Search query syntax

The runZero search syntax works across the Asset, Service, and Wireless Inventory, across organizations and sites, and through the Query Library. The runZero Export API uses the same inventory search syntax to filter results. For ready-to-run searches built with this syntax, see the query examples.

Query syntax

Boolean operators

Combine search terms with the `AND` and `OR` operators and group them with parentheses.

AND

The Asset Inventory query `os:"Windows 10" AND protocols:http AND protocols:smb2` shows only assets identified as Windows 10 where both SMB and a web server were discovered. Wrap search values that contain spaces in double quotes.

OR

By contrast, `os:"Windows 10" AND protocols:http OR protocols:smb2` matches Windows 10 assets running a web server `or` any asset with the SMB service exposed. Besides `AND` and `OR`, the `NOT` operator filters a query: `os:"Windows 10" AND NOT protocols:http` shows Windows 10 systems

without a web server. When the negation is the first term, drop the `AND`. The query `NOT protocol:http AND os:"Windows 10"` is equivalent to the previous search, with the terms reversed.

Wildcard and fuzzy searches

Most keywords are a fuzzy match by default. To force an exact, prefix, or suffix match, add the `=` prefix to the search term and use the `%` character as a wildcard. To match an operating system name of just `Windows`, the Asset Inventory query is `os:="Windows"`; for a prefix match on `Ubuntu Linux`, use `os:="Ubuntu Linux%"`.

Single-character wildcard

The `%` wildcard matches any number of characters. To match exactly one character, use the `_` single-character wildcard. For example, `os:="Window%"` matches both `Windows` and `Windows 10`, while `os:="Window_"` matches `Windows` but not `Windows 10`.

Time and date values

Search time and date (timestamp) fields with the `<` (less than) and `>` (greater than) operators to compare against the current time. Add `-` to compare against a relative time in the past; for example, `<-3years` finds timestamps from before three years ago. Supported units:

- hours
- minutes
- seconds
- months
- years

The special value `now` also works.

For example, the asset search `first_seen:<1year` finds assets first detected within the past year. Other examples:

```
first_seen:<3days

first_seen:>2019-08-01

first_seen:>8/1/2019

last_seen:<1week

last_seen:<2months

last_seen:<1year

created_at:>2weeks

created_at:<30minutes

updated_at:>1year

updated_at:<12hours
```

```
os_eol:<now

os_eol:>4weeks

os_eol:<-2years

os_eol_extended:>now

os_eol_extended:>90days
```

Empty values

To search for an empty value, use the `=` prefix with nothing after it. For example, the query `os:=` finds assets with no identified operating system.

This works only for single-valued attributes such as `os` and `type`, not for multi-value attributes such as `names` or `addresses`.

Asset and service inventory searches

Beyond the documented keywords, Asset and Service searches support two special search types:

- Asset Inventory searches treat unknown keywords as filters against individual Asset **attributes**.
- Service Inventory searches treat unknown keywords as filters against individual Service **data** values.

When an Asset keyword conflicts with a Service data key, or an Asset attribute conflicts with a Service keyword, use the `_asset.` and `_service.` prefixes to disambiguate.

The two searches also differ in reach. Service queries can filter against Asset attributes (`os:linux`) and Service attributes (`banner:Password`), but Asset queries are limited to summary information about services (`protocol:ssh`).

Query examples

Terms and operators combine into queries in endless ways. Use the examples below as they are, or adjust them with the search query syntax.

Network configurations and access

- Multihomed assets with public and private IP addresses:

```
alive:t AND has_public:t AND has_private:t
```

- Multihomed assets connected only to private networks:

```
multi_home:t AND has_public:f
```

- Default SSH configuration using passwords for authentication:

```
alive:t AND protocol:"ssh" AND ssh.authMethods:"=password"
```

- Microsoft FTP servers:

```
alive:t AND protocol:"ftp" AND banner:"=%Microsoft FTP%"
```

- Remote access services/protocols:

```
protocol:rdp OR protocol:vnc OR protocol:teamviewer
```

- Assets with public IPs running remote access services:

```
has_public:t OR has_public:t AND alive:t AND (protocol:rdp OR protocol:vnc OR protocol:teamviewer)
```

- Open ports used by cleartext protocols:

```
port:21 OR port:23 OR port:80 OR port:443 OR port:139 OR port:445 OR port:3306 OR port:1433 OR port:161 OR
port:8080 OR port:3389 OR port:5900
```

- Telnet on nondefault ports:

```
protocol:telnet AND NOT port:23
```

- Windows assets offering SMB services:

```
os:windows AND protocol:smb1 OR protocol:smb2
```

- Switch assets accepting username and password authentication:

```
type:switch AND (_asset.protocol:http AND NOT _asset.protocol:tls) AND ( html.inputs:"password:" OR
last.html.inputs:"password:" OR has:http.head.wwwAuthenticate OR has:last.http.head.wwwAuthenticate )
```

- Assets more than 8 hops away:

```
attribute:"ip.ttl.hops" AND ip.ttl.hops:>"8
```

Asset lifecycle and hardware

- Assets created from arbitrary responses:

```
has_mac:f AND has_name:f AND os:= AND hardware:= AND detected_by:icmp AND service_count:<2
```

- End-of-life assets:

```
os_eol:<now
```

- Assets with both OS support and extended support expired:

```
os_eol:<now AND os_eol_extended:<now
```

- Assets where OS support is EOL but still covered by extended support:

```
os_eol:<now AND os_eol_extended:>now
```

- EOL Linux operating systems:

```
os:linux AND os_eol:<now
```

- EOL Windows operating systems:

```
os:windows AND os_eol:<now
```

- Assets discovered within the past two weeks:

```
first_seen:"<2weeks"
```

- All available serial number sources:

```
protocol:snmp has:snmp.serialNumbers OR hw.serialNumber:t OR ilo.serialNumber:t
```

- Asset serial numbers from SNMP:

```
protocol:snmp has:snmp.serialNumbers
```

- Older Windows OSes:

```
os:"Windows Server 2012" OR os:"Windows 7"
```

- Older Linux OSes:

```
OS:linux AND os_eol:<now
```

- BACnet devices:

```
type:bacnet
```

- Hikvision DVRs:

```
type:dvr AND os:hikvision
```

- IoT devices:

```
type:"IP Camera" OR type:"thermostat" OR type:"Amazon Device" OR hw:"Google Chromecast" OR
type:"Game Console" OR type:"Robotic Cleaner" OR type:"Nest Device" OR type:"Network Audio" OR
type:"Smart TV" OR type:"VR Headset" OR type:"Voice Assistant""
```

- Video-related assets:

```
type:"IP Camera" OR type:"DVR" OR type:"Video Encoder"
```

Misconfigurations

- SMBv1:

```
protocol:"smb1"
```

- Remote access with common services:

```
protocol:rdp OR protocol:vnc OR protocol:teamviewer OR protocol:spice OR protocol:pca
```

- Switches with default configurations for web access:

```
type:switch AND (_asset.protocol:http AND NOT _asset.protocol:tls) AND ( html.inputs:"password:" OR
last.html.inputs:"password:" OR has:http.head.wwwAuthenticate OR has:last.http.head.wwwAuthenticate )
```

- Default SSH configurations using passwords for authentication:

```
alive:t AND protocol:"ssh" AND ssh.authMethods:"=password"
```

- Switches using Telnet or HTTP for remote access:

```
type:switch AND protocol:telnet OR protocol:http
```

- Microsoft FTP servers:

```
alive:t AND protocol:"ftp" AND banner:"=%Microsoft FTP%"
```

- Virtual machines not syncing time with the host:

```
@vmware.vm.config.tools.syncTimeWithHost:"False"
```

Weak configurations

- Telnet (vs. SSH):

```
protocol:telnet
```

- FTP on ports 10-21 (vs. FTPS on port 990):

```
protocol:ftp
```

- FTP on ports 20-21 (vs. SCP on port 22):

```
protocol:ftp
```

- HTTP on port 80 (vs. HTTPS on port 443):

```
protocol:http
```

- SSH versions < 2.0:

```
protocol:ssh AND NOT banner:"SSH-2.0"
```

- TLS:

```
tls.versionName:"=TLSv1.3" OR tls.versionName:"=TLSv1.2" OR tls.versionName:"=TLSv1.1" OR
tls.versionName:"=TLSv1.0"
```

- LDAP on port 389 (vs. LDAPS on port 636):

```
protocol:ldap OR port:389
```

- Wireless access points without WPA authentication:

```
not authentication:WPA
```

- Online assets with SSH accepting password authentication:

```
alive:t AND has:"ssh.authMethods" AND protocol:"ssh" AND (ssh.authMethods:"=password" OR
ssh.authMethods:"=password%publickey")
```

- OpenSSL versions 3.0 to 3.0.6:

```
product:openssl AND version:3.0
```

EDR / MDM

- CrowdStrike coverage gaps:

```
not edr.name:crowdstrike AND (type:server OR type:desktop OR type:laptop)
```

- Assets with CrowdStrike Agent status “Not Provisioned”:

```
@crowdstrike.dev.provisionStatus:"NotProvisioned"
```

- Assets with CrowdStrike Agent mode “Reduced Functionality”:

```
@crowdstrike.dev.reducedFunctionalityMode:"yes"
```

- Assets with CrowdStrike Agent status “Normal”:

```
@crowdstrike.dev.status:"normal"
```

- SentinelOne coverage gaps:

```
not edr.name:Sentinelone AND (type:server OR type:desktop OR type:laptop)
```

- Assets whose SentinelOne Agent requires a patch:

```
(alive:t OR scanned:f) AND has:"@sentinelone.dev.appsVulnerabilityStatus" AND
@sentinelone.dev.appsVulnerabilityStatus:"=patch_required"
```

- Assets missing either CrowdStrike or SentinelOne EDR agents:

```
NOT edr.name:crowdstrike AND (type:server OR type:desktop OR type:laptop) OR NOT edr.name:sentinelone AND (type:server OR type:desktop OR type:laptop)
```

- Miradore coverage gaps:

```
not source:Miradore AND (os:google android OR os:apple ios) AND type:mobile
```

- Microsoft Defender coverage gaps:

```
not edr.name:"Defender" AND os:Windows
```

- Assets not managed by a Microsoft product:

```
source:runzero AND NOT (source:ms365defender OR source:intune OR source:azuread)
```

- Mobile devices on the network:

```
(os:google ANDroid OR os:apple ios) AND type:mobile
```

- Known FCC security threats, like Kaspersky:

```
alive:t AND edr.name:Kaspersky
```

Virtual machine configurations

- Virtual machines with less than 8 GB of memory:

```
@vmware.vm.config.hardware.memoryMB:<"8192"
```

- Virtual machines with less than 16 GB of memory:

```
@vmware.vm.runtime.maxMemoryUsage:"16384"
```

- Virtual machines not syncing time with the host:

```
@vmware.vm.config.tools.syncTimeWithHost:"False"
```

- Virtual machines configured with floppy drives:

```
@vmware.vm.config.extra.floppy0.autodetect:"true"
```

- Virtual machines running VMware tools:

```
@vmware.vm.config.extra.guestinfo.vmtools.versionString:"_"
```

- Virtual machines running Windows:

```
source:VMware AND os:Windows
```

- Virtual machines running Linux:

```
source:VMware AND os:Linux
```

Vulnerability concerns

- PCI compliance failures (Rapid7):

```
test.pciComplianceStatus:"fail"
```

- High and Critical severity vulnerabilities on CISA's Known Exploited list (Tenable):

```
plugin.xrefs.type:"CISA-KNOWN-EXPLOITED" AND (severity:high OR severity:critical)
```

- Critical severity vulnerabilities with exploits available (Tenable):

```
plugin.exploitabilityEase:"Exploits are available" AND severity:critical
```

- High and Critical severity vulnerabilities where no exploit is required (Tenable):

```
plugin.exploitabilityEase:"No exploit is required" AND (severity:critical OR severity:high)
```

Wireless results

- Authentication exceptions on a given ESSID:

```
essid:"<ESSID>" AND NOT authentication:"wpa2-enterprise"
```

- Unknown BSSIDs broadcasting a known ESSID (exclude known BSSIDs in the query for gap analysis):

```
essid:="<ESSID>" AND NOT bssid:"<MAC address>"
```

Inventory keywords

Query and filter your runZero inventories with the search syntax and each inventory's keywords. Keywords and example values are documented for these inventories:

- Assets
- Services
- Software
- Vulnerabilities
- Certificates
- Wireless networks
- Users
- Groups

Asset inventory

When viewing assets, use these keywords to search and filter. For ready-made queries that combine these keywords, see Query examples.

User-specified fields

Comments

Use `comment:<text>` to search comments on an asset.

```
comment:"contractor laptop"

comment:"imaging server"
```

Tags

Use `tag:<term>` to search tags added to an asset. The term is the tag name, or the tag name followed by an equal sign and the tag value. Tag value matches must be exact. An empty tag value matches only tags without a value.

```
tag:"group"

tag:"group="

tag:"group=production"
```

Organization name or ID

Use `organization:<term>` to filter by organization name or ID.

```
organization:runZero

organization:"Temporary Project"

organization:f1c3ef6d-cb41-4d55-8887-6ed3cfb3d42d
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary

site:"Branch Office"

site:ad67d649-041b-439d-af59-f200053a8899
```

Explorer name or ID

Use `explorer:<term>` to filter by Explorer name or ID.

```
explorer:DESKTOP-AB451F

explorer:8b927a8e-d405-40e9-aa47-d6afc9bff237
```

Hosted zone

Use `hosted_zone:<zone name>` to filter by the hosted runZero Explorer that found the asset. After a hosted scan, this filter can help you locate externally facing assets.

Owner

Use `owner:<term>` to filter by owner name.

```
owner:user@runzero.com
owner:"Security Team"
```

Ownership status

Use `owner_count:<number>` to filter by owner count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
owner_count:>0
owner_count:0
```

Use `has_owner:<boolean>` to find assets with owners or assets missing owners.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_owner:t
has_owner:f
```

Use `ownership_type:<term>` to filter assets by ownership type name. This returns assets that have an owner assigned for that ownership type.

```
ownership_type:"Asset Owner"
ownership_type:"Security Owner"
```

Asset fields

Asset ID

Use `id:<uuid>` to filter by the ID field, the asset's unique identifier written as a UUID.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Operating system

Use `os:<text>` to search the operating system field, a string describing the detected operating system software. Use `os_version:<number>` to search the OS version, if available.

```
os:"Windows"

os:"Ubuntu Linux"
```

```
os_version:8
```

OS CPE

Use `os.cpe23:<text>` to search the operating system Common Platform Enumeration (CPE) field, a string describing the detected operating system software in the CPE naming scheme. When runZero fingerprints the operating system but the NIST database has no official matching entry, runZero generates an unofficial CPE that includes `r0_unofficial` in the `other` field of the CPE.

```
os.cpe23:"ubuntu"

os.cpe23:="cpe:/o:canonical:ubuntu_linux:22.04.1"

os.cpe23:="cpe:/o:alma:linux:-::~~~~~r0_unofficial"
```

Type

Use `type:<text>` to search the type field, a string describing the detected system type, such as Desktop, Laptop, Server, BMC, or Mobile.

```
type:Desktop

type:BMC

type:"Game Console"
```

Hardware

Use `hardware:<text>` to search the hardware field, a string describing the detected physical hardware, such as `macMini` or `Nintendo Switch`.

```
hardware:Switch

hardware:macMini
```

Hostnames

Use `name:<text>` to search the hostnames associated with an asset. runZero obtains these names from DNS and exposed services.

```
name:"www"

name:"TV"
```

To find assets with a hostname that has a specific prefix or suffix, use the `:=` exact match operator with `%` as a wildcard:

```
name:="FTP.%"

name:="%-09"
```

Use `name_count:<number>` to search the hostname count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
name_count:>1
```

Use `name_overlap:<boolean>` to find assets sharing the same name. The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
name_overlap:t
```

Domains

Use `domain:<domainname>` to search the domain names associated with an asset. runZero obtains these domains from DNS and exposed services.

```
domain:"amazon.com"
```

```
domain:"corp.lan"
```

```
domain:"WORKGROUP"
```

Use `domain_count:<number>` to search the domain count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
domain_count:>1
```

Addresses

Use `address:<ip>` to search the primary and secondary addresses associated with an asset, `primary_address:<ip>` to search only the primary addresses, or `secondary_address:<ip>` to search only the secondary addresses. These keywords also match by CIDR mask and by wildcard using '%'. runZero treats a comma-separated list of addresses as an efficient multiple-match.

```
address:192.168.0.1
```

```
address:10.0.0
```

```
address:10.1.2.0/24
```

```
address:%.0.1
```

```
address:10.%.254
```

```
address:10.0.0.1,10.0.0.2,10.0.0.3
```

Use `address_count:<term>` and `address_extra_count:<number>` to search the primary and secondary address counts. These terms support numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
address_extra_count:0
```

Use `address_overlap:<boolean>` to find assets sharing primary IP addresses. Add the `site` keyword to narrow the results to a single site. The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
address_overlap:t
```

Use `address_extra_overlap:<boolean>` to find assets sharing secondary IP addresses. Add the `site` keyword to narrow the results to a single site. The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
address_extra_overlap:t
```

Networks

Use `net:<cidr>` to search the primary and secondary addresses associated with an asset by CIDR mask.

```
net:192.168.0.0/24
```

Default SNMP communities

Use `has:snmp.v2DefaultCommunities` to find assets with a default SNMP community (public, private, and other defaults).

```
has:snmp.v2DefaultCommunities
snmp.v2DefaultCommunities:public
```

Public address

Use the keyword `has_public` and the syntax `has_public:<boolean>` to locate any asset with a non-reserved IP address. These are often public-facing systems, though public IPs can also be used internally behind a firewall. This filter includes public IPv6 addresses; to search for only public IPv4 addresses, use `has_public_v4`.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_public:true
```

Private address

Use the keyword `has_private` and the syntax `has_private:<boolean>` to locate any asset with a private IP address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_private:false
```

IPv6 address

Use the keyword `has_ipv6` and the syntax `has_ipv6:<boolean>` to locate any asset with an identified IPv6 address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_ipv6:false
```

Link-local IPv6 address

Use the keyword `has_link_local` and the syntax `has_link_local:<boolean>` to locate any asset with an identified IPv6 link local (`fe80::`) address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_link_local:true
```

MAC address

Use `mac:<term>` to search the MAC addresses associated with an asset.

```
mac:00:5c:04
```

```
mac:00:00:1c
```

Use `mac_count:<number>` to search the MAC address count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
mac_count:>2
```

With the exact match operator (`:=`), you can also search for full MAC addresses in Cisco format or dash-separated format:

```
mac:=00-10-fa-c2-bf-d5
```

```
mac:=0010.fac2.bfd5
```

Use `mac_overlap:<boolean>` to find assets sharing the same MAC address. The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
mac_overlap:t
```

MAC address vendors

Use `mac_vendor:<text>` to search the vendors associated with an asset's MAC addresses.

```
mac_vendor:Apple
```

```
mac_vendor:"Intel Corporate"
```

Use `newest_mac_vendor:<text>` to search only the vendor of the newest MAC address.

```
newest_mac_vendor:Apple
```

Use `mac_vendor_count:<number>` to search the MAC address vendor count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
mac_vendor_count:0
```

MAC address age

Use `mac_age:<term>` to search the allocation date of the newest MAC address associated with an asset. The term supports the standard runZero time comparison syntax.

```
mac_age:>1year
```

```
mac_age:<6months
```

```
mac_age:2019-12-31
```

Outlier score

Use `outlier_score:<value>` to search the calculated outlier score of assets. The outlier score ranges from 0 to 5 inclusive. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
outlier_score:>2
```

```
outlier_score:0
```

Upstream switch IP address

Use `switch.ip:<address>` to search the IP address of the upstream switch that assets connect to.

```
switch.ip:192.168.1.1
```

```
switch.ip:fe80::81f2:1c9d:5ac9:5420
```

Upstream switch name

Use `switch.name:<hostname>` to search the hostname of the upstream switch that assets connect to.

```
switch.name:"SWITCH-1"

switch.name:office
```

Upstream switch port

Use `switch.port:<address>-<port number string>` to search the port on the upstream switch that assets connect to.

```
switch.port:192.168.1.1-25

switch.port:10.1.2.3-0/1/2
```

Upstream switch shared port

Use `attribute:switch.portShared` to find assets that connect to a switch port reporting multiple MAC addresses.

```
attribute:switch.portShared
```

Attributes

Use `attribute:<term>` to search the asset attribute fields, such as the port used to detect the TTL.

```
attribute:"ip.ttl.port"

attribute:"cpe:/a:isc:bind:9.11.3"

attribute:"9.11.3"
```

Use the `has:<attribute-name>` keyword to check whether an asset has an attribute defined. Invert the `has` keyword with `not has:<term>` to find missing fields.

```
has:"ip.ttl.port"

not has:"rdns.names"
```

Alongside the standard fields, these special attributes are available:

- `has:screenshot` returns assets where at least one screenshot was obtained.
- `has:icons` returns assets where at least one icon was obtained (HTTP, UPnP, or similar).
- `has:uplink` returns assets seen in the CAM table of a network switch.
- `has:switch` is an alias of `has:uplink`.
- `has:downlink` returns assets where the CAM table was queried and at least one other asset was connected.

- `has:unmapped` returns assets where the CAM table was queried and at least one other asset was connected but not identified by IP.

See Topology role below for the related `is:`, `gateway_kind:`, `child_kind:`, `downlink_count:`, and `hops_from_public:` keywords.

You can also use the attribute name directly as a term. If the attribute name conflicts with an existing term, add the `_asset.` prefix to disambiguate the query.

```
ip.ttl.port:80

rdns.names:"router"

_asset.ip.ttl.hops:"1"
```

Use `@<integration>.<source>.<attribute>:<term>` to query foreign attributes from third-party inbound integrations. The table below lists the prefix for each integration.

Integration	Prefix
Miradore	`@miradore.dev.`
AWS EC2	`@aws.ec2.`
AWS ELB & ELBv2	`@aws.elb.`
AWS RDS	`@aws.rds.`
CrowdStrike	`@crowdstrike.dev.`
Azure Load Balancer	`@azure.vm.`
Azure VM	`@azure.vm.`
Azure Scale Set VM	`@azure.vmss.`
Censys	`@censys.host.`
VMWare	`@vmware.vm.`
GCP Load Balancer	`@gcp.lb.`
GCP E2-Micro VM	`@gcp.vm.`
GCP CloudSQL	`@gcp.cloudsql.`
SentinelOne	`@sentinelone.dev.`
Tenable.io & Nessus	`@tenable.dev.`
Rapid7 Nexpose & InsightVM	`@rapid7.dev.`
Qualys VMDR	`@qualys.dev.`
Shodan	`@shodan.dev.`
Azure AD	`azuread`
Active Directory (LDAP)	`@ldap.computer.`

Microsoft 365 Defender	`@ms365defender.dev.`
Microsoft Intune	`@intune.dev.`
Google Workspace ChromeOS	`@googleworkspace.chromeos.`
Google Workspace Endpoint	`@googleworkspace.endpoint.`
Google Workspace Mobile	`@googleworkspace.mobile.`

```
@aws.ec2.region:="us-east-2"
```

```
@crowdstrike.dev.agentVersion:="6.49.16201.0"
```

```
@googleworkspace.chromeos.model:="HP Chromebook"
```

Foreign attributes support semver-style comparison operators (`>`, `>=`, `<`, `<=`, `=`). Include at least the major and minor version number to use this comparison. Version comparison ignores any non-numerical text in the prefix or suffix.

```
@jamf.custom.osVersion:>6.0
```

Topology role

runZero builds a topology graph from layer-2 switch CAM tables and from sub-asset enumeration through protocol gateways. The keywords below filter assets based on that graph and work in **Inventory** → **Assets**, in any asset search box, and in the Network Map.

Keyword	Matches
`is:switch`	Asset is a layer-2 switch with one or more observed CAM-table entries (downstream ports).
`is:gateway`	Asset is a protocol gateway fronting one or more enumerated sub-assets.
`is:child`	Asset is a sub-asset enumerated through a protocol gateway.
`gateway_kind:<proto>`	Asset is a gateway for the given protocol — `bacnet`, `cip`, `modbus`, `s7comm`, `dnp3`, `profinet`, `ethercat`, `omronfins`, `melsecq`, `hartip`, `c37118`, `knxnet`.
`child_kind:<proto>`	Asset is a sub-asset behind a gateway of the given protocol (same vocabulary as `gateway_kind`).
`downlink_count:<n>`	Number of downstream links (switch ports + OT gateway children). Supports comparisons: `>10`, `<=0`, `>=1`, ranges.
`hops_from_public:<n>`	Minimum topology hops to a public-facing asset (`0` = the asset itself is public-facing). Supports comparisons.
`has:uplink`	Asset is connected to an upstream switch port (alias `has:switch`).
`has:downlink`	Asset is a switch with at least one downstream port observed.

Examples:

```
is:gateway
```

```
gateway_kind:modbus

is:child AND child_kind:cip

is:switch AND downlink_count:>10

category:OT AND hops_from_public:<=2
```

Asset services

Service ports

Use `port:<number>` to search the TCP and UDP services associated with an asset by port number.

```
port:80

port:161
```

Service TCP ports

Use `tcp:<number>` to search the TCP services associated with an asset by port number.

```
tcp:443
```

Use `service_ports_tcp:=<list>` to find assets with a specific list of TCP ports open. List the values in ascending numerical order, separated by commas.

```
service_ports_tcp:=80,443
```

Service UDP ports

Use `udp:<number>` to search the UDP services associated with an asset by port number.

```
udp:53
```

Use `service_ports_udp:=<list>` to find assets with a specific list of UDP ports open. List the values in ascending numerical order, separated by commas.

```
service_ports_udp:=53,123
```

Service protocols

Use `service_protocols:<term>` (or `protocol:<term>` for short) to search the identified service protocols associated with an asset.

```
protocol:http

service_protocol:telnet
```

Use `protocol_count:<number>` to search the protocol count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
protocol_count:>1
```

Service products

Use `service_products:<term>` (or `product:<term>` for short) to search the identified service products associated with an asset.

```
product:openssh
```

```
service_products:nginx
```

Use `product_count:<number>` to search the product count. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
product_count:>3
```

Service counts

Use the following keywords to search the number of services associated with an asset:

- `service_count_tcp:<number>`
- `service_count_udp:<number>`
- `service_count_icmp:<number>`
- `service_count_arp:<number>`

These keywords support numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

Examples:

```
service_count_tcp:>=5
```

```
service_count_arp:0
```

```
service_count_udp:<=1
```

Asset tracking fields

Timestamps

Use the following keywords to search the asset timestamp fields (`first_seen`, `last_seen`, `created_at`, `updated_at`, `os_eol`, `os_eol_extended`):

- `first_seen:<term>`
- `last_seen:<term>`
- `created_at:<term>`
- `updated_at:<term>`
- `os_eol:<term>`
- `os_eol_extended:<term>`

The term supports the standard runZero time comparison syntax.

```
first_seen:<3days
```

```
first_seen:>2019-08-01

first_seen:>8/1/2019

last_seen:<1week

last_seen:<2months

last_seen:<1year

created_at:>2weeks

created_at:<30minutes

updated_at:>1year

updated_at:<12hours

os_eol:<now

os_eol:>4weeks

os_eol_extended:>now

os_eol_extended:>90days
```

Online status

Use `online:<boolean>`, or its inverse `offline:<boolean>`, to search the online status of an asset.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
online:t

online:1

offline:0
```

Operating system support status

Use `os_eol_expired:<boolean>` to find assets based on whether their operating systems are End of Life (EOL). This field evaluates both the `os_eol` and `os_eol_extended` values and returns only assets with expired coverage.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
os_eol_expired:t
```

```
os_eol_expired:1
```

```
os_eol_expired:no
```

Detection method

Use `det:<term>` or `detected_by:<term>` to search the detected-by attribute of an asset. The term is one of `arp`, `icmp`, `<portnumber>-tcp`, or `<portnumber>-udp`. With multiple detections, the priority is `arp`, then `icmp`, then the first detected service.

```
det:arp
```

```
detected_by:80-tcp
```

```
det:53-udp
```

Time to live (TTL) comparisons

Use `ttl:<term>` and `lowest_ttl:<term>` to search the lowest TTL of an asset. TTL is the estimated number of hops between the scan source and the asset.

This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
lowest_ttl:>3
```

Round trip time (RTT) comparisons

Use `rtt:<term>` and `lowest_rtt:<term>` to search the lowest RTT of an asset. RTT is the round-trip response time of a probe, measured in nanoseconds (1,000,000 == 1ms).

This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
lowest_rtt:>50000000
```

Multiple MAC address status

Use `multi_mac:<boolean>` to find assets with multiple MAC addresses.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
multi_mac:t
```

Any MAC address status

Use `has_mac:<boolean>` to find assets with any MAC address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_mac:yes
```

```
has_mac:f
```

Multiple IP address status

Use `multi_home:<boolean>` to find assets with multiple IP addresses.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
multi_home:t
```

Multiple hostname status

Use `multi_name:<boolean>` to find assets with multiple hostnames.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
multi_name:yes
```

```
multi_name:false
```

Software installations

Use `software:<term>>` to find assets with associated software.

The term has three forms:

- `software:<product>` finds assets with a software product that matches the term.
- `software:<product>/<version>` finds assets with a software product and version that match the term exactly.
- `software:<vendor>/<product>/<version>` finds assets with a software vendor, product, and version that match the term exactly.

All three forms accept `%` as a wildcard at the beginning, middle, or end of the term.

Version terms accept comparators to search for versions less than or greater than a specific version. For example, `software:"Google/Google Chrome/>=135"` returns all assets with Google Chrome version 135 or greater.

```
software:IIS
```

```
software:Microsoft/IIS/10.0
```

Certificate fields

Use the following keywords to search certificate-related fields.

Certificate ID

Use `certificate_id:<uuid>` to filter by certificate ID.

```
certificate_id:4e3a2b1c-5d6f-7a8b-9c0d-1e2f3a4b5c6d
```

Certificate type

Use `certificate_type:<term>` to search by certificate type.

```
certificate_type:"x509"
```

Certificate serial

Use `certificate_serial:<term>` to search by certificate serial number.

```
certificate_serial:"01A23B45C"
```

Certificate public key

Use `certificate_public_key:<term>` to search by certificate public key.

```
certificate_public_key:"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A..."
```

Certificate public key algorithm

Use `certificate_public_key_algorithm:<term>` to search by certificate public key algorithm.

```
certificate_public_key_algorithm:"ecPublicKey"
```

Certificate signature

Use `certificate_signature:<term>` to search by certificate signature.

```
certificate_signature:"abcdef1234567890"
```

Certificate signature algorithm

Use `certificate_signature_algorithm:<term>` to search by certificate signature algorithm.

```
certificate_signature_algorithm:"rsaEncryption"
```

Certificate fingerprint (bkhash)

Use `certificate_fp_bkhash:<term>` to search by certificate fingerprint (bkhash).

```
certificate_fp_bkhash:"d2c7e8f9..."
```

Certificate fingerprint (SHA1)

Use `certificate_fp_sha1:<term>` to search by certificate SHA1 fingerprint.

```
certificate_fp_sha1:"349eb7db55bbad9d7deabc5effdfd5521ed984a0"
```

Certificate fingerprint (SHA256)

Use `certificate_fp_sha256:<term>` to search by certificate SHA256 fingerprint.

```
certificate_fp_sha256:"SHA256:Sc0pLUCvlNaGtu4Xy2fOTe1A6cC+KyU3x7xuN8+aLtA="
```

Certificate subject

Use `certificate_subject:<term>` to search by certificate subject.

```
certificate_subject:"CN=example.com,O=ExampleCorp,C=US"
```

Certificate common name (CN)

Use `certificate_cn:<term>` to search by certificate common name.

```
certificate_cn:"example.com"
```

Certificate issuer

Use `certificate_issuer:<term>` to search by certificate issuing authority.

```
certificate_issuer:"CN=Example CA,O=Example Corp,C=US"
```

Certificate subject key ID

Use `certificate_subject_key_id:<term>` to search by certificate subject key ID.

```
certificate_subject_key_id:"1234567890abcdef"
```

Certificate authority key ID

Use `certificate_authority_key_id:<term>` to search by certificate issuing authority key ID.

```
certificate_authority_key_id:"abcdef1234567890"
```

Service inventory

When viewing services, use these keywords to search and filter.

Ports

Use `port:<number>` to search TCP and UDP services by port number. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
port:<=25
```

TCP ports

Use `tcp:<number>` to search TCP services by port number.

```
tcp:53
```

Use `service_ports_tcp:=<list>` to find all services on assets with a specific list of TCP ports open. List the values in ascending numerical order, separated by commas.

```
service_ports_tcp:=80,443
```

UDP ports

Use `udp:<number>` to search UDP services by port number.

```
udp:443
```

Use `service_ports_udp:=<list>` to find all services on assets with a specific list of UDP ports open. List the values in ascending numerical order, separated by commas.

```
service_ports_udp:=53,123
```

Transport

Use `transport:<term>` to search the transport of a service by name.

```
transport:tcp
```

```
transport:udp
```

```
transport:icmp
```

Protocol

Use `service_protocols:<term>` (or `protocol:<term>` for short) to search the protocols associated with services.

```
protocol:http
```

```
protocol:telnet
```

Assets with product

Use `service_products:<term>` (or `product:<term>` for short) to search the identified service products associated with an asset and return all services for the matching assets.

```
product:openssh
```

```
service_products:nginx
```

Virtual host (vHost)

Use `vhost:<text>` to search the virtual hosts associated with a service by name.

```
vhost:"www"
```

Address

Use the `service_address` keyword to match the service IP address.

```
service_address:192.168.0.1
```

Public address

Use the keyword `service_has_public` and the syntax `service_has_public:<boolean>` to locate any service with a non-reserved IP address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
service_has_public:true
```

Private address

Use the keyword `service_has_private` and the syntax `service_has_private:<boolean>` to locate any service with a private IP address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
service_has_private:false
```

IPv6 address

Use the keyword `service_has_ipv6` and the syntax `service_has_ipv6:<boolean>` to locate any service with an identified IPv6 address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
service_has_ipv6:false
```

Link-local IPv6 address

Use the keyword `service_has_link_local` and the syntax `service_has_link_local:<boolean>` to locate any service with an identified IPv6 link local (`fe80::`) address.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
service_has_link_local:true
```

Assets with outlier score

Use `outlier_score:<value>` to search the calculated outlier score of assets and return all services on those assets. The outlier score ranges from 0 to 5 inclusive. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
outlier_score:>2
```

```
outlier_score:0
```

Assets with MAC address vendors

Use `mac_vendor:<text>` to search the vendors associated with an asset's MAC addresses and return all services on those assets.

```
mac_vendor:Apple
```

```
mac_vendor:"Intel Corporate"
```

Use `newest_mac_vendor:<text>` to search only the vendor of the newest MAC address.

```
newest_mac_vendor:Apple
```

Assets with MAC address age

Use `mac_age:<term>` to search the age of the newest MAC address on each asset and return all services on those assets. The term supports the standard runZero time comparison syntax.

```
mac_age:>1year
```

```
mac_age:<6months
```

```
mac_age:2019-12-31
```

Attributes

Use `<attribute>:<term>` to search any service attribute. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

If the attribute name conflicts with an existing term, add the `_service.` prefix to disambiguate the query.

Service attribute searches can be slow, so it is often better to put an `_asset.protocol:<term>` filter in front of the service attribute query. For example, to search SSH banners, use `_asset.protocol:ssh AND banner:<term>`.

```
banner:password
```

```
service.product:"OpenSSH"
```

```
html.title:"Apache2 Ubuntu Default Page"
```

```
http.code:>=500
```

```
screenshot.image.size:=>100000
```

```
_service.arp.macVendor:Xerox
```

Use the `has` keyword to check whether a service has an attribute at all. Invert the `has` keyword with `not has:<term>` to find missing fields.

```
has:"http.head.server"
```

```
not has:"html.title"
```

Certificate fields

Use the following keywords to search the certificates related to a service.

Certificate ID

Use `certificate_id:<uuid>` to filter by certificate ID.

```
certificate_id:4e3a2b1c-5d6f-7a8b-9c0d-1e2f3a4b5c6d
```

Certificate fingerprint (bkhash)

Use `fp_bkhash:=<text>` to find services with a certificate matching the given bkhash fingerprint anywhere in the certificate chain.

```
fp_bkhash:=d2c7e8f9a1b2c3d4e5f6
```

Certificate fingerprint (SHA256)

Use `fp_sha256:=<text>` to find services with a certificate matching the given SHA256 fingerprint anywhere in the certificate chain.

```
fp_sha256:=SHA256:19vJPgdyh3BdGgTiSSavQsKx133lzFkJhu51n6fzG+k=
```

Certificate fingerprint (SHA1)

Use `fp_sha1:=<text>` to find services with a certificate matching the given SHA1 fingerprint anywhere in the certificate chain.

```
fp_sha1:=a1b2c3d4e5f6g7h8i9j0
```

Certificate validity fields

Use the following keywords to search the timestamps of the certificate found on the service. These fields support the standard runZero time comparison syntax.

```
certificate_valid_from:>2025-01-01
certificate_valid_until:<2026-01-01
certificate_created_at:<1year
certificate_updated_at:>2024-12-01
certificate_last_seen:<2months
```

Certificate authority key ID

Use `certificate_authority_key_id:<text>` to search by certificate authority key ID.

```
certificate_authority_key_id:"c03152cd5a50c3827c7471cecbe99cf97aeb82e2"
```

Certificate common name (CN)

Use `certificate_cn:<text>` to search by certificate common name.

```
certificate_cn:"example.com"
```

Certificate public key algorithm

Use `certificate_public_key_algorithm:<text>` to search by public key algorithm.

```
certificate_public_key_algorithm:rsaEncryption
```

Certificate serial number

Use `certificate_serial:<text>` to search by certificate serial number.

```
certificate_serial:123456
```

Certificate signature algorithm

Use `certificate_signature_algorithm:<text>` to search by signature algorithm.

```
certificate_signature_algorithm:sha256WithRSAEncryption
```

Certificate subject key ID

Use `certificate_subject_key_id:<text>` to search by subject key ID.

```
certificate_subject_key_id:"c769916e10f850397928f62bd34ca6ec39d8e00a"
```

Certificate type

Use `certificate_type:<text>` to search by certificate type.

```
certificate_type:x509
```

Certificate key usage

Use `certificate_key_usage:<text>` to search by key usage.

```
certificate_key_usage:"digitalSignature"
```

Certificate extended key usage

Use `certificate_ext_key_usage:<text>` to search by extended key usage.

```
certificate_ext_key_usage:"serverAuth"
```

Certificate OCSP server

Use `certificate_ocsp_server:<text>` to search by OCSP server.

```
certificate_ocsp_server:http://ocsp.example.com
```

Certificate CRL distribution points

Use `certificate_crl_distribution_points:<text>` to search by CRL distribution point.

```
certificate_crl_distribution_points:http://crl.example.com/
```

Certificate issuing certificate URL

Use `certificate_issuing_certificate_url:<text>` to search by issuing certificate URL.

```
certificate_issuing_certificate_url:http://issuer.example.com/
```

Certificate signature

Use `certificate_signature:<text>` to search by certificate signature.

```
certificate_signature:Bden73ipj8B2xb1Ozy5nOvIytCktGrht5xL7ZfFlaLIBQxbGO5Iuf6Y1yICcEpYqsgSJS6JKCdw5dujmPmGRwBZfVhIbSRb0e
```

Certificate public key

Use `certificate_public_key:<text>` to search by certificate public key.

```
certificate_public_key:MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDJ899KGnqHjVuBekYqosp2l8zWbiyu2I62CzaqaouLtqn1nXaQLMdruhlN
```

Certificate comments

Use `certificate_comments:<text>` to search comments on a certificate.

```
certificate_comments:"our self signed wildcard"
```

Certificate self-signed

Use `certificate_self_signed:<boolean>` to search for self-signed certificates.

```
certificate_self_signed:true
```

Certificate hidden

Use `certificate_hidden:<boolean>` to search for certificates hidden from the inventory.

```
certificate_hidden:true
```

Certificate signature algorithm insecure

Use `certificate_signature_algorithm_insecure:<boolean>` to search for certificates with insecure signature algorithms.

```
certificate_signature_algorithm_insecure:true
```

Certificate public key insecure

Use `certificate_public_key_insecure:<boolean>` to search for certificates with insecure public key algorithm and key size combinations.

```
certificate_public_key_insecure:true
```

Certificate is CA

Use `certificate_is_ca:<boolean>` to search for certificates that are certificate authorities (CAs).

```
certificate_is_ca:true
```

Software inventory

When viewing software groups, use these keywords to search and filter. For software instances on assets, see the software instance inventory keywords.

Vendor

Use `vendor:<name>` to search the software vendor by name.

```
vendor:oracle
```

Product

Use `product:<name>` to search the software product by name.

```
product:java
```

Version

Use `version:<name>` to search the software version.

```
version:1.2.3
```

Software instance inventory

When viewing software instances on assets, use these keywords to search and filter. For software groups, see the software inventory keywords.

Source

Use `source:<name>` to search or filter by the name of the source that reported the installed software.

```
source:runzero
```

Vendor

Use `vendor:<name>` to search the software vendor by name.

```
vendor:oracle
```

Product

Use `product:<name>` to search the software product by name.

```
product:java
```

Vulnerabilities inventory

When viewing vulnerability groups, use these keywords to search and filter.

Name

Use `name:<term>` to search the name field.

```
name:"Cisco IOS Software DHCP Remote Code Execution Vulnerability"
```

```
name:"PHP < 5.3.12 / 5.4.2 CGI Query String Code Execution"
```

CVE

Use `cve:<term>` to search the CVE field.

```
cve:CVE-2021-44228
```

```
cve:CVE-2016-2183
```

KEV

Use `kev:<term>` to search by membership in a Known Exploited Vulnerability (KEV) list.

```
kev:t
```

matches vulnerabilities that appear on a KEV list.

You can also search a specific KEV list by name.

- `kev:cisa` matches vulnerabilities listed as actively exploited in the CISA Known Exploited Vulnerabilities Catalog.
- `kev:vulncheck` matches vulnerabilities listed as actively exploited in the VulnCheck Catalog.
- `kev:true` matches vulnerabilities in either list.

Severity

Use `severity:<term>` to search the severity field.

```
severity:info
```

```
severity:medium
```

Risk

Search the `Risk` and `Risk Score` fields with either numeric or keyword values. Risk score is an integer from zero through four, where `0` is Info level risk and `4` is Critical risk.

```
risk:"Critical"
```

```
risk:2
```

Vulnerability instance count

Use `count:<text>` to search the `Asset count` field.

```
count:>0
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary
```

EPSS score

Use `epss_score:<term>` to search the EPSS score. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
epss_score:>0.5
```

```
epss_score:<=0.1
```

```
epss_score:=0.9
```

Timestamps

Use the following keywords to search the vulnerability group timestamp fields (`created_at`, `suppressed_at`):

- `created_at:<term>`
- `suppressed_at:<term>`

The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks
```

```
created_at:<30minutes
```

```
suppressed_at:<1day
```

Suppression

Use `suppressed:<term>` to search vulnerability groups by their suppressed or unsuppressed state.

```
suppressed:t
```

matches suppressed vulnerability groups.

```
suppressed:f
```

matches unsuppressed vulnerability groups.

The special value `any` displays all records, regardless of suppression settings:

```
suppressed:any
```

displays both suppressed and unsuppressed vulnerability groups.

To find the user who performed the suppression, use:

`suppressed_by:<username>`

Vulnerability instance inventory

When viewing vulnerability instances on assets, use these keywords to search and filter.

Vulnerability ID

Use `id:<uuid>` to filter by the ID field, the vulnerability's unique identifier written as a UUID.

`id:a124a141-e518-4735-9878-8e89c575b1d2`

Source

Use `source:<name>` to search or filter by the name of the source that reported the vulnerability.

`source:tenable`

Severity

Use `severity:<term>` to search the severity field.

`severity:info`

`severity:medium`

Severity score

Use `severity_score:<term>` to search the severity score. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

`severity_score:<5.0`

`severity_score:>=9.0`

Risk

Use `risk:<term>` to search the risk field.

`risk:none`

`risk:critical`

Risk score

Use `risk_score:<term>` to search the risk score. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
risk_score:>7.0

risk_score:=10.0
```

Modified risk

Use `has_modified_risk:<boolean>` to find vulnerabilities whose risk was modified, either manually or by an alert rule trigger.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_modified_risk:true

has_modified_risk:1

has_modified_risk:no
```

Category

Use `category:<term>` to search the category field.

```
category:Local

category:Remote
```

Name

Use `name:<term>` to search the name field.

```
name:"Cisco IOS Software DHCP Remote Code Execution Vulnerability"

name:"PHP < 5.3.12 / 5.4.2 CGI Query String Code Execution"
```

Description

Use `description:<term>` to search the description field.

```
description:"The remote device is missing a vendor-supplied security patch."

description:"remote code execution"
```

Solution

Use `solution:<term>` to search the solution field.

```
solution:patch

solution:upgrade
```

CVE

Use `cve:<term>` to search the CVE field.

```
cve:CVE-2021-44228
```

```
cve:CVE-2016-2183
```

KEV

Use `kev:<term>` to search by membership in a Known Exploited Vulnerability (KEV) list.

```
kev:t
```

matches vulnerabilities that appear on a KEV list.

You can also search a specific KEV list by name.

- `kev:cisa` matches vulnerabilities listed as actively exploited in the CISA Known Exploited Vulnerabilities Catalog.
- `kev:vulncheck` matches vulnerabilities listed as actively exploited in the VulnCheck Catalog.
- `kev:true` matches vulnerabilities in either list.

Exploitable

Use `exploitable:<boolean>` to search for exploitable vulnerabilities.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
exploitable:true
```

```
exploitable:1
```

```
exploitable:no
```

CVSSv2 score

Use `cvss2_base_score:<term>` and `cvss2_temporal_score:<term>` to search the CVSSv2 fields, `cvss2_base_score` and `cvss2_temporal_score`. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
cvss2_base_score:>6.5
```

```
cvss2_base_score:<=3.0
```

```
cvss2_temporal_score:=10.0
```

```
cvss2_temporal_score:<5.0
```

CVSSv3 score

Use `cvss3_base_score:<term>` and `cvss3_temporal_score:<term>` to search the CVSSv3 fields, `cvss3_base_score` and `cvss3_temporal_score`. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
cvss3_base_score:>6.5

cvss3_base_score:<=3.0

cvss3_temporal_score:=10.0

cvss3_temporal_score:<5.0
```

EPSS score

Use `epss_score:<term>` to search the EPSS score. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
epss_score:>0.5

epss_score:<=0.1

epss_score:=0.9
```

Address

Use `address:<term>` to search the address field.

```
address:192.168.0.1
```

Transport

Use `transport:<term>` to search the transport field.

```
transport:tcp

transport:udp
```

Port

Use `port:<term>` to search the port. The term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
port:22

port:443
```

Operating system support status

Use `os_eol_expired:<boolean>` to find vulnerabilities on assets based on whether their operating systems are End of Life (EOL). This field evaluates both the `os_eol` and `os_eol_extended` values and returns only vulnerabilities on assets with expired coverage.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
os_eol_expired:t

os_eol_expired:1

os_eol_expired:no
```

Finding code

Use `finding_code:<term>` to filter by the finding code field, the unique identifier for a finding.

```
finding_code:rz-finding-internet-exposed-database
```

Finding name

Use `finding_name:<text>` to search by finding name.

```
finding_name:"Internet Exposed Database"
```

Finding risk

Search the `Finding Risk` and `Finding Risk Score` fields with either numeric or keyword values. Risk score is an integer from zero through four, where `0` is Info level risk and `4` is Critical risk.

```
finding_risk:"Critical"

finding_risk:2
```

Detection timestamps (first detected at, last detected at)

Use `first_detected_at:<term>` and `last_detected_at:<term>` to search the timestamp fields `first_detected_at` and `last_detected_at`. The term supports the standard runZero time comparison syntax.

```
first_detected_at:>2weeks

first_detected_at:<30minutes

last_detected_at:>1month

last_detected_at:2hours
```

Publication timestamps (published at)

Use `published_at:<term>` to search the `published_at` timestamp field. The term supports the standard runZero time comparison syntax.

```
published_at:>2weeks

published_at:<30minutes
```

Timestamps (created at, updated at)

Use the following keywords to search the vulnerability timestamp fields (`created_at`, `updated_at`, `suppressed_at`):

- `created_at:<term>`
- `updated_at:<term>`
- `suppressed_at:<term>`

The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks

created_at:<30minutes

updated_at:>1month

updated_at:2hours

suppressed_at:<1day
```

Attributes

Use `<attribute>:<term>` to search any vulnerability attribute. This search term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

If the attribute name conflicts with an existing term, add the `_vulnerability.` prefix to disambiguate the query.

```
plugin.hasPatch:true

severityID:3

_vulnerability.state:REOPENED
```

Use the `has:<term>` keyword to check whether a vulnerability has an attribute defined. Invert the `has` keyword with `not has:<term>` to find missing fields.

```
has:plugin.vpr.score

not has:output
```

Certificate fields

Use the following keywords to search certificate-related fields.

Certificate ID

Use `certificate_id:<uuid>` to filter by certificate ID.

```
certificate_id:4e3a2b1c-5d6f-7a8b-9c0d-1e2f3a4b5c6d
```

Certificate type

Use `certificate_type:<term>` to search by certificate type.

```
certificate_type:"x509"
```

Certificate serial

Use `certificate_serial:<term>` to search by certificate serial number.

```
certificate_serial:"01A23B45C"
```

Certificate public key

Use `certificate_public_key:<term>` to search by certificate public key.

```
certificate_public_key:"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A..."
```

Certificate public key algorithm

Use `certificate_public_key_algorithm:<term>` to search by certificate public key algorithm.

```
certificate_public_key_algorithm:"ecPublicKey"
```

Certificate signature

Use `certificate_signature:<term>` to search by certificate signature.

```
certificate_signature:"abcdef1234567890"
```

Certificate signature algorithm

Use `certificate_signature_algorithm:<term>` to search by certificate signature algorithm.

```
certificate_signature_algorithm:"rsaEncryption"
```

Certificate fingerprint (bkhash)

Use `certificate_fp_bkhash:<term>` to search by certificate fingerprint (bkhash).

```
certificate_fp_bkhash:"d2c7e8f9..."
```

Certificate fingerprint (SHA1)

Use `certificate_fp_sha1:<term>` to search by certificate SHA1 fingerprint.

```
certificate_fp_sha1:"349eb7db55bbad9d7deabc5effdfd5521ed984a0"
```

Certificate fingerprint (SHA256)

Use `certificate_fp_sha256:<term>` to search by certificate SHA256 fingerprint.

```
certificate_fp_sha256:"SHA256:Sc0pLUCvlNaGtu4Xy2fOTe1A6cC+KyU3x7xuN8+aLtA="
```

Certificate subject

Use `certificate_subject:<term>` to search by certificate subject.

```
certificate_subject:"CN=example.com,O=ExampleCorp,C=US"
```

Certificate common name (CN)

Use `certificate_cn:<term>` to search by certificate common name.

```
certificate_cn:"example.com"
```

Certificate issuer

Use `certificate_issuer:<term>` to search by certificate issuing authority.

```
certificate_issuer:"CN=Example CA,O=Example Corp,C=US"
```

Certificate subject key ID

Use `certificate_subject_key_id:<term>` to search by certificate subject key ID.

```
certificate_subject_key_id:"1234567890abcdef"
```

Certificate authority key ID

Use `certificate_authority_key_id:<term>` to search by certificate issuing authority key ID.

```
certificate_authority_key_id:"abcdef1234567890"
```

Suppression

Use `suppressed:<term>` to search vulnerabilities by their suppressed or unsuppressed state.

```
suppressed:t
```

matches suppressed vulnerabilities.

```
suppressed:f
```

matches unsuppressed vulnerabilities.

The special value `any` displays all records, regardless of suppression settings:

```
suppressed:any
```

displays both suppressed and unsuppressed vulnerabilities.

To find the user who performed the suppression, use:

```
suppressed_by:<username>
```

Certificate inventory

When viewing certificates, use these keywords to search and filter. To learn how runZero collects certificates, see Certificates.

General certificate fields

Certificate ID

Use `id:<uuid>` to filter by the ID field, the certificate's unique identifier written as a UUID.

```
id:21e5252d-a6a5-467e-83ed-683657412dff
```

Certificate type

Use `type:<text>` to search for certificates by type.

```
type:x509
```

Name

Use `name:<text>` to search for certificates by name.

```
name:example.com
```

Validity

Use `valid_from:<time>` and `valid_until:<time>` to search for certificates by when they are valid.

```
valid_from:>2025-01-01
```

```
valid_until:<2026-01-01
```

Public key

Use `public_key:<text>` to search for certificates by public key.

```
public_key:MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDJ899KGnqHjVuBekYqosp2l8zWbiyu2I62CzaqaouLtqn1nXaQLMdruhlNN9ShCPfCM2JA
```

Public key algorithm

Use `pk_algorithm:<text>` or `public_key_algorithm:<text>` to search for certificates by public key algorithm.

```
pk_algorithm:rsaEncryption
```

Public key size

Use `pk_size:<number>` to search for certificates by public key size. You will usually want to specify the public key algorithm too, since different algorithms have different key size ranges.

```
pk_algorithm:rsaEncryption and pk_size:<2048
```

RSA modulus

Use `rsa_modulus:<number>` to search for certificates using RSA encryption by modulus.

```
rsa_modulus:24103124269210325885520760221975660748569505485024599426541169488887185854621
```

RSA exponent

Use `rsa_exponent:<number>` to search for certificates using RSA encryption by exponent.

```
rsa_exponent:65537
```

DSA p

Use `dsa_p:<number>` to search for certificates using DSA encryption by parameter p.

```
dsa_p:177153854627485855237091799251665123928681135467044234987602313329516356048326341718397044415619278113858376637095
```

DSA q

Use `dsa_q:<number>` to search for certificates using DSA encryption by parameter q.

```
dsa_q:2388256134509873010646676 7...
```

DSA g

Use `dsa_g:<number>` to search for certificates using DSA encryption by parameter g.

```
dsa_g:115061965280659099184804528749800834868693002715550910755304058605824868089359405648453267489218080170766391220380
```

ECDSA curve

Use `ecdsa_curve:<text>` to search for certificates using ECDSA encryption by curve.

```
ecdsa_curve:P-256
```

Insecure public key

Use `public_key_insecure:true` to search for certificates with insecure public key algorithm and key size combinations.

Signature

Use `signature:<text>` to search for certificates by signature.

```
signature:Bden73ipj8B2xb1Ozy5nOvIytCktGrht5xL7ZfFlaLIBQxbGO5Iuf6Y1yICcEpYqsgSJS6JKCdw5dujmPmGRwBZfVhIbSRb0exFQ4BVp82WtD
```

Signature algorithm

Use `sig_algorithm:<text>` or `signature_algorithm:<text>` to search for certificates by signature algorithm.

```
sig_algorithm:sha512WithRSAEncryption
```

Insecure signature

Use `signature_algorithm_insecure:true` to search for certificates with insecure signatures.

Self-signed

Use `self_signed:true` to search for self-signed certificates.

Certificate authority

Use `is_ca:true` to search for certificates that are certificate authorities (CAs).

Subject

Use `subject:<text>` to search for certificates by X.509 subject DN.

```
subject:"CN=Server Name/O=Company Name"
```

Common name

Use `cn:<name>` to search for certificates by common name. This is equivalent to searching the subject DN for just the CN field.

```
cn:"Server Name"
```

Subject alternative name

Use the following keywords to search the four sets of Subject Alternative Names (SANs):

```
san_dns_name:example.com
san_ip_address:10.0.1.23
san_email_address:postmaster@example.com
san_uri:https://example.com
```

Issuer

Use `issuer:<text>` to search for certificates by X.509 issuer DN.

```
issuer:"CN=Certificate Authority Name"
```

Subject key ID

Use `subject_key_id:<text>` to search by X.509 subject key ID. Values are accepted with or without colons.

```
subject_key_id:"12:90:EF:DD:E1:27:A4:47:3E:32:57:AF:44:75:92:8E:8C:C2:0A:C0"
subject_key_id:1290EFDDE127A4473E3257AF4475928E8CC20AC0
```

Authority key ID

Use `authority_key_id:<text>` to search by X.509 authority key ID. Values are accepted with or without colons.

```
authority_key_id:"12:90:EF:DD:E1:27:A4:47:3E:32:57:AF:44:75:92:8E:8C:C2:0A:C0"
authority_key_id:1290EFDDE127A4473E3257AF4475928E8CC20AC0
```

OCSP server

Use `ocsp_server:<text>` to search for certificates by OCSP server.

```
ocsp_server:http://ocsp.test.com
```

CRL distribution point

Use `crl_distribution_points:<text>` or `crl_distribution_point:<text>` to search for certificates by CRL distribution point.

```
crl_distribution_points:http://x1.c.lencr.org/
```

Issuing certificate URL

Use `issuing_certificate_url:<text>` to search for certificates by issuing certificate URL.

```
issuing_certificate_url:http://x1.i.lencr.org/
```

Hash

Use these keywords to find certificates by their MD5, SHA1, SHA256, or BK hash values:

```
md5:<hash value>
sha1:<hash value>
sha256:<hash value>
bkhash:<hash value>
```

Serial number

Use `serial_number:<text>` to search for certificates by serial number.

```
serial_number:123456
```

Timestamps

Use the following keywords to search the certificate inventory timestamp fields:

- `created_at:<term>`
- `updated_at:<term>`
- `last_seen:<term>`

The term supports the standard runZero time comparison syntax, for example:

```
last_seen:<1week

last_seen:<2months

last_seen:<1year
```

Associate services

Use `associate_services:<number>` or `service_count:<number>` to search for certificates by the number of associated services.

```
associated_services:>10
```

Hidden certificates

Use `hidden:true` to search for certificates hidden from the inventory.

Version

Use `version:<text>` to search for certificates by version, such as x509 version.

```
version:3
```

Tags

Use `tag:<term>` to search tags added to a certificate. The term is the tag name, or the tag name followed by an equal sign and the tag value. Tag value matches must be exact.

```
tag:"group"

tag:"group=production"
```

Comments

Use `comment:<text>` to search comments on a certificate.

```
comment:"contractor laptop"
```

```
comment:"imaging server"
```

Vulnerability name

Use `vulnerability_name:<text>` to search for certificates by associated vulnerability name.

```
vulnerability_name:"Expired Certificate On TLS Service"
```

Vulnerability count

Use `vulnerability_count:<number>` to search for certificates by the number of associated vulnerabilities. This field supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
vulnerability_count:>2
```

Has vulnerability

Use `has_vulnerability:<boolean>` to search for certificates that have (or do not have) associated vulnerabilities.

```
has_vulnerability:true
```

Wireless inventory

When viewing WiFi networks, use these keywords to search and filter.

SSID and ESSID

Use `ssid:<text>` to search the SSID/ESSID field.

```
ssid:"Guest Network"
```

```
ssid:"Corporate"
```

BSSID (MAC)

Use `bssid:<text>` or `mac:<text>` to search the BSSID field.

```
bssid:"00:01:02:03:04:05"
```

```
mac:"00:01:%"
```

Vendor

Use `mac_vendor:<text>` to search the vendor field.

```
mac_vendor:"Google"
```

```
mac_vendor:"Netgear"
```

```
mac_vendor:"Cisco"
```

Family

Use `family:<term>` to search the family field.

```
family:"010304"
```

Channels

Use `channel:<term>` to search the channels field.

```
channel:"11"
```

Type

Use `type:<text>` to search the network type field.

```
type:"infrastructure"
```

Interface

Use `interface:<text>` to search the network interface field.

```
interface:"wlan0"
```

Encryption

Use `encryption:<term>` to search the encryption field.

```
encryption:"aes"
```

```
encryption:"none"
```

Authentication

Use `authentication:<term>` to search the authentication field.

```
authentication:"wpa2-psk"
```

```
authentication:"open"
```

Timestamps

Use `first_seen:<term>`, `last_seen:<term>`, and `created_at:<term>` to search the timestamp fields. The term supports the standard runZero time comparison syntax.

```
first_seen:<30seconds
```

```
first_seen:>2019-08-01
```

```
last_seen:<1week

last_seen:<2months

created_at:>2weeks

created_at:<30minutes
```

Signal

Use `signal:<number>` or `sig:<number>` to search the signal field. The term can include the operators `>`, `=`, `<=`, and `=`. The default operator is `=`.

```
signal:">75"

signal:"<=25"

signal:99
```

Organization name or ID

Use `organization:<term>` to filter by organization name or ID.

```
organization:runZero

organization:"Temporary Project"

organization:f1c3ef6d-cb41-4d55-8887-6ed3cfb3d42d
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary

site:"Branch Office"

site:ad67d649-041b-439d-af59-f200053a8899
```

Explorer name or ID

Use `explorer:<term>` to filter by Explorer name or ID.

```
explorer:DESKTOP-AB451F

explorer:8b927a8e-d405-40e9-aa47-d6afc9bff237
```

Wireless ID

Use `id:<uuid>` to search the ID field, the wireless network's unique identifier written as a UUID.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Last task ID

Use `task:<uuid>` to search the Last Task ID field, the UUID of the task that most recently reported the wireless network.

```
task:39ab0e71-3cf1-4176-b6b0-4ed495288229
```

Wireless attributes

Use `<attribute>:<term>` to search any wireless attribute.

```
radio_type:"802.11n"
```

Users inventory

When viewing the Users inventory, use these keywords to search and filter users. To search groups, see the groups inventory keywords.

Source

Use `source:<name>` to search or filter by the name of the source that reported the users.

```
source:ldap
```

Name fields

Use `<name_field>:<text>` to search or filter any of the name fields in the user attributes.

The name fields you can search this way:

- `name`
- `display_name`
- `first_name`
- `last_name`

```
name:j
```

```
display_name:"john doe"
```

```
first_name:john
```

```
last_name:doe
```

Description

Use `description:<text>` to search the description field.

```
description:shared
```

Groups

Use `group_id:<uuid>` to find users that belong to a group by the group's ID. The `group_id` field is the group's unique identifier, written as a UUID.

```
group_id:f8a26321-cbce-4fb5-a9ca-ffa809489dd5
```

Email

Use `email:<address>` to search for users by email address.

```
email:john@example.com
```

Phone

Use `phone:<phone number>` to search for users by phone number.

```
phone:888-555-1234
```

Title

Use `title:<text>` to search for users by title.

```
title:CISO
```

Location

Use `location:<text>` to search for users by location.

```
location:TX
```

Last logon

Use `last_logon_at:<date>` to search for users by last logon time. This field supports the date comparison operators.

```
last_logon_at:<1year
```

First seen

Use `first_seen_at:<date>` to search for users by when they were first seen in the organization. This field supports the date comparison operators.

```
first_seen_at:<1day
```

Last seen

Use `last_seen_at:<date>` to search for users by when they were last seen in the organization. This field supports the date comparison operators.

```
last_seen_at:>2years
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary
```

```
site:"Branch Office"
```

```
site:ad67d649-041b-439d-af59-f200053a8899
```

Organization name or ID

Use `organization:<term>` to filter by organization name or ID.

```
organization:runZero
```

```
organization:"Temporary Project"
```

```
organization:f1c3ef6d-cb41-4d55-8887-6ed3cfb3d42d
```

Groups inventory

When viewing the Groups inventory, use these keywords to search and filter groups. To search users, see the users inventory keywords.

Source

Use `source:<name>` to search or filter by the name of the source that reported the groups.

```
source:ldap
```

Name fields

Use `<name_field>:<text>` to search or filter either of the two name fields in the group attributes.

The name fields you can search this way:

- `name`
- `display_name`

```
name:admin
```

```
display_name:"Domain Administrators"
```

Description

Use `description:<text>` to search the description field.

```
description:admin
```

Users

Use `user_id:<uuid>` to find groups with a specific member by the user's ID. The `user_id` field is the user's unique identifier, written as a UUID.

```
user_id:f8a26321-cbce-4fb5-a9ca-ffa809489dd5
```

Email

Use `email:<address>` to search for users by email address.

```
email:john@example.com
```

First seen

Use `first_seen_at:<date>` to search for users by when they were first seen in the organization. This field supports the date comparison operators.

```
first_seen_at:<1day
```

Last seen

Use `last_seen_at:<date>` to search for users by when they were last seen in the organization. This field supports the date comparison operators.

```
last_seen_at:>2years
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary
```

```
site:"Branch Office"
```

```
site:ad67d649-041b-439d-af59-f200053a8899
```

Organization name or ID

Use `organization:<term>` to filter by organization name or ID.

```
organization:runZero
```

```
organization:"Temporary Project"
```

```
organization:f1c3ef6d-cb41-4d55-8887-6ed3cfb3d42d
```

Interface keywords

Query and filter the data across your runZero account with the search syntax and each interface's keywords. Keywords and example values are documented for these sections of the runZero Console:

- Scan templates
- Findings
- Tasks
- Analysis reports
- Explorers
- runZero users and groups
- Sites and organizations
- Credentials
- Queries
- Events

Scan templates

When viewing scan templates, use these keywords to search and filter.

ID

Use `id:<uuid>` to filter by the ID field, the template's unique identifier written as a UUID.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Name

Use `name:<text>` to search by scan template name.

```
name:WiFi
```

```
name:"Data Center"
```

Timestamps

Use the following keywords to search the scan template timestamp fields (`created_at`, `updated_at`):

- `created_at:<term>`
- `updated_at:<term>`

The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks
```

```
created_at:<30minutes
```

```
updated_at:>1year
```

```
updated_at:<12hours
```

Created by

Use `created_by_email:<term>` to search by the email address of the user who created the template.

```
created_by_email:user@example.com
```

Findings

When viewing findings, use these keywords to search and filter.

Finding code

Use `finding_code:<uuid>` to filter by the finding code field, the unique identifier for a finding.

```
finding_code:rz-finding-internet-exposed-database
```

Name

Use `name:<text>` to search by finding name.

```
name:"Internet Exposed Database"
```

Description

Use `description:<text>` to search the `Description` field.

```
description:"indicated databases"
```

Solution

Use `solution:<text>` to search the `Solution` field.

```
solution:"indicated databases"
```

Risk

Search the `Risk` / `Risk Rank` value with either numeric or keyword values. Risk rank is an integer from zero through four, where `0` is Info level risk and `4` is Critical risk.

```
risk:"Critical"
```

```
risk_rank:>2
```

Category

Use `category:<text>` to search the finding `Category` field.

```
category:"End-of-Life"
```

Vulnerability instance count

Use `vulnerability_count:<text>` to search the `Instance` field.

```
vulnerability_count:>0
```

Organization and site names

Use these terms to search the names of affected organizations or sites:

- `organization_name:<text>`
- `site_name:<text>`

The IDs are unique and are written as UUIDs.

```
organization_id:0eacf412-6e69-11ec-88b9-f875a414a63a
```

Organization and site IDs

Use these terms to search the IDs of affected organizations or sites:

- `organization_id:<uuid>`
- `site_id:<uuid>`

The IDs are unique and are written as UUIDs.

```
organization_id:0eacf412-6e69-11ec-88b9-f875a414a63a
```

Timestamps

Use the following keywords to search the finding timestamp fields (`last_detected_at`, `created_at`, `updated_at`, `suppressed_at`):

- `created_at:<term>`
- `updated_at:<term>`
- `last_detected_at:<term>`
- `suppressed_at:<term>`

The term supports the standard runZero time comparison syntax.

```
last_detected_at:>2weeks

created_at:<30minutes

updated_at:>1year
```

`updated_at:<12hours`

`suppressed_at:<1day`

Suppression

Use `suppressed:<term>` to search findings by their suppressed or unsuppressed state.

`suppressed:t`

matches suppressed findings.

`suppressed:f`

matches unsuppressed findings.

The special value `any` displays all records, regardless of suppression settings:

`suppressed:any`

displays both suppressed and unsuppressed findings.

To find the user who performed the suppression, use:

`suppressed_by:<username>`

Tasks

When viewing all tasks, use these keywords to search and filter them.

Name

Search the Name field with `name:<text>`.

`name:"test scan"`

Description

Search the Description field with `description:<text>`.

`description:"full scan"`

Created by

Search the `Created By` field with `created_by:<term>`.

`created_by:"admin"`

Type

Search the task type with `type:<text>`.

```
type:scan
```

Status

Search the task status with `status:<text>`.

```
status:error
```

Error

Search the task error message with `error:<text>`.

```
error:"no disk space"
```

Recurrence frequency

Search the recurrence frequency (the “Freq” column) with `recur_frequency:<text>` or `freq:<text>`. To find tasks by whether they recur at all, use `recurring:<boolean>` or `recur:<boolean>`.

```
recur_frequency:hourly
```

```
freq:daily
```

```
freq:continuous
```

```
recur:true
```

For tasks with a frequency of `Nth Weekday of Month`, a query such as `freq:nth_weekday,2 freq:monday` finds tasks that repeat on the second Monday of each month.

Timestamps (created at, updated at)

Search the timestamp fields, `created_at` and `updated_at`, with `created_at:<term>` and `updated_at:<term>`. The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks
```

```
created_at:<30minutes
```

```
updated_at:>1month
```

```
updated_at:2hours
```

Next/last run time

Search by last and next recurrence with `recur_last:<term>` and `recur_next:<term>`. The term supports the standard runZero time comparison syntax.

```
recur_last:<2hours

recur_next:>1day
```

Start time

Search by start time with `start_time:<term>`. The term supports the standard runZero time comparison syntax.

```
start_time:<2hour
```

Grace period

Search the grace period with `grace_period:<term>` or just `grace:<term>`. The term supports the standard runZero time comparison syntax.

```
grace:<2hour
```

Site name or ID

Use `site:<term>` to filter by site name or ID.

```
site:Primary

site:"Branch Office"

site:ad67d649-041b-439d-af59-f200053a8899
```

Template ID

Use `template_id:<term>` to filter by scan template ID.

```
template_id:de657459-041b-439d-af59-ff1f153a7722
```

Source

Search the task data source with `source:<text>` or `source_id:<number>`.

```
source:censys
```

Sources are:

ID	Name	Description
1	`runzero`	runZero
2	`miradore`	Miradore
3	`aws`	Amazon Web Services
4	`crowdstrike`	CrowdStrike

5	`azure`	Microsoft Azure
6	`censys`	Censys
7	`vmware`	VMWare
8	`gcp`	Google Cloud Platform
9	`sentinelone`	SentinelOne
10	`tenable`	Tenable.io & Nessus
12	`rapid7`	Rapid7 Nexpose & InsightVM
14	`qualys`	Qualys VMDR
15	`shodan`	Shodan
16	`azuread`	Azure AD
17	`ldap`	Active Directory (LDAP)
18	`ms365defender`	Microsoft 365 Defender
19	`intune`	Microsoft Intune
20	`googleworkspace`	Google Workspace
21	`sample`	runZero traffic sampling
22	`tenablesecuritycenter`	Tenable Security Center
23	`packet`	runZero packet capture import
24	`wiz`	Wiz

Credential ID

Find tasks that use a specific set of credentials with `credential_id:<id>`.

```
credential_id:d7931a68-6e56-11ec-ad72-f875a414a63a
```

Parameters

Search task parameters with `params:<text>`. This can be useful for finding scan tasks that had specific probes enabled.

```
params:bacnet
```

Asset counts

Search completed tasks by the asset counts in their results. The search terms are:

- New assets: `assets_new:<number>`
- Assets back online: `assets_back_online:<number>`
- Assets marked offline: `assets_marked_offline:<number>`

- Assets changed: `assets_changed:<number>`
- Assets unchanged: `assets_unchanged:<number>`
- Assets ignored: `assets_ignored:<number>`
- Assets updated: `assets_updated:<number>`

These terms support numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
assets_new:>0
```

```
assets_unchanged:>=1
```

Analysis reports

When viewing generated analysis reports, use these keywords to search and filter them.

Name

Search the Name field with `name:<text>`.

```
name:"main"
```

Description

Search the Description field with `description:<text>`.

```
description:"compare secondary"
```

Type

Search the report type with `type:<text>`.

```
type:outliers
```

Report ID

The ID field is the unique identifier for an analysis report, written as a UUID. Search it with `id:<uuid>`.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Timestamps

To search by when a report was generated, use `created_at:`. The term supports the standard runZero time comparison syntax.

```
created_at:>2019-08-01
```

```
created_at:<1week
```

Created by

Search the `Created By` field with `created_by:<term>`.

```
created_by:jsmith
```

Explorers

When viewing deployed Explorers, use these keywords to search and filter them.

Name

Search the Name field with `name:<text>`.

```
name:"main"
```

Site

Search the site with `site:<text>`.

```
site:Primary
```

Up

Search by whether the Explorer is up with `up:<boolean>`.

```
up:true
```

Address

Use `address:<IP address>` to search the IP address(es) the Explorer is deployed on.

```
address:10.0.1.200
```

Version

Search the Explorer software version with `version:<text>`.

```
version:2.9.7
```

Npcap version

Search the npcap library version on Windows Explorers with `npcap_version:<text>`.

```
npcap_version:1.60
```

Architecture

Use `architecture:<text>` to search the machine architecture Explorers are deployed on.

```
architecture:amd64
```

OS

Use `os:<text>` to search the operating system Explorers are deployed on. macOS is recorded as `darwin`, its underlying Unix core.

```
os:windows
```

```
os:darwin
```

Capability

Search Explorer capabilities with `capability:<keyword>`. The two keywords are:

- `screenshot` for Explorers that can screenshot web pages
- `ec2` for Explorers that can describe AWS EC2 instances

Example:

```
capability:screenshot
```

Tags

Use `tag:<term>` to search tags added to an Explorer. The term is the tag name, or the tag name followed by an equal sign and the tag value. Tag value matches must be exact.

```
tag:"admin"
```

```
tag:"group=cloud"
```

runZero users and groups

User search keywords

When viewing users, use these keywords to search and filter them.

Email

Use `email:<address>` to find someone by email address.

```
email:john@example.com
```

Name

Use `name:<text>` to find someone by name.

```
name:john
```

```
name:"John Smith"
```

Superuser

Use `superuser:<boolean>` to find people by whether they have superuser access.

```
superuser:true
```

```
superuser:f
```

Access

Use `access:<term>` to find users with a specific access level. The access levels are `admin`, `user`, `annotator`, `viewer`, `billing` and `none`.

```
access:admin
```

Status

Use `status:<text>` to find users by invitation status. The status values are `activated`, `pending` and `expired`.

```
status:pending
```

SSO

Use `sso:<boolean>` to find people by whether they can only sign in via SSO.

```
sso:true
```

MFA

Use `mfa:<boolean>` to find people by whether they have enrolled an MFA token.

```
mfa:f
```

Group ID

The `group_id` field is the unique identifier for a group, written as a UUID. Use `group_id:<uuid>` to find the members of a group by its ID.

```
group_id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Group name

Use `group_name:<text>` to find the members of a group by its name.

```
group_name:administrators

group_name:"Temp annotators"
```

User groups

Use `has_group:<boolean>` to find users by whether they belong to a group. The value `t` or `true` shows all users who are members of a group.

```
has_group:t
```

The value `f` or `false` shows users who are not members of a group.

```
has_group:f
```

Group count

Use `group_count:<number>` to search by group membership count. This term supports numerical comparison operators (`>`, `>=`, `<`, `<=`, `=`).

```
group_count:>1

group_count:=0
```

Group search keywords

When viewing your groups, use these keywords to search and filter them.

ID

The ID field is the unique identifier for a group, written as a UUID. Use `id:<uuid>` to filter by ID.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Name

Use `name:<text>` to search by group name.

```
name:administrators

name:"Temp annotators"
```

Access

Use `access:<term>` to find groups with a specific access level. The access levels are `admin`, `user`, `annotator`, `viewer`, `billing` and `none`.

```
access:admin
```

Timestamps (created at, updated at)

Filter groups by their timestamp fields, `created_at` and `updated_at`, with `created_at:<term>` and `updated_at:<term>`. The terms support the standard runZero time comparison syntax.

```
created_at:<30days
```

```
updated_at:<1week
```

Expiration

Filter groups by their expiration timestamp, `expires_at`, with `expires_at:<term>`. The term supports the standard runZero time comparison syntax.

```
expires_at:<30days
```

```
expires_at:>8/1/2019
```

The `expired` property records whether a group has expired. Search it with `expired:<boolean>`.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
expired:true
```

```
expired:0
```

Use `has_expiration:<term>` to find groups that have an expiration date.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
has_expiration:true
```

```
has_expiration:0
```

Email

The `created_by_email` property holds the email address of the user who created the group. Search it with `created_by_email:<term>`.

```
created_by_email:user@runzero.com
```

Group mapping search keywords

When viewing your SSO group mappings, use these keywords to search and filter them.

ID

The ID field is the unique identifier for a group mapping, written as a UUID. Use `id:<uuid>` to filter by ID.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

SSO attribute

The `sso_attribute` field is the name of the attribute checked for matching values. Use `sso_attribute:<text>` to search by `sso_attribute`.

```
sso_attribute:department
```

SSO value

The `sso_value` field is the value, or comma-separated list of values, to match. Use `sso_value:<text>` to search by `sso_value`.

```
sso_value:security
```

```
sso_value:"admins, administrators"
```

Group ID

The `group_id` field is the unique identifier for a group, written as a UUID. Use `group_id:<uuid>` to find the mappings for a group by its ID.

```
group_id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Group name

Use `group_name:<text>` to find the mappings for a group by its name.

```
group_name:administrators
```

```
group_name:"Temp annotators"
```

Timestamps (created at, updated at)

Filter group mappings by their timestamp fields, `created_at` and `updated_at`, with `created_at:<term>` and `updated_at:<term>`. The terms support the standard runZero time comparison syntax.

```
created_at:<30days
```

```
updated_at:<1week
```

Email

The `created_by_email` property holds the email address of the user who created the group. Search it with `created_by_email:<term>`.

```
created_by_email:user@runzero.com
```

Sites and organizations

Site search keywords

When viewing sites, use these keywords to search and filter them.

Name

Search the Name field with `name:<text>`.

```
name:"Primary"
```

Description

Search the Description field with `description:<text>`.

```
description:"wireless"
```

```
description:"vlan 50"
```

Scope

Search the Scope field with `scope:<term>`.

```
scope:"10.10.10."
```

Excludes

Search the Excludes field with `excludes:<term>`.

```
excludes:"192.168.0."
```

Timestamps (created at, updated at)

Search the timestamp fields (`created_at`, `updated_at`) with `created_at:<term>` and `updated_at:<term>`. The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks
```

```
created_at:<30minutes
```

```
updated_at:>1month
```

```
updated_at:2hours
```

Organization search keywords

Name

Search the Name field with `name:<text>`.

```
name:"main"
```

Description

Search the Description field with `description:<text>`.

```
description:"branch office"
```

```
description:"pci"
```

Timestamps (created at, updated at)

Search the timestamp fields (`created_at`, `updated_at`) with `created_at:<term>` and `updated_at:<term>`. The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks
```

```
created_at:<30minutes
```

```
updated_at:>1month
```

```
updated_at:2hours
```

Credentials

When viewing saved credentials, use these keywords to search and filter them.

Credential fields

Credential ID

The ID field is the unique identifier for a credential, written as a UUID. Search it with `id:<uuid>`.

```
id:cdb084f9-4811-445c-8ea1-3ea9cf88d536
```

Credential name

Search the credential name with `name:<text>`.

```
name:"AWS read-only account"
```

```
name:"Miradore API key"
```

Credential type

Search the credential type with `type:<text>`.

```
type:aws_access_secret

type:miradore_api_key_v1
```

Credential global property

The global property shows whether all organizations have access to a credential. Search it with `global:<boolean>`.

The term is a boolean value:

- `true`, `t`, `1`, and `yes` represent *true*
- `false`, `f`, `0`, and `no` represent *false*

```
global:true

global:0
```

Credential timestamps

Search the credential timestamp fields (`created_at` and `last_used_at`) with:

- `created_at:<term>`
- `last_used_at:<term>`

The term supports the standard runZero time comparison syntax.

```
created_at:<3days

created_at:>2019-08-01

created_at:>8/1/2019

created_at:<1week

created_at:<2months

last_used_at:<1year

last_used_at:>2weeks

last_used_at:<30minutes

last_used_at:>1year

last_used_at:<12hours

last_used_at:0
```

Credential created by

The created_by_email field holds the email address of the user who created the credential. Search it with `created_by_email:<term>`.

```
created_by_email:user@example.com
```

Queries

When viewing saved queries, use these keywords to search and filter them.

Name

Search the `Name` field with `name:<text>`.

```
name:"smb2"
```

Description

Search the `Description` field with `description:<text>`.

```
description:"smb version 1"
```

```
description:"wep"
```

Type

Search the `Type` field with `type:<term>`.

```
type:"services"
```

Category

Search the `Category` field with `category:<term>`.

```
category:"security"
```

```
category:"audit"
```

Severity

Search the `Severity` field with `severity:<term>`.

```
severity:"info"
```

```
severity:"critical"
```

Created by

Search the `Created By` field with `created_by:<term>`.

```
created_by:"runzero"
```

Timestamps (created at, updated at)

Search the timestamp fields, `created_at` and `updated_at`, with `created_at:<term>` and `updated_at:<term>`. The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks

created_at:<30minutes

updated_at:>1month

updated_at:2hours
```

Events

When viewing system events under alerts, use these keywords to search and filter them.

runZero retains event records for one year.

Action

Use `action:<text>` to search by the action that caused the event.

```
action:agent-reconnected
```

Created timestamp

Search the `created_at` timestamp field with `created_at:<term>`. The term supports the standard runZero time comparison syntax.

```
created_at:>2weeks

created_at:<30minutes

updated_at:>1month

updated_at:2hours
```

Details

Search the event record details with `details:<text>`. This can be useful for finding IP addresses.

```
details:192.168.0.1
```

Source and target name

Search the source (src) column with `src:<text>` or `source:<text>`, and the target (tgt) column with `tgt:<text>` or `target:<text>`.

```
src:crowdstrike
```

```
target:primary
```

Source and target type

Search the source type (shown at the start of the src column) with `src_type:<text>` or `source_type:<text>`, and the target type with `tgt_type:<text>` or `target_type:<text>`.

```
src_type:task
```

```
target_type:site
```

Organization, site, source and target IDs

Search the IDs of organizations, sites, sources and targets mentioned in event details with these terms:

- `organization_id:<uuid>`
- `site_id:<uuid>`
- `source_id:<uuid>` or `src_id:<uuid>`
- `target_id:<uuid>` or `tgt_id:<uuid>`

The IDs are unique and are written as UUIDs.

```
organization_id:0eacf412-6e69-11ec-88b9-f875a414a63a
```

Automating queries

With runZero's query language, you can search and filter your asset inventory by asset fields and values. runZero includes a query library of prebuilt searches, which you can browse from the Queries page and run after a scan to investigate discovery findings.

You can also use the same search syntax to track and monitor events across your network, based on any combination of fields. Save your custom queries to reuse them. Review the query syntax documentation for a refresher on how to build a query, or adapt one of the query examples.

To run a query after every scan, turn on its **Auto** option. After the query runs, its results appear in the Queries table.

In the Queries table, a checkmark in the *Auto* column means the query is active. The *Matches* column next to it shows the number of results; click that number to view them. A query with no matches shows 0.

Turn on automatic search queries

1. Select Queries from the left navigator.
2. Open or create the query you want to turn on.
3. Turn on the **Automatically track query results on the dashboard** option.
4. Save the query.

Data export

Exporting asset data

You can export asset data from the inventory view in a few ways. The `Export` menu offers Export All options in CSV and JSON format, and once you enter a search query, options to export just the search results in either format.

The CSV format opens in tools like Microsoft Excel and imports into other applications, but it omits the full details of some fields, such as Services. The JSON format is a complete export but may need extra processing for other tools.

If you have a specific export/import scenario in mind, contact support and let us know. The Export API uses both formats and accepts any search query in the same syntax as the inventory. For HP iLO servers, see Exporting HP iLO data.

Exporting HP iLO data

HP Integrated Lights-Out (iLO) provides remote management, configuration, and monitoring for HP servers. It centralizes operations for your server environments and simplifies tasks like rebooting servers, booting into single user mode, and bypassing authentication.

Finding the serial number of your HP iLO devices helps you track warranties for support and contract management. With runZero Platform, you can export the runZero data as a CSV to feed into warranty tracking tools.

How to export HP iLO CSV data

In your runZero Console, go to your inventory. From the **Export** menu, choose the **HP iLO CSV format**. This downloads all HP iLO data in the runZero inventory to a CSV file.

To narrow the export, filter the inventory first. For example, to export only iLOs with ProLiant DL360p Gen8 hardware, query:

```
alive:t AND ilo.hardware:"=ProLiant DL360p Gen8"
```

Then, from the Export menu, choose the **HP iLO CSV format** from the Export Search Results submenu. For the other export formats, see Exporting asset data.

HP iLO CSV export data

The exported CSV file contains these fields:

- IP address
- MAC address
- Name
- Serial number
- Product ID
- Model
- Version
- Health status
- IRS
- iLO product name
- Serial number
- Boot block
- HW revision
- FW revision
- PWRM
- Auth local
- Auth Kerberos
- Auth LDAP
- Auth license
- Security message
- Alive

- ID
- Site ID
- Last seen

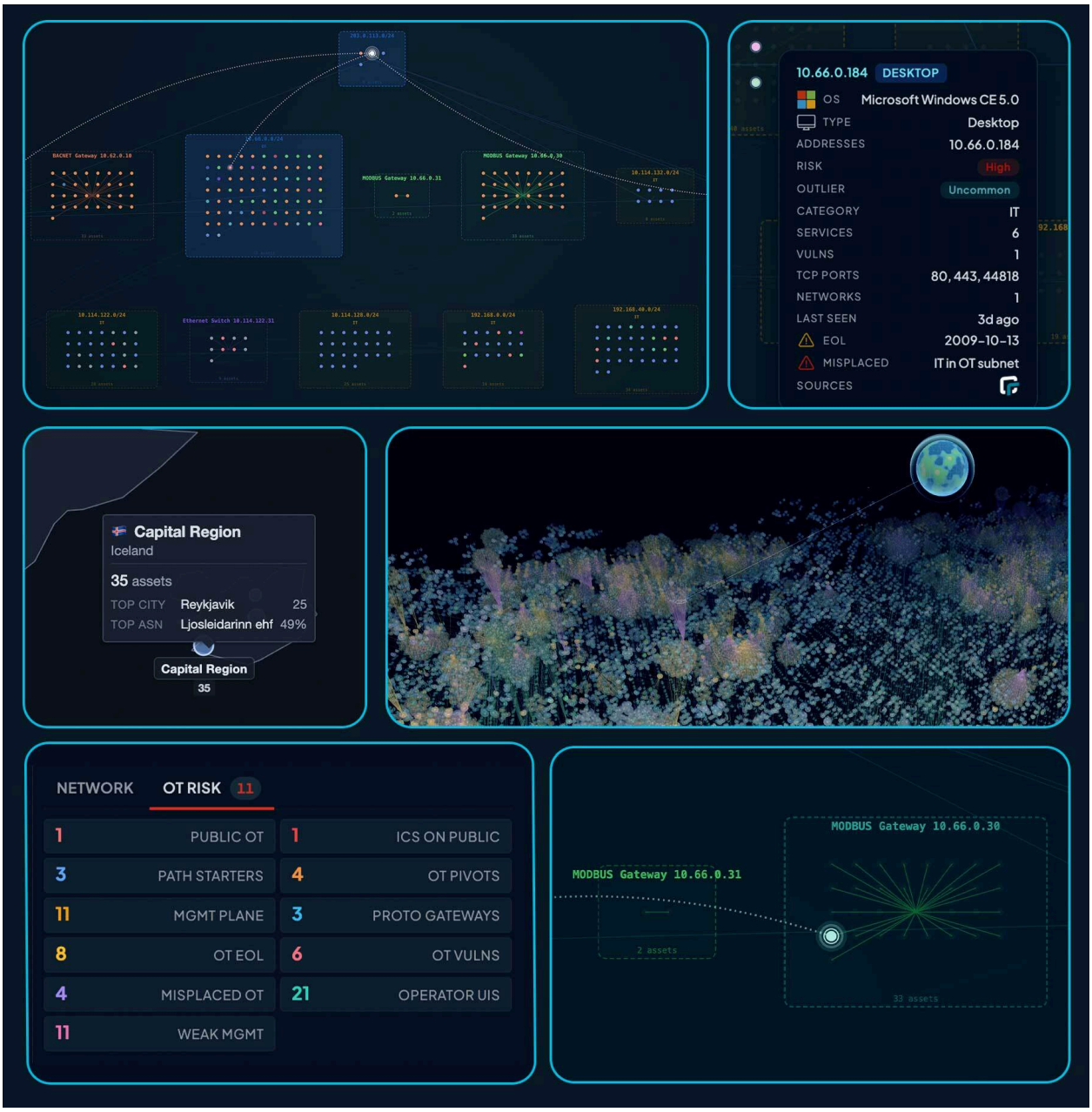
The export holds more data than you may need for your warranty tracker. Serial numbers, physical hardware information, health status, and firmware version may be the most useful fields to import into your tracker.

Advanced reports

Network Map

The Network Map is an interactive Layer-2 / Layer-3 topology of every asset in the current scope. Where the World Map shows assets by physical location, the Network Map shows logical relationships: which subnets contain which assets, which gateways and switches connect them, and which assets pivot between subnets.

The map has two modes, a 2D Canvas view and a 3D rotating globe, that share the same data, search syntax, and trace engine.



Accessing the map

You can open the Network Map from several places in the console:

- In the **Inventory grid**, asset rows show a small map icon when the asset has L2/L3 data. Click the icon to open the map filtered to that asset.
- In **Inventory search**, the “View on map” action above the inventory grid opens the map with the current inventory search applied as the map filter.
- In **Asset details**, the **Manage** menu offers **View on network map** (or **Back to network map** if you arrived from the map). The same menu offers **View internet attack path**,

which pre-loads a trace from the public internet to the selected asset.
- In **Reports**, open it directly at `/reports/network/map` from the Reports menu, or at https://console.runzero.com/reports/network/map.
- In **Queries**, saved query rows in the queries grid include a “View on network map” action that prefills the map with the saved query.

When you open the map from another page, runZero records the previous URL as a return target. A **Back** button in the top-left returns you to that exact view, with your filters and pagination preserved.

2D and 3D modes

- The **2D Network Map** (`/reports/network/map`) is a Canvas-rendered topology with the full feature set: hop-filter trace, choke-point analysis, multi-tab Network Intel card, search results panel, and exports.
- The **3D Network Map** (`/reports/network/map/3d`) is a rotating spherical layout that uses the same data, search syntax, and Source / Target trace engine. It is useful for executive views and demos.

To switch modes, use the **2D** / **3D** button in the toolbar. Your filter, search, and trace state survive the switch.

Subnet layout

The map groups assets into rectangular subnet clusters. Each subnet box contains the assets that have at least one address in that CIDR range, along with any unaddressed (MAC-only) endpoints linked into the subnet by L2 evidence.

Subnet mask

The toolbar mask selector controls how the map aggregates IPv4 addresses into subnets:

- **/24** (default) puts each `/24` in a separate cluster.
- **/16** collapses related `/24`s into the same `/16` parent.
- **/15** through **/8** give broader rollups, useful for large flat networks.

A separate selector controls the IPv6 mask, which defaults to `/56` and ranges from `/32` to `/64` in steps of 4.

When a scanned subnet overlaps a configured site subnet, the map uses the configured CIDR directly. Otherwise it applies the chosen mask to the asset’s address to derive a synthetic subnet boundary.

Multi-homed assets

An asset with IP addresses in more than one subnet appears once in each subnet it belongs to. The renderer draws an L3 link between the copies to show that the asset bridges those segments. Every copy shares the same asset ID, so clicking any copy opens the same asset

detail panel, but each copy has its own position so trace paths through the asset stay visually unambiguous.

The Network Intel card counts multi-homed assets as **Pivots** (`multi_homed:t`).

Public Internet node

A virtual **Public Internet** node is pinned at the top of the layout. Public IP subnets observed during scanning attach to it, and the trace engine accepts it as a source or target for ingress and egress path analysis (see Tracing paths).

By default the map filters out widely-reused public IPs (for example public DNS resolvers like `1.1.1.1` or `8.8.8.8`) that show up on many internal assets and would otherwise produce misleading internet edges. You can change the threshold in the Trace panel's expanded view.

Network Intel card

The **Network Intel** card on the left rail is a clickable stats grid. Each cell shows a derived count for the current view. Click a cell to seed the map's search bar with the matching query and re-run the filter.

Cell	What it counts	Click query
Assets	Total assets in view	(display only)
Subnets	Distinct subnets observed	(display only)
High Risk	Assets with high or critical risk	`risk:high OR risk:critical`
EOL	Assets running an end-of-life OS	`eol:t`
Pivots	Multi-homed assets bridging two or more subnets	`multi_homed:t`
Public	Assets with at least one public IP	`has_public:t`
Misplaced	Assets whose category does not match their subnet's classification	`misplaced:t`
Outliers	Assets that are distinct from their neighbors' norm	`outlier_rank:>1`
Gateways	Protocol gateways bridging IP and non-IP networks	`is:gateway`
Switches	Managed switches that share neighbor information	`is:switch`

The card also includes a device-type legend; clicking a legend cell runs the equivalent `type:` or `category:` query.

OT Risk tab

When the current view contains OT-scoped assets, an **OT Risk** tab appears alongside **Network**. It is a grid of one-click investigations driven by the OT risk register. Each card runs a query against the visible map and shows the result count; click a card to filter the map and the inventory below.

Card	What it surfaces
Public OT	OT-scoped assets reachable from the public internet (`ot_scope:t AND has_public:t`)
ICS on Public	Native ICS protocols (Modbus, EtherNet/IP, DNP3, BACnet, S7, OPC UA, IEC-104, KNX, ...) on a public IP
Path Starters	Internet-facing management / lateral-movement services (RDP, SMB, SSH, WinRM, VNC, Telnet, VPN, IPMI)
OT Pivots	OT-scoped assets that are multi-homed across subnets (`ot_scope:t AND multi_homed:t`)
Mgmt Plane	OT-scoped management-plane services (SSH, RDP, SMB, WinRM, VNC, Telnet, IPMI, BMC, KVM)
Proto Gateways	Protocol gateways bridging IP topology to BACnet, EtherNet/IP, Modbus, KNXnet/IP, C37.118, ...
SIS Candidates	Hardware suggesting Safety Instrumented Systems (Triconex / HIMA / ProSafe)
OT EOL	OT-scoped assets on an end-of-life OS or firmware
OT Vulns	OT-scoped assets with high or critical risk
Misplaced OT	OT in a non-OT subnet, or non-OT in an OT subnet (segmentation gaps)
Operator UIs	OT-scoped HMIs, Panel PCs, engineering workstations, operator desktops, thin clients
Weak Mgmt	OT-scoped Telnet, FTP, VNC, TFTP, SNMPv1/v2c: cleartext or unauthenticated management

runZero computes OT scoping from the asset's `category`, the subnet classification, and a small device-type veto list (a Schneider UPS with a stale `category:OT` tag is not OT-scoped).

Search

The search bar accepts a domain-specific query language with keyword filters, free text, boolean operators (`AND`, `OR`, `NOT`), parentheses, and `*` wildcards. Bare text matches across all fields.

```
192.168.1                              # free-text match
name:web-server                        # keyword
type:server AND os:Linux               # boolean
NOT net:10.0.0.0/8                     # negation
(name:PC AND type:server) OR os:Windows
multi_homed:t AND has_public:t
type:Server AND geo:30.27,-97.74,25km
```

Identity, address, and platform

Keyword	Aliases	Matches
`id`		Node or asset ID
`address`	`addr`, `addresses`, `ip`, `host`	IP address(es)
`name`	`names`, `hostname`, `hostnames`	Hostname(s)
`net`	`cidr`	Subnet / network CIDR
`type`		Device type (Server, Desktop, Switch, IP Camera, ...)
`os`		Operating system
`os_version`	`osversion`	OS version
`hw`	`hardware`	Hardware model
`category`	`categories`	Asset category (`IT`, `OT`, `IoT`)
`subnet_class`	`net_class`	Subnet classification (`OT`, `IoT`, `General`)
`ot_scope`		OT-scoped (`t` / `f`), derived from category + subnet + vetoes
`functions`	`function`	Asset function tags
`vendor`	`mac_vendor`, `newest_mac_vendor`	Newest MAC vendor
`tag`	`tags`	Asset tags
`domain`	`domains`	DNS domains
`site`	`site_name`, `site_id`	Site
`explorer`	`agent`, `last_agent_id`	Last Explorer (UUID prefix)
`task`	`last_task_id`	Last task (UUID prefix)
`detected_by`	`det`	Detection method (`arp`, `icmp`, `ndp`, ...)

Boolean / flag fields

Keyword	Matches
`multi_homed`	Asset has addresses in multiple subnets (`t` / `f`)
`has_public`	Asset has a public IP
`has_private`	Asset has a private IP
`has_ipv4`	Asset has at least one IPv4 address
`has_ipv6`	Asset has at least one IPv6 address
`has_link_local`	Asset has a link-local address
`has_mac`	Asset has MAC address data
`has_os`, `has_hw`, `has_name`, `has_type`	Asset has the named field populated
`eol`	Asset is end-of-life (`t` / `f`)
`misplaced`	Asset's category does not match its subnet's classification
`unmapped_mac`	Node is an unmapped-MAC stub from a switch port
`is`	Topology role: `switch`, `gateway`, `child` (sub-asset behind a gateway)
`gateway`	Legacy alias: `t` / `f` for any gateway, or a specific kind. Prefer `is:gateway` / `is:switch`

Numeric fields

All numeric fields support comparisons (`>`, `<`, `>=`, `<=`, ranges).

Keyword	Matches
`risk_rank`	Numeric risk rank (0-4+)
`outlier_rank`	Outlier score rank (0-4+)
`criticality`	Criticality rank
`modified_risk_rank`	Modified risk rank (0 = unmodified)
`vulnerability_count`	Number of known vulnerabilities
`software_count`	Number of detected software packages
`service_count`	Number of detected services
`mac_count`	Number of MAC addresses
`address_count`	Number of IP addresses

`name_count`	Number of hostnames
`service_ports_tcp`	TCP port numbers (`service_ports_tcp:443`, `service_ports_tcp:>1024`)
`service_ports_udp`	UDP port numbers
`rtt`, `lowest_rtt`	Lowest round-trip time in microseconds
`hops`, `ttl`	Lowest TTL / hop count
`newest_mac_age`	Newest MAC age in seconds

Services and software

Keyword	Matches
`service_protocols`	Service protocols (`http`, `ssh`, `tls`, `modbus`, `bacnet`, ...)
`service_products`	Service product names (`Apache*`, `nginx`, `OpenSSH*`)
`protocol`	Convenience alias for `service_protocols`
`product`	Convenience alias for `service_products`

Time

`first_seen` (alias `firstseen`) and `last_seen` (alias `lastseen`) accept relative ranges like `<7days`, `>24hours`, `<30days`, or absolute timestamps like `>2026-01-01`.

Risk

Keyword	Matches
`risk`	Risk level label (`Critical`, `High`, `Medium`, `Low`, `None`)

Node and gateway types

Keyword	Matches
`node_type`	Graph node type: `a`=asset, `n`=subnet, `u`=unaddressed, `g`=gateway, `m`=unmapped MAC, `r`=router hop
`class`	Object class: `asset`, `subnet`, `gateway`, `unaddressed`, `unmapped-mac`, `router-hop`, plus device-type aliases (`switch`, `firewall`, ...)
`gateway_kind`	Gateway protocol kind: `switch`, `bacnet`, `modbus`, `cip`, `knxnet`, `s7comm`, `dnp3`, `profinet`, `ethercat`, `omronfins`, `melsecq`, `hartip`, `c37118`

`child_kind`	Asset is a sub-asset behind a gateway of the given kind (same vocabulary as `gateway_kind`)
`downlink_count`	Number of downstream links (switch ports + OT gateway children). Supports comparisons (`>5`, `<=0`, `>=10`)
`hops_from_public`	Minimum topology hops to a public-facing asset (0 = the asset itself is public-facing). Supports comparisons
`switch_port`	Switch port observation on an unmapped-MAC stub (e.g. `Gi1/0/1`)

Geographic search

Keyword	Aliases	Matches
`geo`	`geo_within`, `geo_radius`	Distance: `geo:<lat>,<lon>[,<radius>]` (`km`, `mi`, `m`, default 50km)
`country`		ISO country code
`city`		City
`state`	`subdivision`	State / subdivision
`continent`		Continent
`postal_code`	`postal`	Postal code
`timezone`		Timezone (`America/Los_Angeles`)
`asn`	`isp`	ASN ID or network owner
`geo_source`		Source of the geo point (`isp`, `cloud-region`, `attribute-ip`, ...)

See Geolocation for how runZero produces these values.

Tracing paths

The Network Map's trace engine answers "how does traffic get from A to B?" by computing shortest paths through the rendered topology. The **Trace** panel at the top of the map controls every aspect of the trace.

Sources and targets

A trace needs at least one of:

- One or more **Sources**, the start of the path.
- One or more **Targets**, the end of the path.

You can supply sources and targets in three ways:

1. Right-click any node on the map and choose **Add as Source** or **Add as Target**.

2. Type a query into the search bar, then click **Set Query As Sources** or **Set Query As Targets** in the trace panel. Every node matching the query becomes a source or target.
3. From asset details, use the **View internet attack path** action, which pre-loads `Public Internet` as the source and the asset as the target.

The same selection methods work in 2D and 3D.

Trace modes

The button in the trace panel changes its label to match what you have set:

Sources set	Targets set	Mode label	Behavior
Yes	Yes	**Trace Path**	Shortest paths between every source and every target.
Yes	No	**Trace Outbound**	All paths reachable outward from the sources.
No	Yes	**Trace Inbound**	All paths converging on the targets from the rest of the graph.

Trace from the internet

To answer “what can reach this asset from the public internet?”, set **Public Internet** as the source (right-click the public internet node and choose **Add as Source**) and the asset of interest as the target, then click **Trace Path**. The asset details page **View internet attack path** action does this in one click.

To answer “what does this server expose to the internet?”, reverse it: set the asset as the source and **Public Internet** as the target.

IP version routing

The trace engine respects two IP-version toggles in the panel’s expanded view:

- **Route IPv4** includes IPv4 paths in the trace (default on).
- **Route IPv6** includes IPv6 paths in the trace (default on).

Two more toggles control IPv6 visibility on the canvas itself, independent of routing:

- **Show IPv6 link-local** renders `fe80::/10` addresses (default on).
- **Show IPv6 global** renders global unicast `2000::/3` addresses (default on).

Widely-reused public IP filter

By default the trace ignores public subnet nodes linked to more than 5 distinct internal assets. This removes misleading edges from shared resolvers (`1.1.1.1`, `8.8.8.8`) and content-delivery

anycast addresses. You can change the threshold in the trace panel.

The Trace panel and per-hop filters

The collapsed trace panel shows the current Sources, Targets, and the trace button. Click the ▾ chevron to expand it and reveal the per-hop filter grid and the routing toggles described above.

Per-hop filters

With the hop-filter grid, you can specify an exact attack path, hop by hop. Each row is one hop from the source and has two inputs:

- **Include** allows only nodes matching this query at this hop.
- **Exclude** removes nodes matching this query from this hop.

The inputs accept the full search query language described above, and the panel shows the live match count next to each input as you type.

These filters reduce the source-to-target paths to just the hops that satisfy the constraints. They are subtractive: they cannot add a path that does not already exist in the graph, but they can carve a much smaller subset out of a noisy trace.

Examples:

- To force the path to pivot through SSH, put `service_protocols:ssh` in the **Hop 1** row's **Include** input.
- To exclude known firewalls and routers, put `type:Firewall OR type:Router` in the **Hop 1** row's **Exclude** input.
- To reach OT through a specific HMI, put `type:HMI AND name:hmi-prod-*` in the **Hop 2** row's **Include** input.
- To find paths that cross a multi-homed pivot, put `multi_homed:t` in the **Hop 1** row's **Include** input.
- To eliminate a specific subnet from consideration, put `net:10.50.0.0/16` in any hop's **Exclude** input.

Filters apply hop by hop, counting from the source side. A trace with all four hop filters empty is the unconstrained shortest path; adding any include or exclude carves the result set without re-running the search server-side.

What each hop row constrains

The hop index refers to the node's position along the traced path, counting from the source:

Filter	Applies to	Typical use
Hop 0	The source node(s) themselves; hop 0 is "0 hops from the start"	Restrict which matched sources are used, e.g. only internet-facing assets (`has_public:t`)

Hop 1	Nodes reachable one step from the source	Force or forbid the first pivot, e.g. exclude `os:Windows` so the path can't route through Windows hosts at hop 1
Hop 2	Nodes two steps from the source	Same idea for the second pivot, e.g. require `port:22` so hop 2 must expose SSH
Hop 3	Nodes three steps from the source	Constrain the deepest explicitly-controlled hop before the path continues to the target

An **Include** query means "at this hop, only nodes matching this query are allowed on the path"; an **Exclude** query means "nodes matching this query are removed from consideration at this hop." Both use the same search terms as the map's global search bar.

Effect on the tracing visualization

When you run a trace (outbound, inbound, or shortest path), the path solver treats hop filters as routing constraints:

- The animated trace path (dashed trace lines and flowing path particles) re-routes so that each hop position honors its include/exclude rules. If the unconstrained shortest path went through a node your Hop 1 exclude removes, the trace shows an alternate, often longer, route instead.
- Per-cluster hop labels and hop-colored borders on the map update to reflect the constrained path, so you can confirm the path passes through the node classes you demanded at each depth.
- If no path can satisfy all hop constraints, the trace reports "No path found between source and target". That is a useful signal in itself (for example, "there is *no* route to the OT network that avoids Windows hosts").

Effect on the choke points panel

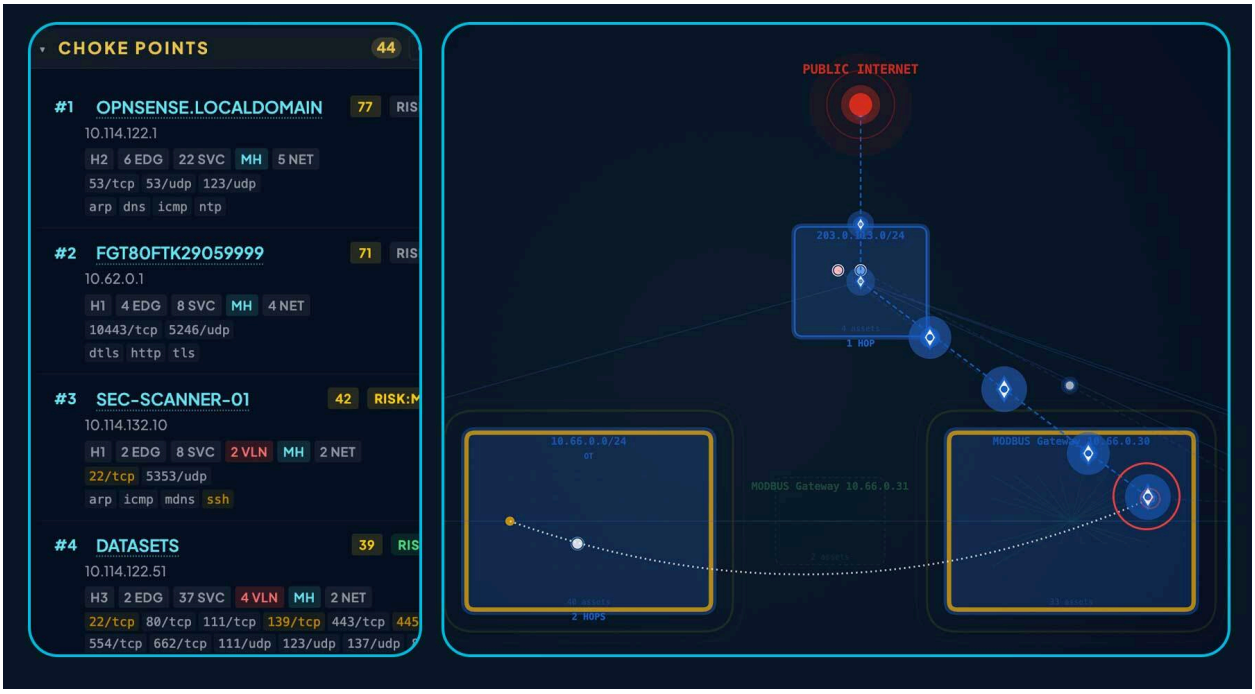
The **Choke Points** panel (left rail) derives entirely from the active trace path: it collects the intermediate pivot assets on that path, scores them (edges, clusters bridged, multi-homed, services, protocol-gateway status, and so on), sorts by choke score, and tags each entry with an **H0/H1/H2/H3 hop badge** ("Hop from source").

Hop filters are "what-if" routing constraints at each path depth, and because they change which path is traced, they reshape this list:

- An include at hop N limits the choke points shown at that depth to assets matching your query, so you can answer questions like "if an attacker must go through SSH at hop 2, which asset becomes the critical pivot?"
- An exclude at hop N removes matching assets from that hop level, and the choke-point ranking recomputes around whichever alternate pivots the rerouted path now uses.
- The choke-score priorities, gateway badges, and the CSV export (which includes the `hop` column) all reflect the filtered path, so exporting after applying hop filters gives you a prioritized pivot list for that constrained attack scenario.

Choke points

When a trace produces results, a **Choke Points** panel appears on the left rail. It lists the assets that sit on the largest share of source-to-target paths: the systems that, if removed or hardened, would disrupt the most attack paths.



How choke points are scored

Each candidate gets a **Pivot Risk Score** that combines connectivity signals with a heavy bias toward devices that are the only path into a non-IP backplane. Candidates sort by this score (descending), with ties broken by hop position and then by asset risk rank. Harden the top of the list first.

The score is built from these signals:

Signal	Weight	Why it matters
Protocol-gateway pivot	+1000 (dominates the score)	A protocol gateway (BACnet, Modbus, EtherNet/IP CIP, KNXnet, S7, DNP3, ...) is the **only** way to reach the assets on its backplane. Blocking it severs every downstream device, so it always ranks at the top.
Gateway pivot is also EOL or vulnerable	+200	Adds a “replace this first” boost when a protocol-gateway pivot is end-of-life or carries known vulnerabilities.

Cluster count (subnets bridged)	+15 per additional cluster beyond the first	Multi-homed assets that bridge two or more subnets concentrate transit traffic; each extra subnet they touch increases their importance as a pivot.
Edge count on the trace	+2 per edge, capped at 50 edges	Counts how many distinct neighbors the asset talks to on the trace path. Higher fan-in / fan-out means more flows pass through it.
Network count (distinct networks)	+5 per additional network beyond the first, capped at 10	Rewards bridging across separate networks (different CIDRs or routing domains) rather than just multiple subnets within the same network.
Multi-homed flag	+10 (one-time)	A small constant bump for any multi-homed asset, on top of the cluster / network counts above.
Service count	+1 per service, capped at 20	Heavily-exposed assets (many open services) are higher-value pivots than single-purpose hosts.

Candidates below a minimum bar are left out of the panel: an asset has to either bridge into a protocol gateway, have more than one neighbor on the trace, or be multi-homed across subnets. Protocol-gateway pivots are always kept regardless of degree because they are by construction the choke point for everything behind them. The list is capped at 100 entries so large traces stay readable.

Reading the panel

Each row shows the priority number (1 = highest), the asset's name and main attributes, and its Pivot Risk Score. Protocol-gateway rows are highlighted and include the gateway protocol kind plus the count of backplane assets that depend on it. Click a row to focus the asset on the canvas and reveal its full details.

The **Show only protocol gateways** toggle filters the list to gateway pivots, and the count badge in the section header still shows how many entries are hidden so you know whether more candidates exist. You can export the full list, filtered or not, to CSV from the **Export** menu.

Exporting

The **Export** menu in the toolbar produces both image and graph-data formats from the current map view.

Images

- **Current view (PNG)** captures exactly what is on screen, at the rendered resolution.
- **Full network (PNG)** renders the whole topology to a high-resolution PNG, regardless of the current pan and zoom.

The 3D map exports the current rotated view as PNG.

Graph data

These formats include nodes, edges, subnet clusters, and node positions, suitable for loading into other graph tools:

- **GraphML** opens in Gephi, yEd, Cytoscape, NetworkX, and any GraphML-aware tool.
- **GEXF** is the Gephi-native format, with richer attribute typing than GraphML.
- **GML** opens in Cytoscape, R `igraph`, and other classic tools.
- **Cytoscape JSON** imports directly into Cytoscape Web and Cytoscape Desktop.
- **Visio (VSDX)** opens in Microsoft Visio.

Tabular data

- **Assets (CSV)** is a flat asset table of the current view (or the search-filtered subset when a search is active).
- **Choke Points (CSV)** has one record per choke-point asset for the active trace.

Sharing a view

The map's URL captures the current filter, pan / zoom, theme, mask, search query, and any active trace. Copy the browser URL; recipients with access to the same organization see exactly the view you copied.

World Map

The World Map plots every asset that has a known geographic location on an interactive globe. It is the geographic counterpart to the Network Map: where the Network Map shows logical relationships between assets, the World Map shows where in the world each asset is.

Prerequisites

The World Map shows only assets with at least one resolved location. runZero produces locations from MaxMind GeoIP lookups, cloud region centroids, integration attributes, and user-supplied `geo*` tags. See Geolocation for the full list of sources and how to pin an asset's location manually.

Assets with no geolocation data stay off the map, and the **View on World Map** action and inventory globe icon appear only when an asset has at least one resolved location.

Accessing the map

You can open the World Map from several places in the console:

- In the **Inventory**, asset rows show a small globe icon when the asset has a known location. Click the icon to open the World Map filtered to that asset.
- In **Inventory search**, the "View on map" action above the inventory grid opens the World Map with the current inventory search applied as the map filter.
- In **Asset details**, the **Manage** menu offers **View on World Map** (or **Back to World Map** if you arrived from the map). The link appears only when the asset has geolocation data.
- In **Reports**, open it directly at `/reports/world/map` from the Reports menu, or at https://console.runzero.com/reports/world/map.
- In **Queries**, saved query rows in the queries grid include a "View on World Map" action that prefills the map with the saved query.

When you open the map from the inventory or from an asset's detail page, runZero records the previous URL as a return target. A **Back** button in the top-left of the map returns you to that exact view, including any filters and pagination you had set.

2D and 3D modes

The World Map has two modes:

- The **2D World Map** (`/reports/world/map`) is a flat projection of every asset location, optimized for searching, filtering, and exporting.
- The **3D globe** (`/reports/world/map/3d`) is an interactive rotating globe rendering the same point set, useful for executive and dashboard displays.

Both modes use the same data, search syntax, and filter state. To switch:

- From the 2D map, click the **3D** button in the top-right toolbar.

- From the 3D globe, click the **2D** button in the top-right toolbar.

Searching the map

The map's search box accepts a domain-specific query language with keyword filters and free text. Type a keyword followed by a colon and a value, or start typing and press **Tab** to autocomplete from the current point set.

The boolean operators `AND`, `OR`, and `NOT` work here, along with parentheses for grouping. Bare text (no keyword) matches across all displayed fields.

Supported keywords

Keyword	Aliases	Matches
`country`		ISO alpha-2 country code
`city`		City name
`state`	`subdivision`	State, province, or admin subdivision
`continent`		Continent name
`asn`	`isp`	ASN organization name or number
`id`		Asset ID
`name`	`names`, `hostname`, `hostnames`	Asset name or any of its hostnames
`addr`	`address`, `addresses`, `ip`, `host`	Asset IP address
`os`		Operating system
`hw`	`hardware`	Hardware identifier
`type`		Asset type
`category`	`categories`	Asset category (IT / OT / IoT)
`risk_rank`	`risk`	Numeric risk rank
`last_seen`	`lastseen`	Last-seen timestamp
`site`	`site_name`	Site name
`site_id`		Site UUID

Values with spaces or punctuation must be quoted (`city:"San Francisco"`). The search field auto-quotes when you accept an autocomplete suggestion.

Examples

```
country:US
country:US AND state:CA
```

```
city:"San Francisco" OR city:Berlin
asn:"AS15169 GOOGLE"
type:server AND country:DE
category:OT AND continent:EU
NOT country:US
"web-prod-01"
```

Searching the asset inventory by location

The asset inventory and services inventory accept a `geo` keyword (synonyms: `geo_within`, `geo_radius`) that matches assets whose geolocation is within a circular area around a point. This is the recommended way to scope inventory queries by location.

```
geo:<lat>,<lon>[,<radius>]
```

- `<lat>` and `<lon>` are decimal degrees.
- `<radius>` is optional and defaults to **50 km**. Supply a number followed by `km`, `mi`, or `m`, for example `25km`, `15mi`, `5000m`. Plain numbers are treated as kilometers. The radius is clamped between 1 m and 1,000 km.

Examples

```
geo:30.2672,-97.7431              # within 50 km of downtown Austin
geo:30.2672,-97.7431,25km         # within 25 km
geo:52.5200,13.4050,100mi         # within 100 miles of Berlin
geo_within:51.5074,-0.1278,5km    # synonym
```

The query matches if **any** of the asset's geolocation sources falls within the circle. See Geolocation for details on how runZero builds those locations.

Exporting

The **Export** menu in the top-right toolbar exports the currently filtered point set in these formats:

- **CSV** is a flat table of every visible asset and its location fields.
- **JSON** holds full point records for programmatic use.
- **GeoJSON** is a standard GeoJSON `FeatureCollection`, suitable for loading into GIS tools.
- **KML** and **KMZ** are for Google Earth and compatible viewers.

Sharing a view

The **Share link** button copies a URL that captures the current filter, pan, zoom, and any opened asset popup. Recipients with access to the same organization see exactly the view you copied.

Organization Overview

The Organization Overview Report is a point-in-time snapshot of the asset data in your organization and sites. It organizes your asset inventory into sections and summarizes the major findings, so you can share it with teams and leaders who may not have access to runZero and need an at-a-glance look at their network.

The report gives you high-level metrics for your organization and sites across categories such as asset types, operating systems, hardware, protocols, and products, plus a summary of your RFC 1918 coverage, subnet utilization, and switches. For organizations with fewer than 50,000 assets, you can add asset details and screenshots from your inventory.

You can schedule the report to run at a specified start time and frequency: once, daily, weekly, monthly, or on the Nth weekday of each month. When a report runs on a schedule, runZero can notify one or more email addresses after each run.

runZero generates the report in HTML. To distribute it as a PDF, use your browser's Print to PDF feature.

Generating the Organization Overview Report

1. Go to the Reports page.
2. Open the Organization Overview Report configuration form.
3. Enable **Include asset details** to add information about each asset: asset type, operating system, hardware, outlier score, first seen date, last seen date, site, addresses, names, MAC addresses, protocols, products, and services. This option applies only to organizations with fewer than 50,000 assets.
4. Enable **Include screenshots** to add captured screenshots to the report. This option applies only to organizations with fewer than 50,000 assets.
5. Configure **Start time** and **Report frequency** to run the report at a specified time or on a recurring schedule.
6. Enter one or more email addresses in the **Email addresses** field to get email notifications when the report runs.
7. Generate the report. When generation completes, the report opens automatically in your browser. Save it as a PDF to share.

Email notifications

When you schedule the report, you can enter a comma-separated list of email addresses. Each address receives a notification when a scheduled run finishes, with a link to the report in the runZero Console. Only registered users with access to the relevant organization can open the completed report.

RFC 1918 Report

Coverage reports show which IP spaces you have scanned, which contain assets, and which are still unknown, so you can uncover potential blind spots on your network. With this information, you can find missing subnets, rogue devices, and misconfigurations.

To open the coverage reports, go to **Reports** on the main menu and scroll down to **RFC 1918 coverage**.

RFC1918 coverage report

The RFC1918 coverage report tracks the subnets in use on your internal network, which of them you have scanned, and which you have not.

TCP/IP version 4 reserves three ranges of IP addresses for private use. Specified in RFC1918, they are:

CIDR	Address range	Number of addresses
10.0.0.0/8	10.0.0.0 - 10.255.255.255	16,777,216
172.16.0.0/12	172.16.0.0 - 172.31.255.255	1,048,576
192.168.0.0/16	192.168.0.0 - 192.168.255.255	65,536

Most companies use these ranges for their internal IPv4 networks and connect them to the Internet through Network Address Translation (NAT).

To help you visualize and assess your RFC1918 coverage, the RFC1918 report includes:

- Coverage maps
- Subgrids
- Statistics

RFC1918 coverage maps

A common network security and administration goal is to scan all of the available private IP addresses to detect which subnetworks are in use on your internal network. With so many addresses, this can take a long time, and you need a way to track which addresses you have scanned and which you have not.

The Coverage report solves this with a graphical map of the RFC1918 private address spaces that shows which pieces you have scanned and what percentage of each you have scanned.

runZero sometimes detects that a device has additional IP addresses outside the range being scanned. That can mean the device is present on an unscanned part of the private IP address space, and the coverage reports mark it by drawing a red border around the matching grid cell.

Hover over a cell to see a tooltip with the CIDR address range the cell represents, how many unscanned hosts are believed to be in that range, and what percentage of the entire range you have scanned.

RFC1918 sub-grids

On the 192.168.0.0/16 map, each grid cell is a /24 (256 addresses). Clicking a cell takes you to the subnet analysis report for that range, which lists the assets found.

On a map of a large address range, such as 10.0.0.0/8, each cell is an entire /16 of 65,536 addresses. To narrow the search for assets and unscanned hosts, click any /16 cell to open a grid map of just that range. From a /16 grid, the link at top right takes you back to the full range map.

On the /16 sub-grids, each cell is a /24, so clicking one takes you to the subnet analysis for that cell's address range, as on the 192.168.0.0/16 map.

Statistics

At the top of the coverage report page, statistics show how much of the RFC1918 address space you have scanned. Another box breaks the coverage down by the three blocks of reserved addresses.

Clicking the magnifying glass icon in the summary box creates a sample scan task covering the unscanned address ranges. The refresh buttons create scan tasks to rescan all of the corresponding range. To understand and optimize your coverage after a full discovery scan, see Identifying gaps in scanning.

External Assets

The External Asset Report is a point-in-time snapshot of the external asset data in your organization and sites. It organizes your asset inventory into sections and summarizes the major findings, so you can hand it to teams and leaders who may not have runZero access and need an at-a-glance look at their public-facing assets.

The report gives you high-level metrics for external assets across categories such as asset types, operating systems, hardware, protocols, and products, plus a summary of top Certificate Authorities and GeoIP countries. For organizations with less than 50,000 assets, you can add asset details and screenshots from your inventory.

You can schedule the report to run at a specified start time and frequency: once, daily, weekly, monthly, or on the Nth weekday of each month. When a report runs on a recurring schedule, runZero can notify one or more email addresses after each run.

runZero generates the report in HTML. To distribute it as a PDF, use your browser's Print to PDF feature.

Generating the External Asset Report

1. Go to the Reports page.
2. Open the External Asset Report configuration form.
3. Enable **Include asset details** to add information about each asset: asset type, operating system, hardware, outlier score, first seen date, last seen date, site, addresses, names, MAC addresses, protocols, products, and services. This option only applies to organizations with less than 50,000 assets.
4. Enable **Include screenshots** to add captured screenshots to the report. This option only applies to organizations with less than 50,000 assets.
5. Enable **Include unscanned subnet details** to add results from external subnets that runZero has not scanned.
6. Enable **Include TLS certificate details** to add the details runZero gathered about TLS certificates.
7. Use the **External IP exclusions** if your internal network uses public IP address space and you do not want those assets in the report.
8. Configure **Start time** and **Report frequency** to run the report at a specified time or on a recurring schedule.
9. Enter one or more email addresses in the **Email addresses** field to receive an email notification when the report runs.
10. Generate the report. When generation completes, the report opens in your browser. Save it as a PDF to share and distribute.

Email notifications

When you configure the report to run on a schedule, you can enter a comma-separated list of email addresses. Each address receives a notification when a scheduled run finishes, with a link to the report in the runZero Console. Only registered users with access to the relevant

organization can open the completed report. To share a dashboard or inventory view with people who do not have a runZero account, see External reporting.

Site Comparison

The Site and organization comparison feature generates a side-by-side analysis of two sites, so you can see:

- How assets change over time, including their TCP/UDP services, TCP/UDP ports, and service protocols. Use this data to evaluate historical changes to assets at a specific point in time.
- How exposure changes when you scan your network from different locations. For example, if you use public IP addresses internally and externally, you may want to scan those addresses from inside and outside your network to understand your potential exposure.

The report is a summary view of differences. It only captures certain attributes that were added to or removed from an asset, such as IP addresses, TCP ports, TCP service counts, UDP ports, UDP service counts, service protocols, and service counts. It does not track every modification to an asset, such as fingerprint or service banner changes.

After the report runs, its data is static. If your inventory changes afterward, some assets may no longer be reachable from the report.

Generate a site comparison

A site comparison needs a current site and a comparison site. The sites can be in different organizations. You can also select “All sites” to compare all sites in an organization against a different site and organization.

When the report runs, runZero assembles the two sets of assets, compares them with its asset matching algorithms, and generates a set of differences that you can browse in the report.

To generate a site comparison:

1. Verify that your current organization contains the inventory of assets you want to compare.
2. Go to the site comparison page.
3. On the site comparison configuration page, the current organization is the one you have selected. Change the current site if needed. For the comparison, choose the organization and site you want to run the analysis against.
4. Run the report. runZero creates a task to perform the comparisons and takes you to the task page.
5. When the task completes, the report appears in the list of recent analysis reports at the top of the Reports page. You can then view and search the results.

View how assets change over time

To see how assets have changed over time, compare the data from an old scan task with your most recent inventory. A point-in-time comparison needs a new project to import the old scan

data into.

To set up a point-in-time analysis:

1. Go to the organization or project that contains the scan task data you want to use.
2. Go to your completed tasks and locate the task with the data for the point in time you want to compare.
3. From the task page, download the task data. The file name starts with `scan_` and ends with `.json.gz`. This is the file you'll import into your new project. You don't need to uncompress it, unless you're curious about the JSON data.
4. Create a new project for your import. If you intend to run the analysis regularly, you can create an organization.
5. After you create the project, go to the Inventory page and import your scan task data into it.

Now you can compare your current inventory with its previous version. Go to the site comparison page and select the organization and site for your current inventory as the site to compare against.

After the report runs, it shows a table with the differences between the two sites, which in this case are two points in time, going from past to present.

You can also run the comparison the other way: select your current organization first, then choose the project with the past data as the comparison site. The results are the same with their sense reversed. Services that show as added going from past to present show as removed going from present to past.

View how exposure differs between networks

The site comparison report can also show how exposure varies with where you scan your network from. Comparing two inventories taken from two perspectives gives you a better view of your attack surface, which can help with active defense or risk reduction. For example, if you use external IPs internally and externally, you may want to scan those addresses from inside your network and from outside it, then run the site comparison report on the results from those two sites.

To set up a diff of exposures between networks:

1. Set up a site with an Explorer on one network.
2. Set up another site with an Explorer on a different network. This site can be in the same organization as the first.
3. Scan the same address range with each Explorer. Verify you have the correct site selected for each scan.
4. After the scans complete, run the site comparison to generate the diff. The report shows a table with the differences between the two sites.

Analyze the results in the site comparison report

The site comparison generates a table showing how assets' addresses, names, services, ports, and protocols differ between sites. Red text with a minus (-) sign marks attributes

removed going from left to right. Green text with a plus (+) sign marks attributes added.

How to read the columns:

- **Address and Other Address** - The first address column contains the asset's addresses in the organization that was current when you requested the report. The second address column, *Other Address*, contains the asset's addresses in the organization and site compared against it.
- **Name and Other Name** - The first name column contains the asset's names in the organization that was current when you generated the report. The second name column, *Other Name*, contains the asset's names in the organization and site compared against it.
- **TCP and UDP services** - These columns show how the total number of TCP and UDP services differs between sites.
- **TCP and UDP ports** - These columns show the ports added or removed between sites.

Click the green info (i) icon to view a more detailed comparison of the asset.

Clicking one of the asset addresses opens the full current asset record, if the asset still exists in the relevant organization and site.

Search the site comparison report

You can search the report with the runZero search query language. In the keyword descriptions below, the main set is the assets in the organization that was current when you generated the report (the address and name columns). The comparison set is the assets in the organization and site you chose to compare against (the other address and other name columns).

Keyword	Meaning
`address:`	Search for assets in the main set with a specified IP address. Use `none` to find assets that are missing from the main set.
`net:` or `cidr:`	Filter assets by their network CIDR range in the main set.
`other_address:`	Search for assets in the comparison set with a specified IP address. Use `none` to find assets that are missing from the comparison set.
`other_net:` or `other_cidr:`	Filter assets by their network CIDR range in the comparison set.
`id:`	Search by asset ID in the main set.
`other_id:`	Search by asset ID in the comparison set.
`tcp:`	Search for a TCP port change by number.
`udp:`	Search for a UDP port change by number.
`protocol:`	Search for a TCP or UDP port change by service name.

External reporting

External reporting packages a dashboard, an inventory view, or a runZero report into a point-in-time **snapshot** that you can share with people who do not have a runZero account. Recipients open a link, verify their email address, and view the snapshot in a read-only browser page. Snapshots are immutable, scoped to the organizations you select, and can be regenerated on a recurring schedule.

Use it to give executives, auditors, customers, or other teams an at-a-glance view of your asset inventory, security posture, or a specific report without provisioning runZero accounts for them.

What you can share

You can share two types of external report:

- A **Dashboard** report captures any dashboard you can view, including cross-organization shared dashboards. The snapshot holds the dashboard widgets for a configurable date window.
- An **Inventory** report captures an asset, software, service, or vulnerability inventory query, including the columns you have selected and the current sort order. The snapshot freezes the matching rows and serves them to the recipient as a read-only table.

Sharing a report

You can start an external report from anywhere in the runZero Console that displays one of the supported views:

- Open the dashboard and click the **Share** button in the header. The button appears on dashboards that can be shared, such as runZero-managed and cross-organization dashboards.
- From an inventory page (Assets, Software, Services, or Vulnerabilities), run the query you want to share and use the Share button to create an external report. The current query, columns, and sort order are saved with the report.

The External Reports page lists every report you have created and lets you edit or delete them.

Share dialog options

When you create or edit an external report, the share dialog asks for:

- **Report name** identifies the report in the runZero Console and appears in the email sent to recipients.
- **Email recipients** lists one or more email addresses to notify each time a snapshot is generated. The notification email contains the link recipients use to open the report.

- **Report frequency** sets how often the report regenerates. Choose **One time** for a single snapshot, or **Daily**, **Weekly**, or **Monthly** for a recurring schedule.
- **Organizations** (My Organizations view) selects the organizations whose data the snapshot includes. You can only include organizations you have access to.
- **Automatically include new organizations** (My Organizations view) adds organizations created after the report is set up to future runs of a recurring report, provided you have access to them.

Click **Send report** to save the configuration and queue the first snapshot. Snapshot generation runs in the background, and runZero emails the recipients as soon as the snapshot is ready.

How recipients view a report

When a snapshot is generated, each recipient receives an email with a link to the report. The first time they open the link, runZero asks them to accept the runZero terms of service.

runZero then emails a single-use verification link to that address. Clicking it confirms the recipient's identity, sets a secure session cookie that lasts for 30 days, and redirects them to the report. While the cookie is valid, the recipient can return to the same report, or to any other report shared with the same email address by the same runZero account, without verifying again.

The verification link expires 30 minutes after it is sent. A recipient who misses that window can request a new one by reopening the report link and entering their email address again.

Recipients whose runZero account has access to one of the report's organizations can open the report with their existing session and skip the email verification step.

Managing reports and snapshots

The External Reports page shows every external report your organization has created, grouped under three tabs:

- **All** shows every external report.
- **One-time** shows reports configured to run once.
- **Recurring** shows reports configured to run on a daily, weekly, or monthly schedule.

Clicking a report opens its details page, which shows the report's configuration and a table of every snapshot generated so far. From this page you can:

- **Edit** the parent report to change its name, recipients, schedule, date window, or organizations. Edits affect future runs only; existing snapshots stay as they are.
- **Delete** the parent report, which stops future runs and removes all of its snapshots.
- **Copy** the share link for an individual snapshot.
- **Revoke** a snapshot's access link. Revoking invalidates the URL so it can no longer open the report, but leaves the snapshot record in place for auditing.
- **Delete** an individual snapshot, which revokes its link and removes the captured data.

Each snapshot link is valid for 90 days from the time the snapshot is created. After that the link stops working; recipients of recurring reports continue to receive a fresh link in each run's email.

Security and access

External reporting is designed so that report data leaves runZero only through links you explicitly create:

- Each snapshot link contains a 256-bit random token signed with a per-account HMAC key. runZero rejects tampered or fabricated links without a database lookup.
- Before a recipient can view a snapshot, they must prove ownership of an email address by clicking a verification link sent to that address. Their identity is then carried in a secure, account-scoped browser cookie.
- Each time a recurring report runs, runZero re-checks the original creator's access to the report's organizations. Organizations they no longer have access to are dropped from the snapshot. If they have lost access to all of the report's organizations, the run is skipped.
- Every successful access of an external report is recorded as an `external-report-viewed` event for each organization in the snapshot. The event includes the recipient's email address, IP, and user agent, and appears in the runZero activity log.
- The public verification and access endpoints are rate limited to mitigate abuse.
- You can revoke any snapshot link at any time from the report's details page. Revoked links cannot be reactivated.

Switch Topology

Most users should start with the Network Map, which combines switch topology, network bridges, and route pathing into a single interactive view with search, hop-filter tracing, and choke-point analysis.

The runZero switch topology report draws a graph of the switches and routers on your network and shows how they interconnect. It also shows which assets connect to each switch.

The switch topology report requires a source of layer 2 topology information. That usually comes from SNMP, or from connecting an integration that provides topology information, such as Cisco Meraki.

Generating the switch topology report

Once you have topology data, open the switch topology report.

The initial view is a graph of network switches, with the links between them drawn as lines. Beneath each switch is a count of the known assets detected with that switch as their upstream network connection.

You may also see a count of unmapped assets. These are assets inferred from MAC addresses found in the topology data that aren't known to belong to any scanned asset. The Unmapped MACs report lists unmapped MACs in tabular form.

The bottom of the page shows the total number of assets in the graph and the total number of unmapped MACs.

The report attempts to lay out the graph sensibly. You can drag nodes around to make it clearer.

Clicking a switch node opens a pop-up window with more information about it, including a link to the switch's asset details and a link to a list of its unmapped assets.

Double-clicking a node expands it to show the individual assets connected to it. Clicking an asset opens a pop-up window with a link to its asset details.

Filtering the switch topology report

The switch topology report can display only a limited number of assets. To focus on the assets you care about, use the filter box at the top of the report page. It accepts search strings in the standard runZero search language, so you can filter to a subnet with a search string such as `cidr:10.1.6.0/24`.

Use the drop-down to switch quickly between sites.

Buttons in the filter box collapse or expand all of the switch nodes at once.

The *Export view* button renders the graph as currently displayed to a PNG file and downloads it.

Limitations of the switch topology report

The switch topology report may not always be entirely accurate, because the data runZero can gather is limited.

When SNMP data is available, runZero pulls a snapshot of it from each device at scan time, then builds the topology from that. In complicated environments, a single infrequent snapshot is often not enough to show the complete picture, and links may end up missing. Only recent SNMP data is used: if a device has not been scanned in the last 9 days, its SNMP topology data is ignored.

Without SNMP information, runZero attempts to compute topology from which switch claims to have seen the MAC, which may not be the nearest access switch. The algorithm looks for the port with the fewest shared MACs to find the best match, but that depends on the switch cache timeouts and how the switch was scanned, so the graph may show links that don't exist as direct physical connections.

Cisco Catalyst devices

SNMPv3 on Cisco Catalyst devices will not let you pull the bridge port information runZero needs unless you enable per-VLAN access.

SNMP v3 access to VLAN ARP/FDB tables requires this access rule:

Version	Command
Newer IOS:	`snmp-server group YourGroupName v3 auth context vlan- match prefix`
Older IOS:	`snmp-server group YourGroupName v3 auth context vlan-1` (repeated for every VLAN)

Even with this rule in place, runZero has to send a separate SNMP request for every VLAN, which can significantly slow down scans with SNMP enabled on a network with many Catalyst devices. For more on gathering SNMP data during scans, see Scanning with SNMP.

Network Bridge Report



Most users should start with the Network Map. It surfaces the same multi-homed bridge assets and adds subnet clustering, choke-point ranking, hop-filter tracing, and exports to Gephi, Cytoscape, and Visio.

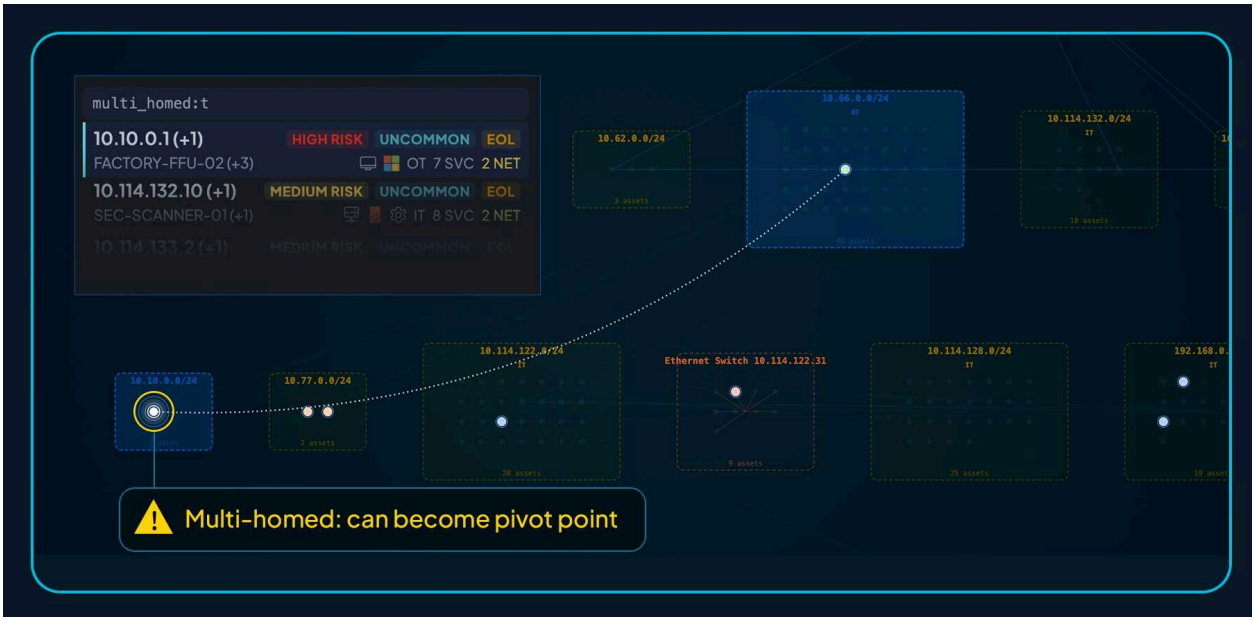
The network bridge report finds assets that bridge two or more network segments. These multi-homed assets are the pivots an attacker uses to move between zones: between guest Wi-Fi and corporate LAN, between IT and OT, or between an internal segment and the public internet.

Accessing the report

- Open it from the **Reports** menu, or go directly to `/reports/analysis/bridges` (https://console.runzero.com/reports/analysis/bridges).
- On the Network Map, the same assets appear under the **Pivots** stat in the Network Intel card (`multi_homed:t`). Each multi-homed asset shows up once in every subnet it has an address in, with edges drawn between the copies so you can see exactly which segments it bridges.

How bridges are detected

runZero detects network bridges by collecting extra IP addresses returned in responses to common discovery probes (NetBIOS, SNMP, MDNS, UPnP, and others) and by correlating MAC addresses across subnets. It reports a bridge whenever the same asset is observed with addresses in two or more distinct subnets.



Detection is opportunistic. Hardened endpoints (host firewalls, disabled discovery services) may not reveal their other interfaces, so the report can miss bridges. Treat what it shows as a floor, not a ceiling.

Reading the report

- External networks are drawn in red, internal networks in green.
- Edges connect a subnet to every multi-homed asset that has an address in it.
- Single-homed assets are omitted to keep the graph readable.

The visualization focuses on segmentation by design; it is not a full topology view. For a full Layer-2 / Layer-3 picture, use the Network Map. For switch-port-level connectivity, use the Switch Topology report.

Querying multi-homed assets directly

Every bridge in this report is a multi-homed asset. You can query them straight from the inventory:

```
multi_homed:t
multi_homed:t AND has_public:t        # bridges with at least one public IP
multi_homed:t AND category:OT         # OT pivots — top-of-register OT risk
```

Combine with `subnet_class:OT`, `category:IT`, `risk:high`, or `eol:t` to surface the bridges that matter most for segmentation hardening.

See also

- Network Map: full interactive topology with bridge and pivot analysis built in.

- Understanding network segmentation: background on segmentation gaps and how to remediate them.
- Switch Topology: switch-port-level connectivity from SNMP and integrations.

Asset Route Pathing



Most users should start with the Network Map, which lets you set sources and targets, trace inbound or outbound paths, apply per-hop include/exclude filters, and rank choke points in one interactive view.

The asset route pathing report visualizes the potential network paths between a source asset and a destination asset in an organization. Following the paths, you can see the assets connected between source and target. Each of them is an opportunity an attacker could use to break into your target asset.

The runZero Explorer performs a traceroute between itself and the source, and another between itself and the target. runZero then compares the data to infer shared points between the assets. runZero does not get any paths from a direct traceroute. This is its best effort, based on the scan data it has, to identify the assets it sees as viable points between two assets.

Share this report with your IT and security teams to show which assets could act as pivot points to your critical assets. With that information, they can identify systems that may need hardening and check whether the appropriate critical controls are in place to prevent unauthorized access to those assets.

If your environment is highly segmented, you can use this report to quickly find paths from low security assets to critical assets. For example, it can show whether a device in a wireless guest network can reach a system in the PCI cardholder data environment.

Terminology

The report uses these terms:

- **Hop** - Any node between the source and destination.
- **Node** - Any asset or hop. A node can be an IP, an asset, or unknown.
- **Asset** - Any device in your runZero inventory.
- **Network path** - runZero does not get any paths from a direct traceroute. Instead, it uses the data it has to identify and display potential network paths between the target and source.

Generating the asset route pathing report

1. Launch the asset route pathing report.
2. On the *Asset route pathing* page, select a source asset and a destination asset. Use the search to filter assets by keyword, or the table pagination to browse all of your assets.
3. With a source and destination selected, start the trace.

4. The results show the potential paths between the source and destination asset.

Analyzing the report

Locate your source asset and target asset. Review any hops between them and secure those paths. Look at the services running on those systems that may offer entry points for attacks, and harden them.

Nodes in the asset route pathing report are color-coded so you can identify the source asset and destination asset.

The report uses these colors:

- Green - The source asset
- Red - The destination asset
- Orange - A multi-homed asset that may act as a pivot point
- Blue - A standard layer-3 routing hop
- Gray - Asset is unknown

A hop labeled `Unknown` is an intermediate hop in the layer-3 path that did not respond with ICMP errors for TTL exceeded packets.

Sharing the report

You can share the results from the asset route pathing report by exporting a PNG or dotfile of the report, or by sharing a direct link to it.

- To export a PNG snapshot of the asset route pathing report, click **Export view**. The PNG downloads to your computer.
- To render the image in formats like SVG, PSD, and PNG, export a dotfile and feed it into a Graphviz engine or an open source visualization tool.
- To share a link, copy the URL while viewing the generated report. Other team members with a runZero account and access to the organization can open it.

Exporting a dotfile

A dotfile is a text file that Graphviz engines and open source visualization tools can read. With the dotfile, you can render the image in other formats, like SVG, PSD, and PNG.

1. Launch the asset route pathing report.
2. On the *Trace path* page, select a source asset and a destination asset. Use the search to filter assets by keyword, or the table pagination to browse all of your assets.
3. With a source and destination selected, generate the report.
4. When the visualization appears, click the **Export report** button. A window prompts you for a name and download location for the file.
5. Enter a name for the file and choose where to save it on your computer.
6. Save the file.

FAQs

Why are there hops in the report that aren’t in my inventory?

Not every hop is a runZero asset. runZero fills in IP addresses for some asset hops from information in the traceroute data. Sometimes that data includes asset information that isn’t part of the inventory, which is why those hops appear in the report.

Can runZero determine how two assets are talking to each other?

No. runZero can only identify potential paths, not whether they are routable. It does not test or validate the paths.

Why isn’t the asset route pathing report available?

The asset route pathing report is only available to runZero Platform customers.

Cryptographic bill of materials

The **Cryptographic bill of materials** (CBOM) report on the Reports page inventories the cryptographic algorithms, protocol versions, certificates, public keys, and reused key material observed in your organization. Use it to identify cryptography that is already weak, find classical public-key cryptography that will need a post-quantum migration, and prioritize remediation.

You can also export the cryptographic inventory as CycloneDX JSON from the asset export API:

```
https://console.runzero.com/api/v1.0/export/org/assets.cbom.json
```

Post-quantum status

The report assigns each observed cryptographic component a post-quantum status. The status is not a simple claim that an entire service is quantum-safe or quantum-unsafe. It accounts for the algorithm's cryptographic role, protocol version, negotiated key exchange, and known classical weaknesses.

Status	Meaning
Unsafe	The cryptography is broken or provides little or no protection today, independently of quantum computing. Examples include SSL 2.0, SSL 3.0, RC4, single DES, MD5, NULL, and export-grade cipher suites.
Deprecated	The cryptography is superseded and should be removed, but is not considered practically broken in the observed role. Examples include TLS 1.0 and 1.1, SHA-1, 3DES, 64-bit block ciphers such as Blowfish and CAST, RIPEMD, and severely truncated authentication tags.
Transitional	The cryptography is classically strong today but does not demonstrate post-quantum readiness, or it uses a classical asymmetric algorithm that a sufficiently capable quantum computer could break. Examples include classical key agreement, RSA key transport, classical public-key encryption and signatures, TLS 1.2, TLS 1.3 without a recognized post-quantum key exchange, and SSH with only classical key exchange.
Safe	The observation includes a recognized post-quantum or hybrid algorithm, or a symmetric or hash construction that retains useful security against quantum attacks. AES-128, AES-192, AES-256, SHA-256, and SHA-384 qualify; they are not downgraded solely because of Grover's algorithm.
Unknown	Cryptographic material was observed, but the algorithm, protocol evidence, or cryptographic role could not be recognized reliably. Unknown is a visibility or classification gap, not an assumption that the cryptography is safe.

TLS and SSH rules

For TLS, **Safe** requires both TLS 1.3 and a recognized post-quantum or hybrid key exchange. TLS 1.3 without that exchange is **Transitional**. TLS 1.0 or 1.1 is **Deprecated**, and SSL 2.0 or 3.0 is **Unsafe**, even when a service also supports TLS 1.3 and a post-quantum exchange.

For SSH, a recognized post-quantum or hybrid key-exchange algorithm advertised in the server's KEXINIT response is **Safe**. A server that advertises only classical key exchange is **Transitional**. If no key-exchange algorithm is observed, the result is **Unknown**.

Combined results

When a certificate contains both a public-key algorithm and a signature algorithm, the certificate receives the worse status of the two. A cipher suite is assessed the same way from its component algorithms. For an asset, the report uses the worst status observed across its relevant TLS and SSH services.

The status severity order is:

Unsafe > **Deprecated** > **Transitional** > **Safe**

Unknown stays an explicit uncertainty state and is not counted as Safe.

Migration deadlines

The report shows published transition schedules alongside the inventory. For United States federal high-value assets and high-impact systems, Executive Order 14412 sets these dates:

- December 31, 2030: post-quantum cryptography for key establishment, as defined in FIPS 203.
- December 31, 2031: post-quantum cryptography for digital signatures, as defined in FIPS 186-5.

These dates are published milestones, not runZero estimates. The report keeps FIPS approval separate from post-quantum status: an algorithm can be post-quantum without being FIPS-approved, and a FIPS-approved algorithm can still be vulnerable to quantum attacks.

Interpreting the results

Prioritize **Unsafe** findings: they are present-day cryptographic weaknesses. Address **Deprecated** findings next, then use **Transitional** findings to build and execute your post-quantum migration plan. Investigate **Unknown** findings to decide whether you need more scanning, service evidence, or algorithm identification.

Additional resources

Leveraging the API

runZero provides three primary APIs, plus integration-specific endpoints:

- The Export API provides read-only access to a specific organization.
- The Organization API provides read-write access to a specific organization (Professional and Platform licenses).
- The Account API provides read-write access to all account settings and organizations (Platform license).

To get started, you need an API key / token or API client credentials.

API keys and tokens

The console supports five types of runZero API key. Each type grants its own fixed level of access to runZero APIs; for a credential with an explicit set of permissions, use an API client instead. As a further safeguard, you can create an IP address allowlist that restricts API access to specific IP addresses and network ranges (in CIDR notation).

Export tokens

An export token can only export data through the export API endpoints under the `/api/v1.0/export` path. It cannot access any other API.

An export token has its organization encoded into it, so it can only export data for that organization and you never need to specify the organization when using it.

Export tokens start with the `ET` prefix.

To generate an export token, go to the Organizations page and click the organization to open its details page. Click **Edit organization**, scroll down to the export tokens section, and use the button to generate or regenerate the token.

Download tokens

A download token lets you download the runZero Explorer, which is useful for automating downloads for deployment across many machines or for containerizing the Explorer. It cannot access any other API.

Like an export token, a download token has the organization encoded into it, which determines the downloaded Explorer's organization.

Download tokens start with the `DT` prefix.

To get the download API token, go to the Organizations page and click the organization to open its details page. Click **Edit organization** and scroll down to the download token section.

Organization API tokens

An organization API token gives read and write access to the data in one organization through the organization API. Most organization endpoints are under the `/api/v1.0/org` path. Organization tokens can also access the export APIs.

An organization API token has its organization encoded into it, so it can only access that organization and you never need to specify the organization in a call.

Organization API tokens start with the `OT` prefix.

To generate an organization API token, go to the Organizations page and click the organization to open its details page. Click **Edit organization** and scroll down to the organization API tokens section, where buttons let you generate named organization API tokens and revoke them.

Account API tokens

An account API token gives read and write access to the account API endpoints, which perform account-level operations such as creating and deleting organizations and users. Account API tokens also work for organization-level access and for the export API.

Account API tokens require a Platform license and are generated from the Account settings page. To use one with the Organization or Export API, add the parameter `_oid=[organization-id]` to the query parameters.

An account API token has no organization encoded into it, so any call that applies to an organization must specify one by appending an `_oid` value to the URL query parameters. The `_oid` is the organization's unique ID, shown on the organization's information page.

Account API tokens start with the `CT` prefix (client token).

To generate an account API token, go to the Account settings page under Account in the left navigator, where you can create and delete account API tokens.

API client credentials

The tokens above are bearer strings with fixed access. API client credentials are different: a client ID and secret exchanged for a short-lived access token through OAuth2. Each client carries an explicit set of permissions and the organizations those permissions apply to.

Because their access is explicit, API clients are the right choice for anything that should not hold full account access. A client can also be tied to the user who created it, so it narrows automatically when that person's access changes.

Users create their own from My API clients. Superusers manage account-wide ones from the API clients page. Your REST client should use the OAuth 2.0 authorization type and the Client Credentials grant type. See the OpenAPI specification for the access token details.

See API clients for the full details.

Authentication

For export, organization, and account tokens, your REST client sends the token in the standard `Authorization: Bearer` header.

For API client credentials, use the client ID and client secret in an OAuth2 call to generate an access token. For example:

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "client_id=<CLIENT_ID>" \
  --data-urlencode "client_secret=<CLIENT_SECRET>" \
  https://console.runzero.com/api/v1.0/account/api/token
```

API rate limiting

API calls are rate limited. You can make as many API calls per day as you have licensed assets in your account. For example, with 1,000 licensed assets you can make 1,000 API calls per day. Each response carries rate limit information in its HTTP headers:

- `X-API-Usage-Total` - Total number of calls made to the API
- `X-API-Usage-Today` - Number of calls made to the API today
- `X-API-Usage-Limit` - Your daily API call limit, shared across all API keys
- `X-API-Usage-Remaining` - The number of API calls remaining from your daily limit

A second limit allows 2,000 requests per 5 minutes per source IP address. Beyond that, runZero returns an HTTP 429 response with the error “request limit exceeded”, and your client code should delay the next request.

IP address allowlist

An IP address allowlist restricts API access to specific IP addresses and network ranges (in CIDR notation). You configure it in the Account Settings section of the console, under the setting “API key IP address allowlist”.

The field accepts a comma-separated list of IPv4 and IPv6 addresses and network ranges (in CIDR notation) and applies to all API usage across the console. An empty allowlist disables the feature, so every IP address can access the API.

runZero API usage examples

These examples show how to use `curl` with the runZero APIs, including the Organization API for creating scan tasks and the Export API for retrieving data.

Scan API usage

You'll need:

- An Organization API token (prefixed with `OT`)
- The Site ID where you want scan results stored

Set environment variables

Set your Organization API token and Site ID as environment variables:

```
export RUNZERO_ORG_TOKEN="OT-YOUR-API-TOKEN-HERE"
export YOUR_SITE_ID="YOUR-SITE-ID-HERE" # Replace with the actual Site ID
```

API endpoint

The endpoint that starts a scan task in a specific site:

```
PUT /api/v1.0/org/sites/{site_id}/scan
```

Example base URL:

```
https://console.runzero.com/api/v1.0/org/sites/${YOUR_SITE_ID}/scan
```

Create a scan task for a specific Explorer

To run a scan on a specific Explorer, set the `"explorer"` property in the JSON payload to the Explorer's UUID.

```
curl -X PUT "https://console.runzero.com/api/v1.0/org/sites/${YOUR_SITE_ID}/scan" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${RUNZERO_ORG_TOKEN}" \
  -d '{
    "explorer": "<YOUR-EXPLORER-ID>",
    "targets": "192.168.1.0/24",
    "scan-name": "Scan via Specific Explorer",
    "rate": "3000"
  }'
```

Create a scan task using a hosted Explorer

To run an external scan with a runZero hosted Explorer, use the `"hosted-zone-name"` key.

- Set `"hosted-zone-name": "auto"` to let runZero choose the best hosted zone.

- You can also name a specific hosted zone.
- Targets must be public IP addresses or hostnames.

```
curl -X PUT "https://console.runzero.com/api/v1.0/org/sites/${YOUR_SITE_ID}/scan" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${RUNZERO_ORG_TOKEN}" \
  -d '{
    "hosted-zone-name": "auto",
    "targets": "example.com",
    "scan-name": "Scan via Hosted Zone",
    "rate": "1000"
  }'
```

Create a scan task for an Explorer group

To distribute a scan task among a group of Explorers, set the `"explorer-group-id"` key to the Explorer group's UUID. The group selects an available Explorer to run the task.

```
curl -X PUT "https://console.runzero.com/api/v1.0/org/sites/${YOUR_SITE_ID}/scan" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${RUNZERO_ORG_TOKEN}" \
  -d '{
    "explorer-group-id": "<YOUR-EXPLORER-GROUP-ID>",
    "targets": "10.0.0.0/8",
    "scan-name": "Scan via Explorer group",
    "rate": "1000"
  }'
```

Additional notes

- The payloads above are minimal working examples.
- You can include any other valid parameters, such as:
    - `"scan-template"`
    - `"tcp-ports"`
    - `"udp-ports"`
    - `"max-hosts"`
    - `"max-duration"`
    - `"probes"`
- Always validate your payload against your organization's scan configuration and API schema.

Export API usage

The Export API provides read-only access to your organization's data, such as assets, sites, and scans. It uses an Export Token (prefixed with `XT`) and suits integrations or reporting that don't modify data.

Set environment variables

```
export RUNZERO_EXPORT_TOKEN="XT-YOUR-EXPORT-TOKEN-HERE"
```

Base endpoint

All export endpoints use the `/api/v1.0/export/org/` path.

Example:

```
https://console.runzero.com/api/v1.0/export/org/assets.json
```

Export all assets (JSON)

Retrieve all assets in JSON format:

```
curl -X GET "https://console.runzero.com/api/v1.0/export/org/assets.json" \
  -H "Authorization: Bearer ${RUNZERO_EXPORT_TOKEN}" \
  -o assets.json
```

Export all assets (CSV)

Retrieve the same data in CSV format:

```
curl -X GET "https://console.runzero.com/api/v1.0/export/org/assets.csv" \
  -H "Authorization: Bearer ${RUNZERO_EXPORT_TOKEN}" \
  -o assets.csv
```

Export sites list

List all sites in your organization:

```
curl -X GET "https://console.runzero.com/api/v1.0/export/org/sites.json" \
  -H "Authorization: Bearer ${RUNZERO_EXPORT_TOKEN}" \
  -o sites.json
```

Export scans

Retrieve a list of completed scans:

```
curl -X GET "https://console.runzero.com/api/v1.0/export/org/scans.json" \
  -H "Authorization: Bearer ${RUNZERO_EXPORT_TOKEN}" \
  -o scans.json
```

Notes on export tokens

- Export tokens are read-only: they cannot modify organization data.
- They can be safely scoped to limit data access.

- Output may be large for organizations with many assets, so consider filtering.
- You can filter exports with the `search` query parameter, for example:

```
curl -X GET "https://console.runzero.com/api/v1.0/export/org/assets.json?search=alive:true" \
  -H "Authorization: Bearer ${RUNZERO_EXPORT_TOKEN}"
```

Additional documentation

The Swagger documentation and the runZero OpenAPI specification describe the individual API calls.

API clients

An API client is a credential made of a client ID and a client secret, exchanged for a short-lived access token through OAuth2. Unlike the API keys and tokens, which are bearer strings with fixed access, an API client has an explicit grant: a set of permissions and the organizations they apply to.

The console URLs in this article use `console.runzero.com` as an example. Use the URL for your runZero deployment instead.

API clients come in two kinds, and the difference matters:

	User-bound	Account-bound
Created from	My API clients	Account API clients
Who can create one	Any user with access to grant	Superusers
Maximum access	The creator's current access	Administrator access to the account
When the creator's access changes	The credential narrows or recovers to match	Unaffected
When the creator is locked or removed	The credential stops working	Unaffected
Use it for	Automation on behalf of a person, AI clients, scripts	Service integrations, CI, unattended jobs

The rule of thumb: if the credential is doing something on your behalf, make it user-bound so it dies with your access. If it belongs to a system rather than a person, make it account-bound so it does not break when someone changes jobs.

Both kinds require a Platform license.

Grants

A grant has two halves, both enforced on every request: permissions and organizations.

Each permission is a verb on an area, as described in How permissions work. A grant that does not include an area denies that area.

Organizations are either a list of specific organizations or every organization. Selecting every organization is an explicit choice. A grant with an empty organization list grants nothing rather than everything.

One grant applies one set of permissions uniformly across the organizations it names. To give a credential different permissions in different organizations, create separate API clients.

Expiration

Every API client has an expiration, chosen when it is created. The default is 30 days, with presets for 90 days and a year, a maximum of 10 years, and an option for no expiration. Prefer a real expiration: a credential nobody has to renew is a credential nobody reviews.

An expired credential stops working rather than being deleted. It no longer exchanges its secret for an access token, and it stays in the list so you can see what expired and when. Access tokens are also capped by the credential's expiration, so a token issued shortly before a credential expires dies with it rather than outliving it.

Expiration is one of the two fields you can change after creation, so a credential that has expired, or is about to, can be given a new expiration without reissuing its secret or its grant. This is the way to renew a working integration. It is not a way to widen one, because permissions and organizations stay fixed.

Credentials created by approving an MCP connection are the exception. They expire 90 days after approval, and while their expiration can be shortened, it can never be moved past that original deadline or removed. Renewing one means reconnecting the application, which creates a new credential. Reviving an expired one counts against the limit of 40 active MCP connections per account.

Creating a user-bound API client

Go to **My API clients** at console.runzero.com/settings/api-clients, which lists only the clients you created, and click to register a new one.

1. Enter an **API client name** and choose an **expiration**. See Expiration.
2. Choose the **organizations**. You can pick specific organizations or every organization you can reach. Choosing every organization records the organizations you can reach at that moment; organizations you gain access to later are not added automatically.
3. Optionally choose a **role**, either as a lock or as a starting point. See Role locks.
4. Set the **Access** permissions. The form only offers permissions your current access allows. If you are an administrator in one organization and a viewer in another, selecting just the first organization lets you grant administrator permissions; selecting both clamps the grant to what the two have in common.
5. Review the summary and save. **Copy the client secret before closing the panel.** It is shown once and cannot be retrieved later.

The creator's access is a continuing ceiling

runZero checks a user-bound credential against its creator every time it authenticates, not just when it was created:

- If you lose access, the credential loses it too, immediately.
- If you regain access, the credential recovers it, up to the permissions it was issued with.
- The credential can never gain access above its issued grant, even if you are later promoted.

- If your account is locked or removed, the credential stops working.

This property makes user-bound credentials safe to hand to an AI client or a script: revoking the person revokes everything they issued.

Creating an account-bound API client

Superusers manage account-bound clients from **API clients** under **Account** in the left navigator, at console.runzero.com/account/api/clients.

The form is the same, with two differences. The ceiling is administrator access across every organization in the account rather than your own access, and selecting every organization stays dynamic, so organizations added to the account later are included.

An account-bound credential keeps the access it was issued regardless of what happens to the person who created it. That is the point, and also the risk: you need to track and rotate it deliberately, because nothing else will retire it.

Role locks

When you choose a role while creating an API client, you pick one of two behaviors:

- **Lock this credential to the role.** The credential's permissions are re-derived from the role every time it authenticates. Editing the role changes what the credential can do. This keeps a fleet of credentials consistent, and lets you revoke a class of access by editing one role. A user-bound locked credential is still capped by its issued grant and by your current access, so a role gaining permissions later does not widen it beyond those.
- **Use the role as a starting point.** The role's permissions are copied into the form so you can adjust them. The result is a fixed grant, and later changes to the role do not apply.

If the locked role is deleted, the credential cannot authenticate until a role with that name exists again. The API clients list shows locked credentials whose role is disabled.

Managing existing clients

Once a credential exists, only its name and expiration can be changed. Its organizations, binding, permissions, and role lock are fixed at creation. To change any of those, create a new client and retire the old one.

You can rotate the secret without changing anything else about the credential, which is the right response to a leaked secret. The credential itself performs the rotation, authenticated with the secret being replaced, and the old secret is invalidated immediately:

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "client_id=<CLIENT_ID>" \
  --data-urlencode "client_secret=<CURRENT_SECRET>" \
  https://console.runzero.com/api/v1.0/account/api/rotate
```

The response includes the new secret. Rotation needs the current secret, so if you have lost it, the only option is to delete the client and create a new one. Rotation is also refused on an expired credential; give it a new expiration first, then rotate.

Creation, metadata changes, deletion, and secret rotation are all recorded in the audit log, with the organizations and permissions the credential was granted. Secrets are never recorded.

Clients created before grants existed

API clients created before explicit grants were introduced show as a legacy grant in the list. They keep the access they have always had, which is unrestricted on the REST API and account-wide on MCP. They are not migrated automatically. Replace them with explicitly scoped clients when you can.

Using an API client

Exchange the client ID and secret for an access token, then use the token as a bearer token:

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "client_id=<CLIENT_ID>" \
  --data-urlencode "client_secret=<CLIENT_SECRET>" \
  https://console.runzero.com/api/v1.0/account/api/token
```

The token is accepted anywhere an account API token is, subject to the grant. A call the grant does not permit is refused, and a call against an organization the grant does not name is refused whether or not the organization was requested explicitly.

API clients can also be presented to the MCP endpoint, where the same grant decides which tools are offered.

The IP address allowlist and the rate limits apply to API clients as they do to API keys.

Credentials created by AI clients

When you connect an AI client to runZero through MCP, approving the connection creates an API client for you. Those credentials appear in the same lists as any other, are user-bound unless a superuser chose otherwise, and can be revoked the same way. They expire 90 days after approval, and that deadline cannot be moved. See Expiration and Connecting through OAuth.

Using the CLI

The runZero Command Line Interface (CLI) provides utility commands and, for licensed users, standalone network scanning.

Scanner

The `scan` command takes the same options as the runZero Explorer and performs much the same way. You can upload its output file, `scan.runzero.gz`, to the runZero Console through the Inventory *Import* menu.

The CLI scanner works best with root privileges on Linux and macOS, and Administrator privileges on Windows. It runs without privileged access, but many probe types are then unavailable. Use `sudo` on Linux and macOS, and an elevated command shell on Windows. On Windows, the CLI looks for an existing `npcap` installation and tries to install it if none is found, unless you pass `--nopcap`.

On physical or virtual Windows hosts with Intel Advanced Matrix Extensions (AMX) enabled, the CLI scanner may become unstable or crash. Disable AMX at the BIOS or host level before you scan. For Windows guests on Hyper-V, enabling **Processor Compatibility** can mitigate the problem.

EDR, antivirus, and other endpoint protection software on Windows can also interfere with the scanner, including its packet capture and process execution. If scans fail or stop unexpectedly, add allowlist exceptions for the runZero and npcap/winpcap components, or use monitor-only policy modes while you validate scans. In some environments, you may need to disable endpoint protection on the host temporarily for stable operation. Scanning on Windows with Intel AMX covers the crashes caused by endpoint protection on AMX systems.



Some parts of the application still use the name "Rumble" for backwards compatibility. We'll update the documentation as they change.

By default the runZero CLI runs a semi-interactive terminal interface and writes several output files to a directory named `runzero-[current-date]`. The `--text` option switches to plain text output. To skip the artifacts and produce only the raw JSON output file, use `--text -o disable --output-raw scan.runzero`.

Pass targets as command-line arguments or in an input file specified with `--input` (or `-i`). Targets can be IPv4 addresses or IPv4 CIDRs, and supported formats include:

- `10.0.0.1`
- `10.0.0.0/24`
- `10.0.0.0/255.255.255.0`
- `10.0.0.1-10.0.0.255`
- `example.com`
- `example.com/24`

For a hostname, each IPv4 address in the response is expanded with the optional mask.

This example downloads and runs the CLI on a Linux x86_64 host. Your URL will differ: the current download links for your organization are on the CLI page of the runZero Console. On a self-hosted console, or in a region other than our US-based SaaS, the download link is under the Deploy navigation menu.

```
$ wget https://console.runzero.com/download/cli/[unique-link]/runzero-cli-linux-amd64.bin
$ chmod +x runzero-cli-linux-amd64.bin
$ sudo runzero scan 192.168.0.0/24 -o output-dir
```

The hexadecimal values in the download URL are specific to your account and organization.

Performance

runZero scans default to 1,000 packets per second in a single pass, which works well on reliable wired networks with no stateful firewall between the scanning system and the destination networks. Change the rate with the `--rate` (or `-r`) option; `10000` is a reasonable maximum for most networks. On slow, unreliable networks, a rate of `300` with `--passes` set to `3` may give better results.

A second parameter, `--max-host-rate`, limits how many packets per second go to each host. Its default of `40` is low, but low-power embedded devices may need it. To scan a few hosts (or a single host) quickly, raise `--max-host-rate` to match `--rate`.

Examples

Scan all 65,535 TCP ports on every host in the `192.168.0.0/24` subnet at 10,000 packets per second:

```
$ sudo runzero scan 192.168.0.0/24 -r 10000 --tcp-ports 1-65535 -o output-dir
```

Scan every host in the 192.168.0.0/24 and 10.0.0.0/24 subnets at 5,000 packets per second:

```
$ sudo runzero scan 192.168.0.0/24 10.0.0.0/24 -r 5000 -o output-dir
```

Scan every host in the 192.168.0.0/24 and 10.0.0.0/8 subnets with a max host rate of 20 packets per host:

```
$ sudo runzero scan 192.168.0.0/24 10.0.0.0/8 --max-host-rate 20 -o output-dir
```

Scan every host in the 192.168.0.0/24 subnet and the domain “example.com” at 7,500 packets per second:

```
$ sudo runzero scan 192.168.0.0/24 example.com -r 7,500 -o output dir
```

Scan every host in the 10.0.0.0/8 subnet and an ASN4 value at the default 1,000 packets per second:

```
$ sudo runzero scan 10.0.0.0/8 asn4:[ID] -o output dir
```

Scan every host in the 192.168.0.0/24 subnet with the max TTL set to 128 at 2,500 packets per second:

```
$ sudo runzero scan 192.168.0.0/24 -r 2,500 --max-ttl 128 -o output-dir
```

Scan the targets listed in an input file:

```
$ sudo runzero scan -i /path/to/input-file.txt -o output dir
```

An example input file:

```
www.example.com
192.168.0.0/24
```

Automatic web screenshots

The `--screenshots` option defaults to `true` and tells runZero to capture a screenshot of every web service it identifies during the scan. Screenshots need a local installation of Google Chrome or Chromium on the system running the Explorer. Each screenshot is reported as a base64 string in the “screenshot.image” field of the service’s scan result.

To disable automatic web screenshots, set `--screenshots` to false (`--screenshots=false`).

Scanner defaults

Standard ports scanned

1	7	9	13	17	19	21	22	23	25	37
42	43	49	53	69	70	79	80	81	82	83
84	85	88	102	105	109	110	111	113	119	123
135	137	139	143	161	179	222	264	280	384	389
402	407	442	443	444	445	465	500	502	512	513
515	523	524	540	541	548	554	587	617	623	631
636	664	689	705	717	743	771	783	789	830	873
888	902	903	910	912	921	949	990	993	995	998
1000	1024	1030	1035	1080	1083	1089	1090	1091	1098	1099
1100	1101	1102	1103	1128	1129	1153	1158	1199	1211	1220
1234	1241	1260	1270	1300	1311	1337	1352	1433	1434	1440
1443	1468	1494	1502	1514	1521	1530	1533	1581	1582	1583
1604	1610	1611	1723	1755	1801	1811	1830	1883	1900	1911
2000	2002	2021	2022	2023	2024	2031	2049	2068	2074	2082
2083	2100	2103	2105	2121	2181	2199	2207	2222	2224	2323
2362	2375	2376	2379	2380	2381	2404	2443	2455	2525	2533
2598	2601	2602	2604	2638	2809	2947	2967	3000	3001	3003
3033	3037	3050	3052	3057	3071	3080	3083	3128	3142	3200
3210	3217	3220	3260	3268	3269	3273	3299	3300	3306	3311
3312	3351	3389	3460	3478	3500	3502	3628	3632	3690	3780
3790	3817	3868	3871	3872	3900	4000	4092	4322	4343	4353
4365	4366	4368	4369	4406	4433	4443	4444	4445	4567	4659

4679 4730 4786 4840 4843 4848 4900 4949 4950 4987 5000 5001
5003 5006 5007 5022 5037 5038 5040 5044 5050 5051 5060 5061
5093 5094 5168 5222 5247 5250 5275 5347 5351 5353 5355 5392
5400 5405 5432 5433 5466 5498 5520 5521 5554 5555 5560 5580
5601 5631 5632 5666 5671 5672 5678 5683 5800 5814 5900 5901
5902 5903 5904 5905 5906 5907 5908 5909 5910 5911 5920 5938
5984 5985 5986 5988 5989 6000 6001 6002 6050 6060 6070 6080
6082 6101 6106 6112 6161 6262 6274 6277 6333 6379 6405 6432
6443 6481 6502 6503 6504 6514 6542 6556 6660 6661 6667 6905
6988 7000 7001 7002 7021 7070 7071 7077 7080 7100 7144 7181
7210 7373 7443 7444 7474 7510 7547 7579 7580 7676 7700 7770
7777 7778 7780 7787 7800 7801 7860 7879 7902 7990 7999 8000
8001 8002 8003 8004 8006 8008 8009 8010 8012 8014 8020 8023
8028 8030 8080 8081 8082 8083 8084 8085 8086 8087 8088 8089
8090 8095 8098 8099 8100 8123 8127 8160 8161 8172 8180 8181
8182 8188 8193 8200 8205 8222 8265 8300 8303 8333 8400 8428
8443 8444 8445 8446 8447 8448 8449 8471 8488 8500 8503 8530
8531 8545 8649 8686 8787 8800 8812 8833 8834 8850 8871 8880
8883 8888 8889 8890 8899 8900 8901 8902 8903 8983 9000 9001
9002 9042 9060 9080 9081 9084 9090 9091 9092 9099 9100 9111
9152 9160 9200 9300 9380 9390 9391 9401 9418 9428 9440 9443
9444 9471 9495 9524 9527 9530 9593 9594 9595 9600 9809 9855
9997 9998 9999 10000 10001 10008 10050 10051 10080 10098 10162 10202
10203 10250 10255 10257 10259 10443 10616 10628 10800 11000 11099 11211
11234 11333 11434 11443 12174 12203 12221 12345 12379 12397 12401 13364
13400 13500 13778 13838 14330 15200 15671 15672 16102 16443 16992 16993
17185 17200 17472 17775 17776 17777 17778 17781 17782 17783 17784 17790
17791 17798 18245 18264 18789 18881 19000 19300 19530 19810 19888 20000
20010 20031 20034 20101 20111 20171 20222 20256 20293 21047 22222 23472
23791 23943 24442 25000 25025 25565 25672 26000 26122 27000 27017 27018
27019 27080 27888 28017 28222 28784 29418 30000 31001 31099 32400 32764
32843 32844 32845 32913 33060 34205 34443 34962 34963 34964 37718 37777
37890 37891 37892 38008 38010 38080 38102 38292 40007 40317 41025 41080
41523 41524 44334 44343 44818 45230 46823 46824 47001 47002 47290 48898
48899 49152 50000 50013 50021 50051 50070 50090 50121 50505 51443 52302
52311 53282 54321 54921 54922 54923 55553 55580 57772 61614 61616 62078
62514 65002 65535

Scan outputs

By default the runZero CLI writes a directory of output files:

- `scan.runzero.gz`: The raw scan data, gzip-compressed. Import or reprocess it with `--import`.
- `assets.jsonl`: The new optimized format for correlated, fingerprinted assets.
- `nmap.xml`: An Nmap XML compatible data file that other security tools can import.
- `urls.txt`: The discovered web services in URL format.
- `protocols.csv`: The protocols with their ports and URLs.
- `assets.html`: A rudimentary HTML report with screenshots.

- `screenshots`: A directory of raw screenshot images, headers in JSON format, and HTML bodies.
- Text lists including `addresses.txt`, `addresses_all.txt`, `hostnames.txt`, and `domains.txt`.

Raw scan data

The CLI stores its raw data as JSONL records in `scan.runzero.gz` in the output directory. An ARP response record looks like this:

```
{
  "type": "result",
  "host": "192.168.0.1",
  "port": "0",
  "proto": "arp",
  "probe": "arp",
  "name": "192.168.0.1",
  "info": {
    "mac": "f0:9f:c2:11:1a:13",
    "macDateAdded": "2014-12-17",
    "macVendor": "Ubiquiti Networks Inc."
  },
  "ts": 1551584126253853200
}
```

The `info` field is a JSON map of strings to strings. Multiple values are joined with the `tab` character (0x09). Elsewhere, a tab is escaped as `\t`, a carriage return as `\r`, a line feed as `\n`, and a null byte as `\x00`. A scan may return more than one record of the same type for the same host if it received multiple responses.

Besides `result` records, the file holds `status` messages, `stats` records, and an initial `config` record with the scan parameters.

runZero Command Line Interface (CLI)

The runZero CLI supports a wide range of commands and options. As well as offline scans, you can run third-party integrations and custom scripts on the command line.

The `--help` output provides basic documentation on the available options.

Most commands below accept the following global flags:

`--verbose`
: Display verbose output.

`--very-verbose`
: Display very verbose output.

Import Censys data files

```
runZero censys [avro files] [targets] [flags]
```

Flags:

`-i, --input-targets string`
: Read search targets from the specified input file

`--output-raw string`
: Write results to the specified output file

Import Censys data from a local database

```
runZero censys-db [path-to-database] [targets] [flags]
```

Flags:

`-i, --input-targets string`
: Read search targets from the specified input file

`--output-raw string`
: Write results to the specified output file

Convert Censys Avro files into a local database

```
runZero censys-db-convert [path-to-avro-directory] [path-to-db-directory] [flags]
```

Flags:

`--shard-max-records int`
: Specify the maximum record count per shard (default 500000)

`--test-mode`
: Replace keys in the source data with random values

`--test-multiplier int`
: Multiply the source data by a given factor (default 1)

Serves a Censys database from a web server

```
runZero censys-db-server [path-to-database] [flags]
```

Flags:

`--port int`
: The TCP port for the web server to listen on (default 55555)

Generate the autocompletion script for the specified shell

```
runZero completion [command]
```

Available commands:

`bash`: Generate the autocompletion script for bash

`fish`: Generate the autocompletion script for fish

`powershell`: Generate the autocompletion script for powershell

`zsh`: Generate the autocompletion script for zsh

Help about any command

```
runZero help [command] [flags]
```

Display license information

```
runZero license [flags]
```

Start a runZero active scan, passive discovery, or integration task

```
runZero scan <options> [targets] [flags]
```

Flags:

`--api-key string`
: Specify the runZero API key

`--api-no-verify`
: Disable TLS verification for API communication

`--api-url string`
: Specify the runZero API server hostname (default "https://console.runzero.com/api/v1.0")

`--arp-fast`
: Enables fast mode by ARP scanning at the scan rate vs host rate

`--atg-ports string`
: The destination ports for ATG probes (default "10001")

`--aws-instances-access-key string`
: The access key for the AWS account

`--aws-instances-assume-role-name string`
: The role to assume for all accounts in the organization for cross-account access

`--aws-instances-delete-stale`
: Automatically delete stale AWS assets

`--aws-instances-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--aws-instances-include-stopped`
: Include assets that are not currently running

`--aws-instances-regions string`
: The comma-separated list of regions for the AWS account

`--aws-instances-secret-access-key string`
: The secret access key for the AWS account

`--aws-instances-service-options string`
: The comma-separated list of services to sync data from (defaults,ec2,elb,elbv2,rds,lambda) (default “defaults”)

`--aws-instances-site-per-account`
: Automatically create a new site per account

`--aws-instances-site-per-vpc`
: Automatically create a new site per VPC

`--aws-instances-token string`
: The session token for the AWS account

`--azure-client-id string`
: The application ID (client ID) for the Azure account

`--azure-client-secret string`
: The client secret for the Azure account

`--azure-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--azure-multi-subscription`
: Access all subscriptions in the directory (tenant) for the Azure account

`--azure-password string`
: The password for the Azure account

`--azure-service-options string`
: The comma-separated list of services to sync data from (defaults,vm,vmss,azsql,cosmos,lb,functionapp) (default “defaults”)

`--azure-site-per-subscription`
: Automatically create a new site per subscription

`--azure-subscription-id string`
: The subscription ID for the Azure account

`--azure-tenant-id string`
: The directory ID (tenant ID) for the Azure account

`--azure-username string`
: The username for the Azure account

`--azuread-client-id string`
: The application ID (client ID) for the Azure account

`--azuread-client-secret string`
: The client secret for the Azure account

`--azuread-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--azuread-filter string`
: An optional filter. Only import devices that match this filter.

`--azuread-include-inactive`
: Include assets that are marked as inactive in the AzureAD account

`--azuread-password string`
: The password for the AzureAD account

`--azuread-service-options string`
: The comma-separated list of services to sync data from (defaults,dev,user,group) (default “defaults”)

`--azuread-tenant-id string`
: The directory ID (tenant ID) for the Azure account

`--azuread-username string`
: The username for the AzureAD account

`--bacnet-ports string`
: The destination ports for BACnet probes (default “46808,47808,48808”)

`-b, --baseline string`
: Use the specified file as an asset baseline for tracking

`--bedrock-ports string`
: The destination ports for Bedrock probes (default “19132”)

`--bjnp-printer-ports string`
: The UDP ports to send Canon printer discovery requests (default “8611”)

`--bjnp-scanner-ports string`
: The UDP ports to send Canon scanner discovery requests (default “8612”)

`--censys-api-url string`
: The API endpoint to use for Censys Search (default “https://search.censys.io”)

`--censys-client-id string`
: The Client ID to use for Censys Search authentication

`--censys-client-secret string`
: The Client Secret to use for Censys Search authentication

`--censys-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--censys-mode string`
: The search mode (assets or query). The assets option queries the scan targets (default “assets”)

`--censys-query string`
: The search string to use in query mode

`--coap-port uint`
: The destination port for CoAP probes (default 5683)

`-c, --config string`
: Specify the config file name to load. It must be in a JSON format

`--cpu string`
: Write a cpu profile after the scan completes

`--crestron-port uint`
: The destination port for Crestron probes (default 41794)

`--crowdstrike-api-url string`
: The URL used for the CrowdStrike account's API access

`--crowdstrike-client-id string`
: The client ID for the CrowdStrike account

`--crowdstrike-client-secret string`
: The client secret for the CrowdStrike account

`--crowdstrike-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--crowdstrike-filter string`
: An optional Falcon Query Language (FQL) filter for imported assets

`--crowdstrike-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--crowdstrike-risks string`
: Minimum risk of imported vulnerabilities (None, Low, Medium, High, Critical) (default "None,Low,Medium,High,Critical")

`--crowdstrike-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default "Info,Low,Medium,High,Critical")

`--custom-integration-entry-function-name string`
: Function to call that will return ImportAssets (default "main")

`--custom-integration-id string`
: UUID of custom integration from console

`--custom-integration-script-args string`
: Arguments for the script

`--custom-integration-script-kwargs string`
: Keyword arguments for the script

`--custom-integration-script-source string`
: Source code of the custom integration script

`--dahua-dhip-ports string`
: The destination ports for Dahua DHIP discovery probes (default "37810")

`--defender365-client-id string`
: The application ID (client ID) for the Azure account

`--defender365-client-secret string`
: The client secret for the Azure account

`--defender365-exclude-notonboarded`
: Exclude assets that have not been fully onboarded

`--defender365-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--defender365-filter string`
: Exclude assets using a Graph API $filter string

`--defender365-include-inactive`
: Include assets that have stopped reporting to the Microsoft 365 Defender service

`--defender365-tenant-id string`
: The directory ID (tenant ID) for the Azure account

`--disabled-probes string`
: Specifically exclude these probes, comma-delimited

`--dnp3-address-probe-timeout int`
: Time limit (in seconds) for DNP address discovery. (default 30)

`--dnp3-banner-address-discovery string`
: One of ‘require’, ‘prefer’, or ‘ignore’. (default “ignore”)

`--dnp3-destination-address-discovery-range string`
: A numeric range of addresses to attempt to discover. (default “0-32”)

`--dnp3-explorer-address int`
: Source DNP3 address for the explorer. (default -1)

`--dns-disable-google-myaddr`
: Disables resolution of upstream DNS via Google myaddr service

`--dns-disable-meraki-detection`
: Disables detection of Meraki DNS interception

`--dns-port uint`
: The destination port for DNS probes (default 53)

`--dns-resolve-name string`
: The target hostname for DNS queries (‘off’ to disable) (default “www.google.com”)

`--dns-trace-domain string`
: The subdomain to use for trace requests (‘off’ to disable) (default “helper.rumble.network”)

`--dtls-ports string`
: The destination ports for DTLS probes (default “443,3391,4433,5246,5349,5684”)

`--echo-report-errors`
: Report errors from intermediate in-scope hosts

`--ethernetip-cip-enumeration-method string`
: (BETA) Set this to the preferred CIP enumeration method. (default “none”)

`--ethernetip-udp-ports string`
: The destination ports for EtherNet/IP UDP probes (default “44818”)

`--exclude string`
: Specify scan exclusions

`--excludefile string`
: Read exclusions from an input file

`--filter-base64`
: Filter base64-encoded fields

`-f, --fingerprints string`
: Use the specified directory as an alternate fingerprint database

`--fingerprints-debug`
: Enable debug output for the fingerprint processor

`--fins-port uint`
: The destination port for FINS probes (default 9600)

`--gcp-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--gcp-key-path string`
: Path to GCP service account key file

`--gcp-service-options string`
: The comma-separated list of services to sync data from (defaults,vm,lb,cloudsql) (default “defaults”)

`--gcp-site-per-project`
: Automatically create a new site per project

`--genudp-payload-base64 string`
: The generic udp payload as base64

`--genudp-payload-hex string`
: The generic udp payload as hexadecimal

`--genudp-payload-text string`
: The generic udp payload as plain text

`--genudp-ports string`
: The destination ports for the generic udp probe

`--googleworkspace-client-email string`
: The email address of the service account

`--googleworkspace-client-id string`
: The ID of the service account

`--googleworkspace-customer-id string`
: An optional customer ID for multi-tenant environments (default “my_customer”)

`--googleworkspace-delegate string`
: The email address of an admin account with directory access

`--googleworkspace-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--googleworkspace-private-key string`
: The PEM encoded private key

`--googleworkspace-private-key-id string`
: The ID of the private key

`--googleworkspace-project-id string`
: The project ID of the service account

`--googleworkspace-service-options string`
: The comma-separated list of services to sync data from (defaults,chromeos,mobile,endpoint,user,group) (default “defaults”)

`--goroutines string`
: Write a goroutine dump after the scan completes

`--heap string`
: Write a heap profile after the scan completes

`--hiddiscoveryd-port uint`
: The destination port for HID discoveryd probes (default 4070)

`--host-ping`
: Only scan hosts that respond to a ping scan using the host-ping settings

`--host-ping-max-attempts int`
: Set the maximum number of attempts for each probe (default 2)

`--host-ping-max-ttl int`
: Set the default TTL on host-ping probe packets (default 255)

`--host-ping-passes int`
: Set the number of passes for the host-ping phase (default 1)

`--host-ping-probes string`
: Launch a subset of the probes for the host-ping, comma-delimited (default “arp,echo,syn,connect,netbios,snmp,ntp,sunrpc,ike,openvpn,mdns”)

`--host-ping-tcp-ports string`
: The list of TCP ports to host-ping using the syn and connect probes (default “22,80,135,179,443,3389,5040,7547,62078”)

`--host-ping-tos int`
: Set the default ToS on host-ping probe packets

`--igel-discovery-ports string`
: The destination ports for IGEL discovery probes (default “30005”)

`--ike-port uint`
: The destination port for IKE probes (default 500)

`-I, --import stringArray`
: Import existing scan data from the specified input files (‘scan.rumble’ format)

`--import-pcap stringArray`
: Import pcap packet capture from the specified input files (‘.pcap’ or ‘.pcapng’ format)

`-i, --input-targets string`
: Read scan targets from the specified input file

`--insightvm-api-url string`
: The URL used for the InsightVM account’s API access

`--insightvm-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--insightvm-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--insightvm-insecure`
: Set this to true to authenticate to untrusted endpoints (self-signed or no IP SAN) (default true)

`--insightvm-password string`
: The password for the InsightVM account

`--insightvm-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--insightvm-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--insightvm-thumbprints string`
: A set of IP=SHA256:B64HASH pairs to trust for authentication

`--insightvm-username string`
: The username for the InsightVM account

`--insightvmcloud-host string`
: The host for the InsightVM Cloud API

`--insightvmcloud-api-key string`
: The API key for the InsightVM Cloud account

`--insightvmcloud-exclude-unnknown`
: Exclude assets that cannot be merged into an existing asset

`--insightvmcloud-exclude-tags`
: Do not collect asset tags

`--insightvmcloud-exclude-vulns-import`
: Do not collect asset vulnerabilities

`--insightvmcloud-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--insightvmcloud-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--intune-client-id string`
: The application ID (client ID) for the Azure account

`--intune-client-secret string`
: The client secret for the Azure account

`--intune-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--intune-filter string`
: An optional filter. Only import devices that match this filter.

`--intune-password string`
: The password for the Intune account

`--intune-tenant-id string`
: The directory ID (tenant ID) for the Azure account

`--intune-username string`
: The username for the Intune account

`--ipmi-port uint`
: The destination port for IPMI probes (default 623)

`--ipp-browse-port uint`
: The destination port for IPP-browse probes (default 631)

`--iscsi-discover`
: Enable iSCSI target discovery probe (default true)

`--kerberos-port uint`
: The destination port for kerberos probes (default 88)

`--knxnet-ports string`
: The destination ports for knxnet probes (default “3671”)

`--l2t-port uint`
: The destination port for L2T probes (default 2228)

`--l2tp-ports string`
: The destination ports for L2TP probes (default “1701”)

`--lantronix-port uint`
: The destination port for Lantronix probes (default 30718)

`--layer2-add-targets`
: Set this false to skip scanning discovered targets (default true)

`--layer2-force`
: Set this to true to force discovery even without local targets

`--layer2-max-retries uint`
: The desired number of retries (default 3)

`--layer2-tcp-ports string`
: The TCP ports to ping for local device discovery (default “22,80,135,179,443,3389,5040,7547,62078”)

`--layer2-udp-trace-port uint`
: The UDP port number to use for UDP trace requests (default 9)

`--ldap-base-dn string`
: The base DN used for LDAP searches

`--ldap-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--ldap-insecure`
: Set this to true to authenticate to untrusted endpoints (self-signed or no IP SAN)

`--ldap-legacy-tls`
: Set this to true to authenticate over legacy TLS versions (< 1.2)

`--ldap-password string`
: The password for the LDAP account

`--ldap-service-options string`
: The comma-separated list of services to sync data from (defaults,computer,user,group) (default “defaults”)

`--ldap-thumbprints string`
: A set of IP=SHA256:B64HASH pairs to trust for authentication

`--ldap-url string`
: The URL used for the LDAP server

`--ldap-username string`
: The username for the LDAP account

`--max-attempts int`
: Set the maximum number of attempts for each probe (default 3)

`-G, --max-group-size int`
: Set the maximum number of targets to process in each group (default 4096)

`-R, --max-host-rate int`
: Set the maximum packet rate per target (including ARP broadcast) (default 40)

`--max-scan-duration int`
: Set the maximum scan duration in seconds before aborting

`--max-sockets int`
: Set the maximum number of concurrent sockets (default 2048)

`--max-ttl int`
: Set the default TTL on probe packets (default 255)

`--mdns-port uint`
: The destination port for MDNS probes (default 5353)

`--mecm-database-connection-string string`
: The connection string for your MECM Microsoft SQL Server database

`--mecm-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--memcache-port uint`
: The destination port for memcached probes (default 11211)

`--meraki-api-key string`
: The access key for the Meraki.io account

`--meraki-api-url string`
: The URL used for the Meraki.io account’s API access (default “https://api.meraki.com/api/v1")

`--meraki-exclude-no-vlan-clients`

`--meraki-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--meraki-excluded-ssids string`

`--meraki-excluded-vlans string`

`--meraki-networks string`
: An optional list of network names or IDs. Only import devices in the specified networks.

`--meraki-organizations string`
: An optional list of organization names or IDs. Only import devices in the specified organizations.

`--miradore-api-key string`
: The API key for the Miradore account

`--miradore-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--miradore-hostname string`
: The Miradore web console hostname (url)

`--modbus-identification-level string`
: Identification level, one of 'basic', 'regular', or 'extended'. (default "regular")

`--mssql-port uint`
: The destination port for MSSQL Browser probes (default 1434)

`--nameservers string`
: One or more nameservers to use for DNS resolution

`--natpmp-port uint`
: The destination port for NATPMP probes (default 5351)

`--nessus-access-key string`
: The access key for the Nessus Professional account

`--nessus-api-url string`
: The URL used for the Nessus Professional account's API access

`--nessus-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--nessus-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--nessus-insecure`
: Set this to true to authenticate to untrusted endpoints (self-signed or no IP SAN) (default true)

`--nessus-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default "None,Low,Medium,High,Critical")

`--nessus-secret-key string`
: The secret key for the Nessus Professional account

`--nessus-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default "Info,Low,Medium,High,Critical")

`--nessus-thumbprints string`
: A set of IP=SHA256:B64HASH pairs to trust for authentication

`--netbios-port uint`
: The destination port for NetBIOS Name Service probes (default 137)

`--netbox-api-key string`
: The access key for the NetBox instance

`--netbox-api-url string`
: The URL to the NetBox instance

`--netbox-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--netbox-include-no-ip`
: Include NetBox assets that have no associated IP addresses

`--netbox-include-site-names`
: Comma-separated list of NetBox site names to include assets from

`--netbox-include-cidrs`
: Comma-separated list of CIDRs to include assets from

`--netbox-field-mappings`
: Map NetBox custom fields to runZero attributes. See the NetBox documentation for more information

`--nowait`
: Exit the user interface immediately upon completion

`--ntp-port uint`
: The destination port for NTP probes (default 123)

`--openvpn-ports string`
: The destination ports for OpenVPN probes (default “1194”)

`--oracledb-fingerprint`
: Enable Oracle DB version fingerprinting using a TNS connect sequence (default true)

`-o, --output string`
: Output directory for scan results and analysis (‘disable’ to skip)

`--output-raw string`
: Set the raw output file for scan data

`--overwrite`
: Overwrite and replace the output directory if it already exists

`--passes int`
: Set the number of passes for each probe (default 1)

`--pca-port uint`
: The destination port for PCAnywhere probes (default 5632)

`--pcworx-ports string`
: The destination ports for PCWORX probes (default “1962”)

`--probes string`
: Launch a subset of the probes, comma-delimited (default “defaults”)

`--psdisco-ports string`
: The destination ports for playstation discovery probes (default “987,9302”)

`--qualys-api-url string`
: The URL used for the Qualys account’s API access

`--qualys-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--qualys-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--qualys-include-unscanned`
: Include assets that have not been assessed for vulnerabilities

`--qualys-password string`
: The password for the Qualys account

`--qualys-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--qualys-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--qualys-tags string`
: An optional list of tags. Only import devices that match any of the specified tags

`--qualys-network-ids string`
: An optional list of network IDs. Only import devices that match any of the specified network IDs

`--qualys-username string`
: The username for the Qualys account

`-r, --rate int`
: Set the maximum packet rate for the overall scan (default 1000)

`--rdns-max-concurrent int`
: The maximum number of concurrent DNS lookups (default 64)

`--rdns-timeout uint`
: The DNS PTR lookup timeout in seconds (default 3)

`--rpcbind-port uint`
: The destination port for RPCBind probes (default 111)

`--rpcbind-port-nfs uint`
: The destination port for NFS probes (default 2049)

`--s7comm-request-extended-information`
: If true, request extended device information.

`--sadp-ports string`
: The destination ports for Hikvision SADP discovery probes (default “37020”)

`--sample-duration string`
: Specify the duration in seconds to sample network traffic (or ‘0’ for non-stop) (default “300”)

`--sample-excludes string`
: Specify host exclusions

`--sample-interfaces string`
: Specify a comma-separated list of network interfaces (or 'all' for everything)

`--sample-targets string`
: Specify the discovery scope (default "10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 169.254.0.0/16")

`-S, --screenshots`
: Capture screenshots from scan target web services (default true)

`--sentinelone-api-url string`
: The URL used for the SentinelOne account's API access

`--sentinelone-api-key string`
: The API key for the SentinelOne accounts API access

`--sentinelone-exclude-software-import`
: Do not import software records from the SentinelOne account

`--sentinelone-exclude-vulnerability-import`
: Do not import vulnerability records from the SentinelOne account

`--sentinelone-severities string`
: Severity levels of imported vulnerabilities (False Positive, LOW, MEDIUM, HIGH, CRITICAL) (default "False Positive,LOW,MEDIUM,HIGH,CRITICAL")

`--sentinelone-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--servicetag-port uint`
: The destination port for Solaris Service Tag probes (default 6481)

`--shodan-api-key string`
: The key used for the Shodan account's API access

`--shodan-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--shodan-mode string`
: The search mode (assets or query). The assets option queries the scan targets (default "assets")

`--shodan-query string`
: The search string to use in query mode

`--sip-port uint`
: The destination port for SIP probes (default 5060)

`--snmp-comms string`
: The comma-separated list of SNMP v1/v2c communities (default "public,private")

`--snmp-disable-bulk`
: If true, do not use bulk walking operations

`--snmp-max-repetitions uint`
: The maximum number of repetitions in a bulk walk operation (default 16)

`--snmp-max-retries int`
: The maximum number of retries for an SNMP operation (default 1)

`--snmp-poll-interval uint`
: The minimum number of seconds between polling each host after initial discovery (default 300)

`--snmp-port uint`
: The destination port for SNMP probes (default 161)

`--snmp-timeout uint`
: The maximum number of seconds for each individual SNMP operation (default 5)

`--snmp-v3-auth-passphrase string`
: The authentication passphrase

`--snmp-v3-auth-protocol string`
: The authentication protocol (none, md5, sha, sha224, sha256, sha384, sha512) (default “none”)

`--snmp-v3-context string`
: The optional SNMP v3 context to supply

`--snmp-v3-privacy-passphrase string`
: The privacy passphrase

`--snmp-v3-privacy-protocol string`
: The privacy protocol (none, des, aes, aes192, aes256, aes192c, aes256c) (default “none”)

`--snmp-v3-username string`
: The username to use for SNMP v3 authentication

`--snmp-walk-timeout uint`
: The maximum number of seconds for each SNMP walk operation (default 60)

`--ssdp-port uint`
: The destination port for UPnP/SSDP probes (default 1900)

`--ssh-fingerprint`
: Enable fingerprinting using partial authentication (default true)

`--ssh-fingerprint-username string`
: The username to use for partial authentication SSH fingerprinting (default “*STATUS*”)

`--steam-ports string`
: The destination ports for Steam discovery probes (default “27036”)

`--subnet-ping`
: Only scan subnets that have at least one active response using the subnet-ping settings

`--subnet-ping-max-attempts int`
: Set the maximum number of attempts for each probe (default 1)

`--subnet-ping-max-ttl int`
: Set the default TTL on subnet-ping probe packets (default 255)

`--subnet-ping-mode string`
: Set the subnet-ping discovery profile: auto (default “auto”)

`--subnet-ping-net-size int`
: Set the subnet size to use for the subnet ping (default 256)

`--subnet-ping-passes int`
: Set the number of passes for the subnet-ping phase (default 1)

`--subnet-ping-probes string`
: Launch a subset of the probes for the subnet-ping, comma-delimited (default "arp,echo,syn,connect,netbios,snmp,ntp,sunrpc,ike,openvpn,mdns")

`--subnet-ping-sample-rate int`
: Set the sample rate of addresses within each subnet as a percentage (default 4)

`--subnet-ping-tcp-ports string`
: The list of TCP ports to subnet-ping using the syn and connect probes (default "22,80,135,179,443,3389,5040,7547,62078")

`--subnet-ping-tos int`
: Set the default ToS on subnet-ping probe packets

`--syn-disable-bogus-filter`
: Disable bogus service detection and filtering

`--syn-forwarding-check`
: Perform an IP forwarding check as part of the scan (default true)

`--syn-forwarding-check-target string`
: An external IPv4 address for the forwarding check (default:runzero) (default "13.248.161.247")

`--syn-max-retries uint`
: The maximum number of retries trace and SYN requests (default 2)

`--syn-report-resets`
: Set this to true to report RST responses (default true)

`--syn-reset-sessions`
: Reset middle-box/firewall sessions automatically (default true)

`--syn-reset-sessions-delay uint`
: Minimum delay in milliseconds between a SYN and a session reset

`--syn-reset-sessions-limit uint`
: Maximum number of in-flight sessions before forcing session resets (default 50)

`--syn-traceroute`
: Perform a multi-protocol traceroute as part of the scan (default true)

`--syn-udp-trace-port uint`
: The UDP port number to use for UDP trace requests (default 9)

`--tanium-api-token string`
: The API token for the Tanium account

`--tanium-api-url string`
: The URL used for the Tanium account's API access

`--tanium-computer-groups string`
: Filter endpoints to members of the specified computer groups

`--tanium-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--tanium-severities string`
: Severity levels of imported vulnerabilities (Low, Medium, High, Critical) (default “Low,Medium,High,Critical”)

`--tcp-excludes string`
: The list of TCP ports to always exclude

`-p, --tcp-ports string`
: The list of TCP ports scan using the syn and connect probes (see below for default)

`--tcp-skip-protocol`
: Set this to skip protocol detection on TCP ports

`--tenable-access-key string`
: The access key for the Tenable.io account

`--tenable-api-url string`
: The URL used for the Tenable.io account’s API access

`--tenable-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--tenable-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--tenable-include-unscanned`
: Include assets that have not been assessed for vulnerabilities

`--tenable-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--tenable-secret-key string`
: The secret key for the Tenable.io account

`--tenable-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--tenable-sources string`
: An optional list of sources. Only import devices that match the specified sources

`--tenable-tags string`
: An optional list of tags. Only import devices that match the specified tags

`--tenablesecuritycenter-access-key string`
: The access key for the Tenable Security Center account

`--tenablesecuritycenter-api-url string`
: The URL used for the Tenable Security Center account’s API access

`--tenablesecuritycenter-batch-size string`
: The number of records to request at a time. (between 2000 and 10000) (default “2000”)

`--tenablesecuritycenter-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--tenablesecuritycenter-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--tenablesecuritycenter-insecure string`
: Set this to true to authenticate to untrusted endpoints (self-signed or no IP SAN)

`--tenablesecuritycenter-query-id string`
: The ID of an existing vulnerability query in the Tenable Security Center account

`--tenablesecuritycenter-query-mode string`
: Set to ‘filters’ to provide ‘severities’ and ‘risks’ values to import. Set to ‘query-id’ to provide a value for ‘query-id’. (default “filters”)

`--tenablesecuritycenter-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--tenablesecuritycenter-secret-key string`
: The secret key for the Tenable Security Center account

`--tenablesecuritycenter-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--tenablesecuritycenter-sync-since string`
: Specify an initial date to sync data from.

`--tenablesecuritycenter-thumbprints string`
: A set of IP=SHA256:B64HASH pairs to trust for authentication

`--text`
: Force text-only mode (no console ui)

`--tftp-ports string`
: The destination ports for TFTP probes (default “69”)

`--tos int`
: Set the default ToS on probe packets

`--ubnt-port uint`
: The destination port for Ubiquiti probes (default 10001)

`--upload`
: Automatically upload results to the runZero Console

`-u, --upload-site string`
: Specify the Site ID or Name to upload the raw scan results to if –upload is specified (default “Primary”)

`--vmware-insecure`
: Set this to true to authenticate to untrusted endpoints (self-signed or no IP SAN) (default true)

`--vmware-password string`
: The password to use for VMware SDK authentication (read-only)

`--vmware-thumbprints string`
: A set of IP=SHA256:B64HASH pairs to trust for authentication

`--vmware-username string`
: The username to use for VMware SDK authentication (read-only)

`--webmin-ports string`
: The destination ports for webmin probes (default “10000”)

`--wiz-api-url string`
: The URL used for the Wiz account’s API access

`--wiz-auth-url string`
: The URL used for the Wiz account’s authentication

`--wiz-client-id string`
: The client ID for the Wiz account

`--wiz-client-secret string`
: The client secret for the Wiz account

`--wiz-exclude-unknown`
: Exclude assets that cannot be merged into an existing asset

`--wiz-fingerprint-only`
: Import vulnerabilites for fingerprinting purposes only

`--wiz-include-unscanned`
: Include assets that have not been assessed for vulnerabilities (default true)

`--wiz-risks string`
: Risk levels of imported vulnerabilities (None, Low, Medium, High, Critical) (default “None,Low,Medium,High,Critical”)

`--wiz-severities string`
: Severity levels of imported vulnerabilities (Info, Low, Medium, High, Critical) (default “Info,Low,Medium,High,Critical”)

`--wlan-list-poll-interval uint`
: The minimum number of seconds between polls of the access point list (default 300)

`--wsd-port uint`
: The destination port for WSD probes (default 3702)

`--xdmcp-ports string`
: The destination ports for XDMCP probes (default “177”)

Run a custom starlark script

```
runZero script --filename filename [--args a] [--args b] [--kwargs a=b] [--kwargs b=c] [flags]

runZero script [command]
```

Available commands:

`repl`: Run a custom starlark script in a REPL

Flags:

`--args stringArray`
: args to pass into script

`-f, --filename string`
: file of script to load and run

`--kwargs stringToString`
: kwargs to pass into script (default [])

Upgrade to the latest version of the runZero Scanner

```
runZero upgrade [flags]
```

Flags:

`--force`
: Apply the update without checking the version

Perform an internal signature verification

```
runZero verify [flags]
```

Print the version number of runZero

```
runZero version [flags]
```

Model Context Protocol

runZero supports the Model Context Protocol (MCP) at the `/mcp` endpoint, over streaming HTTP. Through it, AI clients query runZero's security inventory directly and answer questions about your network assets and vulnerabilities.

The `console.runzero.com` URLs in this article are examples. Use the endpoint for your runZero deployment instead.

The MCP server offers the same tools as the console's AI threads, so a question you can ask in the console can also be asked from an external client. See Available tools.

Models that were not built for tool use or function calling struggle with MCP. They may lack the mechanisms or training to interpret tool schemas, decide when to call a tool, or parse its output reliably. Forcing MCP onto such a model is likely to produce frequent errors and hallucinations, or leave it unable to use the tools effectively.

How much data runZero holds affects the quality of responses. Larger models tend to handle larger data sets better, and prompting for exactly what you need, with filtering criteria, helps the model return the information you asked for.

MCP terms

AI client

The *interface and communication layer* between a human user (or another system) and an AI model. Its job is to carry input to the model and present the model's output, and it is often *reactive* to user input.

AI agent

A more capable version of that client, one that orchestrates multiple steps and makes its own decisions. Asked to "Summarize this document," an agent might first decide it needs a "file reading tool" to reach the document, have the LLM summarize what it retrieved, and then *proactively* suggest sharing the summary by email without being told to.

Model Context Protocol

From the MCP specification:

> Model Context Protocol (MCP) is an open protocol that enables seamless integration between LLM applications and external data sources and tools. Whether you're building an AI-powered IDE, enhancing a chat interface, or creating custom AI workflows, MCP provides a standardized way to connect LLMs with the context they need.

Authentication

You can connect in two ways, depending on what your AI client supports.

OAuth. The client sends you to the runZero console, you approve what it may reach, and runZero issues it a scoped credential of its own. Nothing is copied and pasted, the credential is limited to what you approved, and it expires. This is the better option, and most current MCP clients support it. See Connecting through OAuth.

A credential you supply. You create an API key or an API client yourself and configure it in the client, usually as a request header. Use this when the client cannot do OAuth, or when the connection belongs to a system rather than a person. See Presenting a credential directly.

What a credential can reach

The credential decides the access. The MCP server offers a client only the tools its credential permits, so a read-only credential is not offered start_scan and cannot call it.

Credential	Prefix	Organizations	Access
OAuth connection	-	What you approved	What you approved, never more than your own access
API client	-	The client's grant	The client's grant
Account API token	`CT`	Every organization in the account	Read-write
Organization API token	`OT`	The token's organization	Read-write
Export token	`ET`	The token's organization	Read-only

Download tokens (`DT`) are refused, because they do not map onto an inventory scope.

Connecting through OAuth

Point the client at `https://console.runzero.com/mcp` and let it authenticate. The client discovers runZero as its authorization server, registers itself, and opens a browser. You sign in to the console if you are not already signed in, and land on an approval page.

The approval page tells you which application is asking and where the authorization code will be sent, then asks two questions:

- **What this application may reach.** Read one organization, read and write one organization, or, for superusers, read and write every organization in the account. Only organizations where you can manage API clients are offered.
- **A name for the credential**, so you can tell your connections apart later.

Superusers get a third choice, **Credential binding**, between a user-bound credential tied to their own access and an account-bound credential that is not. Everyone else gets a user-bound credential.

Approving creates an API client on your account and hands the application an authorization code, which it exchanges for an access token. Denying creates nothing.

What the connection can and cannot do

The credential an approval creates is deliberately narrow:

- It **cannot** manage API credentials, scan credentials, console users, or account settings, whatever you approved and whatever your own access is.
- It is limited to what you approved, and, when user-bound, to your current access. If your access narrows, the connection narrows with it. If your account is locked or removed, the connection stops working.
- It **expires 90 days** after approval. That deadline cannot be moved, so renewing the connection means reconnecting the application, which creates a new credential. See Expiration.
- Its access token is valid only for the MCP endpoint; the REST API refuses it.

Approvals are capped at 40 active connections per account.

Managing and revoking connections

A connection created this way is an API client like any other. Find it under My API clients, or, for account-bound ones, on the Account API clients page, and delete it to revoke access immediately. Approvals and the credentials they create are recorded in the audit log.

Who can approve a connection

To approve a connection you need permission to manage API credentials (`tokens:manage`) in at least one organization, which the Administrator role grants. Without it, the approval page tells you so and nothing is created. Approving for the whole account requires a superuser with a Platform license.

Presenting a credential directly

For clients that cannot do OAuth, or for unattended connections, create a credential yourself and configure it in the client. runZero accepts three shapes.

API key as a bearer token. The simplest option, for any client that can send a custom header:

```
Authorization: Bearer CT********
```

API key as an OAuth secret. For a client that can only be configured with OAuth client credentials. Use the client ID `token` and put the API key in the client secret. Send it as HTTP

basic auth against the MCP endpoint, or exchange it at the token endpoint first, which validates the key and returns it as the access token:

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "client_id=token" \
  --data-urlencode "client_secret=CT********" \
  https://console.runzero.com/api/v1.0/account/api/token
```

Registered API client. Use the client ID and secret of an API client you created. Send them as HTTP basic auth against the MCP endpoint, or exchange them at the token endpoint for an access token the way the REST API does. The client's grant decides what the connection can reach, so this is the way to give an unattended integration a narrow, long-lived MCP connection.

OAuth endpoints

When a request carries no usable credential, runZero answers with a 401 whose `WWW-Authenticate` header points at the protected resource metadata, so a client can find its way without being configured by hand.

Endpoint	Purpose
`/.well-known/oauth-protected-resource`	Names the console as the authorization server for `/mcp` (RFC 9728). Also served at `/.well-known/oauth-protected-resource/mcp`.
`/.well-known/oauth-authorization-server`	Advertises the endpoints, grant, and PKCE support (RFC 8414).
`/oauth/register`	Dynamic client registration (RFC 7591), for clients with no client ID metadata document.
`/oauth/authorize`	The approval page.
`/api/v1.0/account/api/token`	The token endpoint.

runZero offers the authorization code grant with PKCE using S256, and supports both dynamic client registration and client ID metadata documents, so a client can register itself without anyone provisioning it first. Registration issues a public client ID and stores nothing. Redirect URIs must use HTTPS, or HTTP on a loopback address. Authorization codes are single use, and no refresh token is issued.

The client credentials exchange described in Presenting a credential directly remains available for API keys and API clients, but it is not advertised as an OAuth grant.

Organization scope

Every inventory tool takes an optional `organization_ids` argument. Omit it and the call covers the whole scope of the credential: the organizations named in an API client's grant or approved for an OAuth connection, every organization in the account for an account API token, and the single organization encoded into an organization key or export token.

An organization-scoped credential is narrowed to its own organization whether or not `organization_ids` was passed. Naming an organization outside the credential's scope returns an error rather than substituting a different one.

Clients

MCP standardizes how AI agents talk to external tools, but how each client implements that connection can vary significantly with its design and intended use.

Connecting to remote MCP servers

Point the client at `https://console.runzero.com/mcp` and select Streamable HTTP as the transport.

If the client supports OAuth, that is all the configuration it needs. It will send you to the console to approve the connection, as described in Connecting through OAuth. If it does not, configure a credential yourself as described in Presenting a credential directly.

Claude

Claude Desktop and Claude.ai connect to remote servers as a custom connector rather than through `claude_desktop_config.json`.

On a Free, Pro, or Max plan, go to **Settings > Connectors** and choose **Add custom connector**. On a Team or Enterprise plan, an owner adds it from **Organization settings > Connectors** with **Add > Custom**, and members then connect to it individually.

Enter `https://console.runzero.com/mcp` as the server URL and leave **Authentication** on **Always required**, which Claude pre-fills because the MCP endpoint answers an anonymous request with a 401. Leave the **OAuth client** setting on its default; Claude registers itself with runZero automatically.

The first time you use the connector, Claude opens the console in a browser. Sign in if you are not already signed in, choose what the connector may reach, name it, and approve. Claude finishes the connection and the connector is ready to use.

Because each member of a Team or Enterprise organization connects the shared connector separately, each one approves their own credential and reaches only what their own runZero access permits. Nobody sees another member's inventory through the connector, and revoking one person's runZero access revokes their connection without affecting anyone else's.

The credential expires after 90 days. Reconnecting the connector in Claude creates a new one.

Anthropic documents the dialog field by field in Third party connectors with remote MCP.

Claude Code

Add the server with the `http` transport and no credential:

```
claude mcp add --transport http runzero https://console.runzero.com/mcp
```

The first call to the server opens a browser for approval. Confirm the result with `claude mcp list`, which reports whether each configured server connected, and use `/mcp` inside Claude Code to check or re-run authentication.

To use a credential you created yourself instead, pass it as a header, which turns off the OAuth flow for that server:

```
claude mcp add --transport http runzero https://console.runzero.com/mcp --header "Authorization: Bearer CT********"
```

Gemini CLI

The Gemini CLI also supports MCP, configured mainly through its `settings.json` file.

The user-scoped `settings.json` should be at `~/.gemini/settings.json` (where `~` is your home directory).

You can find or create the system-wide `settings.json` at:

- Linux: `/etc/gemini-cli/settings.json`
- macOS: `/Library/Application\ Support/GeminiCli/settings.json`
- Windows: `C:\ProgramData\gemini-cli\settings.json`

For more on configuration, see the project's configuration documentation.

Replace the token value below with one created as described in Presenting a credential directly, and leave the other settings Gemini wrote (theme, selected authentication type, and so on) as they are.

```
{
  "mcpServers": {
    "runzero": {
      "httpUrl": "https://console.runzero.com/mcp",
      "headers": {
        "Authorization": "Bearer CT********"
      }
    }
  }
}
```

Other options

You should be able to adapt the configurations above to other tools. Most of these clients support OAuth, in which case no credential needs to be configured at all. Follow each client's

documentation for remote or streaming-HTTP servers rather than its `stdio` instructions:

- VSCode
- OpenAI Codex
- Cursor
- Windsurf
- Zed Editor

Available tools

MCP and AI threads run the same tools against the same inventory. Everything below is shared with AI threads except `start_scan`, which only MCP exposes. MCP offers the whole set at once; inside the console, each turn gets only the tools its task needs.

Every tool takes an optional `organization_ids` argument; see Organization scope.

Inventory search

Live queries against the inventory, written in runZero search syntax.

- **query_assets** runs an asset query and returns the exact match count plus an optional risk-ranked sample. Views: `count`, `compact`, `lifecycle`, `security`, `detail`.
- **query_asset_counts** counts up to eight labeled asset queries in one batch, with no rows. Use it for comparisons and category breakdowns instead of repeated `query_assets` calls.
- **query_services** searches the service inventory, including TLS fields.
- **query_certificates** searches the certificate inventory, sorted by earliest validity end.
- **query_tasks** searches scan and integration tasks. The `series` view returns up to 2,000 runs with timings and result counts for charting.
- **query_inventory** searches any other inventory entity: `software`, `software-group`, `vulnerability` (with CVE, CVSS, and remediation text), `vulnerability-group`, `wireless`, `finding`, `event`, `user`, `group`, `task`, `issue`, `goal`, `credential`, `query`, and `site`.
- **get_attributes** returns selected host-level attributes for 1-20 assets in one batch.
- **list_organizations** lists the organizations the credential can reach, with summary asset counts.

Writing queries

- **get_field_definitions** returns the searchable fields for an entity or integration source, each with a description and example values. Call it with no argument for the index of areas and the integration sources present in the inventory.
- **get_query_syntax** returns the query-writing documentation for one entity: operators, worked examples, and that entity's fields. It covers every entity area `get_field_definitions` lists.
- **translate_query** turns a plain-language phrase into runZero syntax. `entity: "asset"` with `question: "find all cisco devices with ssh exposed"` returns `(os:cisco or hw:cisco or newest_mac_vendor:cisco) and protocol:ssh`. The query is compiled but not executed, so it stays fast on large inventories; pass the result to `query_assets` or `query_inventory` to run it.

`translate_query` calls the account's configured model, so it needs AI to be enabled and is subject to the same entitlement and budget as the console. When AI is unavailable the tool is not offered.

Detection coverage and rapid response

- **search_vuln_coverage** searches the vulnerability match rules runZero ships, by CVE, vendor, or product. This is platform detection coverage, not your findings.
- **search_scan_checks** searches the active scan checks runZero ships, by CVE, check ID, product, or tag.
- **list_rapid_response** returns the account's Rapid Response feed, each entry with the inventory query that finds affected assets.

Organization analytics

These read an organization-wide snapshot rather than running a query.

- **get_overview** returns device type, OS, and attack surface distributions, security-control signal counts, and the total asset count.
- **get_subnets** returns per-subnet rollups (/24 for IPv4, /48 for IPv6) with dominant device types and gateway hints.
- **list_assets_in_subnet** lists the assets whose primary address falls within a CIDR.
- **search_assets** returns compact asset rows matching a case-insensitive substring across address, name, OS, type, and service products.
- **get_outliers** returns assets with notable outlier scores, annotated with the dimensions that make them unusual.
- **get_exposed_assets** returns externally exposed and high-risk internal assets, tagged so the two are not conflated.
- **get_datastores** returns assets exposing databases, file shares, NFS, SMB, and object storage.
- **get_security_controls** returns the histogram of detected EDR, MDM, encryption, firewall, and TLS posture signals.
- **get_critical_systems** returns high-impact assets tagged by role: network core, OT control, identity, hypervisor, datastore, management plane.
- **get_rogue_assets** returns hosts seen by discovery with no identity, EDR, MDM, or configuration-management source.
- **get_eol_assets** returns expired and near-end-of-life assets with their OS version, EOL date, site, and risk.
- **get_eol_crosstab** returns the end-of-life heat-map cells by site or business unit and OS family.
- **get_explorers** lists connected Explorers with their reachable networks, site, and system details.
- **get_recent_tasks** lists recent scans and integration syncs, useful for judging data freshness.
- **get_latest_metrics** returns the latest precomputed metric snapshots.
- **get_goals** lists the security goals the organization is tracking.

Scanning

- **start_scan** starts a scan in an organization, optionally against a specific site or Explorer.

This is the only tool that changes state and the only one MCP does not share with AI threads. Read-only credentials can neither see nor call it.

Retired tools

The tool set converged on the AI thread tools, so several narrower tools were removed. Their replacements:

Retired tool	Use instead
`query_software`	`query_inventory` with entity `software` or `software-group`
`query_vulnerabilities`	`query_inventory` with entity `vulnerability` or `vulnerability-group`
`query_findings`, `critical_findings`, `external_assets_critical_findings`	`query_inventory` with entity `finding`
`query_wireless`	`query_inventory` with entity `wireless`
`query_users`	`query_inventory` with entity `user`
`query_directory_group`	`query_inventory` with entity `group`
`list_sites`	`query_inventory` with entity `site`
`asset_remediation`	`query_inventory` with entity `vulnerability`, which returns remediation text

For bulk, paginated exports of full records, use the export API rather than MCP.

Prompts and resources

Alongside the tools, the server offers:

- The **critical_findings** prompt, which takes an `organization` name or UUID and asks for that organization's critical findings.
- The **docs://querysyntax** resource, which returns the search query syntax reference as Markdown.

Exploring MCP tools, resources, and prompts

MCP Inspector runs locally and lets you add the API key directly in the UI. Once connected, you can copy the server entry to connect other agents, although it may not be the exact

configuration the client needs.

```
$ npx @modelcontextprotocol/inspector@latest
```

1. Enter the API client key as the Authorization value.
2. Connect to the server to view the available tools and resources.

Glossary

The definitions below cover the commonly used terms you will meet throughout the documentation, so you can learn them before you need them.

Terms

Alerts
: Alerts fire when an event matches a rule defined in the Rules Engine.

Analysis reports
: Analysis reports run as tasks rather than being generated on the fly. They are static, so if your inventory changes, assets may no longer be reachable from the report.

Asset
: An asset is a unique network entity from the perspective of the system running the Explorer.

Automatic queries
: Automatic queries are queries you always want to run after a scan. Their results appear in the queries table.

Dashboard
: The dashboard shows trend data and insights that help you see how your inventory changes over time.

Discovery scan
: A discovery scan finds and identifies all the connected devices and assets on your internal network, then builds an inventory of them.

Explorer
: The Explorer is a lightweight scan engine for network and asset discovery.

Insights
: Insights are queries that run automatically after each scan and populate your dashboard.

Inventory
: The inventory lists every asset in the Organization and can be sorted and filtered.

Organization
: An organization is a distinct entity: your business, a department within it, or one of your customers.

Outliers
: Outliers show how often different values occur in specific attributes of assets and services.

Queries
: Queries are filters you apply to your Inventory to find assets of interest.

Rules
: A rule defines the action runZero takes when a set of conditions is met. You can create rules that alert your team to changes in things like Explorers, assets, scans,

organizations, and sites.

Scheduled scans
: Scheduled scans run your scan task at a date and frequency you set.

Self-hosted
: The self-hosted version of runZero lets you run the entire platform on-premises or in your own cloud environment.

Site
: A site is a distinct network segment, usually defined by addressing or accessibility.

Frequently asked questions

These pages solve common runZero issues and answer frequently asked questions (FAQs).

The knowledgebase on the runZero support portal has more solutions and FAQs.

Issues and FAQs

- Why are there so many identical assets in my inventory?
- How do I run runZero without crashing my router?
- How do I scan VMware virtual machines without crashing the host?
- Why do Windows Explorers or CLI scans crash on Intel AMX-capable systems?
- Can EDR or antivirus software interfere with Windows Explorer or CLI scans?
- Why didn't the runZero Explorer capture screenshots?
- What protocols does runZero support?
- What ports does runZero scan?
- Can I safely scan my IoT or OT environments?
- Which browsers are supported when accessing the runZero Console?

For anything not covered here, email runZero support.

Identical assets in inventory

Why are there so many identical assets in my inventory?

Identical assets pile up in your inventory when a network appliance replies to every connection attempt the scan makes. If you can't stop the device from replying to all connections, you have two workarounds:

- Exclude the ports the device responds to from the scan configuration.
- Exclude all or part of the router's IP address range from the scan.

Some enterprise routers and firewalls, like Cisco ASA devices, reply to every unexpected attempt on a particular port with a TCP reset (RST). Some routers also listen for SIP traffic on all addresses and answer it automatically. runZero tries to detect and skip most SIP helper implementations, but can't always do so without risking real results.

runZero generally detects when a router or firewall is replying to every connection attempt and avoids creating assets from those responses. When it misses an appliance that is spoofing responses, a large number of identical assets can appear in your inventory.

If you need help deleting the unwanted records, contact runZero support.

Scanning routers

How do I run runZero without crashing my router?

A router that crashes mid-scan is usually a stateful router that ran out of session slots. Any of these changes can help:

- Deploy additional Explorers so you aren't scanning across routed networks (wired and WiFi, multiple VLANs, and so on).
- Reduce the Max group size in your scan configuration. This caps how many targets runZero scans at once, which corresponds to the number of connections the router sees. The default is 4096.
- Reduce the scan speed. Failed connections then have more time to expire before runZero attempts new ones.
- If the router can run in bridge mode and you don't need its NAT features, bridge mode is likely more reliable. For example, if an ISP-provided router is connected to a WiFi mesh system, you'll likely want to run the mesh system in bridge mode and let the ISP router handle all routing, including NAT. You should then be able to scan across your WiFi network without crashing the base stations.

A stateful router tracks every connection passing through it, and a scan opens thousands of connections in a short burst. The crash usually happens when the router is doing Network Address Translation (NAT) or acting as a stateful security firewall.

Scanning VMWare virtual machines

How do I scan VMware virtual machines without crashing the host?

The best way to scan VMware virtual machines is with a runZero Explorer deployed inside VMware, on a virtual machine connected to the VMnet you want to scan. That Explorer should be able to scan all VMs on the same VMnet without VMware having to track all of the connections.

The Explorer's placement matters because VMnet interfaces normally use Network Address Translation (NAT) to route traffic between the host system and the virtual machines, which makes the VMware software act as a stateful router. A scan that opens thousands of connections can cause the same problems described in Scanning routers.

Scanning on Windows with Intel AMX

Why do Windows Explorers or CLI scans crash on Intel AMX-capable systems?

On systems with Intel Advanced Matrix Extensions (AMX), third-party endpoint protection can crash the runZero Explorer process mid-scan, on both physical hardware and virtual machines. The endpoint protection injects code without sizing its stack correctly; when that code raises an exception, the AMX exception frame it writes corrupts the runZero thread.

Two workarounds are available:

1. Disable AMX.

- If the Explorer runs on Windows as a Hyper-V guest, enable **Processor Compatibility** for that VM, and restart.
- For other types of virtual machines or physical systems, disable AMX in the BIOS or hypervisor CPU settings.

2. Disable or replace the endpoint protection software.

- For Microsoft Defender, excluding the runZero process is not enough: Defender still injects the buggy code into the runZero Explorer.
- The system may need a full restart to return the service to a non-hooked state.

If the crash persists after either workaround, use the “Download Logs” option from the Explorer details page and contact runZero support. The `.err` file in this ZIP contains the crash logs, the current state of AMX, and the list of loaded DLLs.

Scanning on Windows with EDR or antivirus

Can EDR or antivirus software interfere with Windows Explorer or CLI scans?

Yes. Some EDR, antivirus, HIDS, and endpoint protection products on Windows can interfere with runZero Explorer and CLI scanner operations. Common symptoms are blocked packet capture, failed raw socket operations, unstable scan behavior, and process termination.

To reduce or resolve these issues:

1. Add allowlist exceptions for the runZero binaries, services, and directories the Explorer or CLI scanner uses.
2. Check that endpoint security policies aren’t blocking packet capture components (including npcap/winpcap where applicable).
3. If your endpoint tooling supports policy modes, use a monitoring-only mode while validating scan operations.
4. In some environments, you may need to temporarily disable endpoint protection controls on the host to confirm whether they are the root cause.
5. Re-run the scan, or reinstall or restart the Explorer, after policy updates.

If issues continue, collect Explorer diagnostics or CLI output and contact runZero support.

Explorer not capturing screenshots

Why didn’t the runZero Explorer capture screenshots?

The runZero Explorer needs a working install of Google Chrome to capture screenshots. Install Chrome in one of the locations the Explorer checks, listed below for each OS.

Windows

On Windows, the runZero Explorer looks for Chrome in:

```
c:\Program Files (x86)\Google\Chrome\Application\chrome.exe
```

The Explorer also checks these environment variables:

- `ProgramFiles(x86)`
- `ProgramFiles`
- `ProgramW6432`

Each may point to another directory, in which case the Explorer also looks in `\Google\Chrome\Application\chrome.exe` under each of those directories.

To see what the environment variables are set to, open a Windows command prompt and enter the command `set`.

On a default Windows 10 install, `ProgramFiles` and `ProgramW6432` are both `c:\Program Files`, so the Explorer also checks:

```
c:\Program Files\Google\Chrome\Application\chrome.exe
```

This is the default location for Chrome on Windows 10 (64 bit).

macOS

On macOS, the Explorer checks these locations:

- `/Applications/Google Chrome.app/Contents/MacOS/Google Chrome`
- `/Applications/Google Chrome Canary.app/Contents/MacOS/Google Chrome Canary`
- `/Applications/Chromium.app/Contents/MacOS/Chromium`

Linux

On Linux, the Explorer checks these locations:

- `/usr/bin/google-chrome`
- `/usr/bin/google-chrome-beta`
- `/usr/bin/google-chrome-unstable`

- `/usr/local/bin/chrome`
- `/usr/bin/chrome`
- `/opt/google/chrome/google-chrome`
- `/usr/bin/chromium`
- `/usr/bin/chromium-browser`

Chrome is installed, but screenshots still don't work

Check the Explorer log file for details about the scan. If Chrome is installed in one of the standard locations but the Explorer still can't find it, it could be a permissions issue. Chrome can also fail to run for other reasons, such as a corrupt Chrome profile.

You also won't get a screenshot of an http(s) service when:

- The service is on an IPv6 link-local IP address (Chrome cannot access these)
- It uses a TLS version lower than 1.2 (Chrome no longer supports accessing these)
- The browser timed out while taking the screenshot (for example, an unresponsive host)
- The host returned an error code

Explorer security model

runZero Explorers run network scans, sample traffic for passive discovery, and execute integrations inside the environment where you deploy them. This page describes how runZero secures them, from development through uninstallation.

Development

runZero Explorers are written in Go, which gives them cross-platform compatibility and memory safety by default. The runZero engineering team updates the Explorer code and its dependencies continuously so that dependency-level vulnerabilities are resolved quickly. Every engineer must use MFA to access our Git repository. Each code change is reviewed manually before it merges to our development branch and deploys to a test environment, then reviewed again before it merges to our staging branch and deploys to a staging environment. During the build, runZero signs each Explorer binary with a private Ed25519 key and stamps the signature into the end of the binary. Windows binaries also get an Authenticode signature from an extended-validation certificate stored in a cloud-based HSM. Once all automated tests pass and any behavior changes are confirmed, the staging binaries are tagged for release and moved to the production deployment location in AWS S3.

Deployment

You provision an Explorer by hand the first time: download it directly from the runZero Console (cloud or self-hosted). Installing an Explorer covers the steps. The download stamps the link token and organization ID into the binary while preserving the code signatures. On the deployed system, the binary connects back to its stamped Console URL, validates TLS, and registers. Registration starts with the Explorer generating a unique host ID from the operating system's random source. The Explorer stores the ID with limited permissions in its file system

(non-Windows) or the registry (Windows) and uses it to confirm its identity on future connections.

On a new Explorer's first connection, the Console records the Explorer-generated host ID and creates a database record for that instance. No other Explorer, even one on the same system, can claim that record without possessing and verifying the host ID. By default, a new Explorer receives no data from the Console and no tasks (including tasks with credentials) until a user selects it manually. As a result, registered but untrusted Explorers are relatively safe, as long as you don't use them for tasks that include sensitive data (typically integrations with credentials).

Once an Explorer has the Organization ID, it no longer needs the link token (Download Token) to connect and register. runZero plans to require the current Download Token in the near future. It is not a requirement yet, because unauthorized Explorer registrations expose few if any security risks. Once registration requires the current Download Token, older Explorers won't be able to reconnect after the Download Token is rotated. Active Explorers should always receive the latest Download Token as part of the upgrade request.

Commands

The runZero Console sends Explorers new tasks and a few specific requests. Tasks are limited to active scans, passive traffic sampling, and integrations. Integrations can run custom code written in the Starlark language; that code has access to the network but no access to the Explorer host machine or its file system. Outside of tasks, Explorers accept requests to uninstall, update, and post diagnostics (limited to stack traces and Explorer attributes like operating system and hardware specifications). Explorers implement no commands for network tunneling or host access to the Explorer's machine.

The Console sends credentials to an Explorer only for an integration task, and only when the task scope matches the CIDR or other configuration for those credentials. It never sends credentials to runZero-hosted External Explorers, regardless of the task configuration. Credential parameters travel in the same TLS connection as the request and are also encrypted to a unique per-process AGE encryption key. Even if runZero traffic is logged through a TLS proxy with a pinned CA, the credentials cannot be recovered without also extracting the private, in-process-only, temporary AGE key.

Updates

Before scheduling a task (scan, passive, or integration), the runZero Console checks that the Explorer is running the latest version. An out-of-date Explorer is updated first, and the task is scheduled once the update completes. If the update takes longer than an internal threshold, or fails for a reason like a read-only filesystem on the Explorer, the task is scheduled anyway once that threshold passes.

The update request carries a URL for the new binary. The Explorer connects to that URL, verifies the TLS parameters, and downloads the binary to a temporary directory. It then checks that the binary carries a valid embedded code signature and rejects the update if the check fails. If the signature is valid, the Explorer launches the new binary in update mode, which typically replaces the running service with the new executable.

An update can also change the Explorer service's Organization or even its Console; this is how Explorers move between installations and organizations. In either case, the TLS connection must be valid and the binary must carry a valid code signature. The same verification rejects version downgrades by comparing the signed version metadata in the binary against the running version.

Uninstallation

Deleting a runZero Organization automatically uninstalls its Explorers. You can also uninstall an Explorer from the runZero Console or with the "uninstall" CLI option of the Explorer binary. During uninstallation the Explorer removes as many artifacts as it can. You can remove any leftovers by hand (common on Windows systems, where the OS locks the program files and directory while the Explorer is running).

Defense in depth

runZero layers multiple defenses into its architecture to protect customer data:

- An attacker who can push code to our internal repository still needs to pass review before it can merge into our protected staging branch.
- An attacker who compromises our binary distribution point (AWS S3) still needs a valid signing key to affect any customer update. The Explorer also compares authenticated, signed version metadata, which prevents downgrade attacks.
- An attacker who obtains a customer's download link can register new Explorers but gets no data from the Console unless a user specifically chooses those Explorers for a task.
- An attacker who can send commands to an Explorer (by compromising a Console) is limited to a few relatively safe commands and cannot force downgrades or arbitrary code execution on the Explorer host.
- An attacker who can MiTM the TLS communication with the Console cannot decrypt the credentials used in integration tasks.
- An attacker who can schedule a custom integration task with a malicious Starlark script can reach the network from the Explorer's perspective but cannot access the Explorer host.

In every one of these situations, runZero also runs extensive logging and alerting, so we can investigate any malicious action and assess the impact of any issue.

Browsers supported by the runZero Console

Which browsers are supported when accessing the runZero Console?

The runZero Console supports these browser versions or newer:

- *Chrome 123* (released March 18, 2024)
- *Edge 123* (released March 22, 2024)
- *Safari 17.5* (released May 12, 2024)
- *Firefox 120* (released November 20, 2023)
- *Opera 109* (released March 26, 2024)

Internet Explorer is not supported.

Mobile browser support is experimental and not guaranteed to work.

Common sign-in issues

User account has been locked from too many failed sign-in attempts.

The account uses password authentication, and runZero locked it after repeated incorrect sign-in attempts. Have your runZero administrator follow these steps:

1. In the runZero console, visit the Team page.
2. Select the locked account.
3. Use the “Unlock user accounts” option in the “Reset” dropdown menu.

If you can’t reach an administrator, contact support@runzero.com.

Certificate information for single sign-on has expired and must be updated in order to sign in

Replace the expired SAML certificate configured for your runZero tenant:

1. Get a new, valid certificate from your identity provider (for example, Okta or Google Workspace).
2. Sign in to runZero with a superuser account that uses link-based or password-based authentication.
3. Paste the new certificate contents into the “Certificate” field of the Identity provider settings page.

- To reach the “Certificate” field, open the Team page, then click the “SSO settings” button in the page header.

If you are the administrator for your account and can’t sign in at all, contact support@runzero.com and include the new certificate.

Invalid SAML response

Review the SAML configuration in your identity provider (for example, Okta or Google Workspace); the data it sent to the runZero console was in an invalid format. Implementing SSO covers runZero's SAML SSO setup.

The information provided by your identity provider was incomplete or not valid.

Check that your identity provider is configured to send a valid Subject and NameID to the runZero console. Its response was missing one or more of the required assertion elements, `Subject` and `NameID`.

The email address `{email address}` is already in-use by an existing user account.

The email address your SSO identity provider sent already belongs to an account in a different runZero tenant. Change that account's email address or delete the account, in any of these ways:

- Sign in to the runZero console with the email address from the error message and change the account's email address to something different.
- Ask the administrators of the tenant where your email address is registered to delete your account from the Team page.
- Email runZero support at support@runzero.com and ask them to delete your existing account.

Once the email address is free, you can sign in with SSO.

The identity provider has provided an invalid NameID

Configure your identity provider (for example, Okta or Google Workspace) to send an email address in the NameID attribute, or ask your administrators to do so. The NameID it sent is not formatted like an email address, and the runZero console uses this attribute as your email address.

runZero data formats

runZero consumes and produces a handful of data formats. This page shows an example of each and describes its fields and use cases.

Formats

- Scan data (sample)
- Asset data (sample)
- Change reports (sample)

Scan data

Scan data is the raw output of the runZero Explorer and the runZero CLI: newline-delimited JSON (JSONL) holding the unprocessed output of the scan engine. Downloading the task data for an Explorer-run scan returns this format, and it is the same format as the `scan.runzero.gz` file the CLI creates. runZero builds the Inventory view by processing scan data in chronological order to produce the current state at a given point in time.

You can import scan data into an existing site through the Inventory Import menu of the web console or the `--import` parameter of the CLI. Each line of the file is a JSON object with a `type` and a 64-bit Unix timestamp.

This example is the raw scan data for a single Apple Mac Mini:


```
{"type":"config","ts":1597259738842951567,"probes":["arp","bacnet","dns","dtls","echo","ike","ipmi","m
dns","memcache","mssql","natpmp","netbios","ntp","openvpn","pca","rdns","rpcbind","sip","snmp","ssd
p","syn","tftp","ubnt","wlan-list","wsd"],"addresses":["192.168.0.1","192.168.30.1","192.168.40.1"],"n
etworks":["192.168.0.1/24","192.168.30.1/24","192.168.40.1/24"],"params":{"arp-fast":"false","bacnet-p
ort":"47808","clock-offset":"0","dns-port":"53","dns-resolve-name":"www.google.com","dns-trace-domai
n":"helper.rumble.network","dtls-ports":"443,3391,4433,5246,5349,5684","excludes":"","ike-port":"50
0","ipmi-port":"623","max-group-size":"4096","max-host-rate":"40","max-sockets":"512","mdns-port":"535
3","memcache-port":"11211","mssql-port":"1434","nameservers":"","natpmp-port":"5351","netbios-port":"1
37","nopcap":"false","ntp-port":"123","openvpn-ports":"1194","passes":"1","pca-port":"5632","probe
s":"arp,bacnet,dns,dtls,echo,ike,ipmi,mdns,memcache,mssql,natpmp,netbios,ntp,openvpn,pca,rdns,rpcbind,
sip,snmp,ssdp,syn,connect,tftp,ubnt,wlan-list,wsd","rate":"1000","rdns-max-concurrent":"64","rpcbind-p
ort":"111","rpcbind-port-nfs":"2049","screenshots":"true","sip-port":"5060","skip-broadcast":"true","s
nmp-comms":"public,private","snmp-poll-interval":"300","snmp-port":"161","snmp-timeout":"5","snmp-v3-a
uth-passphrase":"","snmp-v3-auth-protocol":"none","snmp-v3-context":"","snmp-v3-privacy-passphras
e":"","snmp-v3-privacy-protocol":"none","snmp-v3-username":"","ssdp-port":"1900","syn-max-retrie
s":"2","syn-udp-trace-port":"65535","tcp-ports":"1300,5554,8020,20034,47001,41080,2601,2604,2638,5060,
7181,10202,4679,2181,34205,13,2323,5601,18881,50070,139,1129,2199,2375,4444,902,1440,2103,32913,1311,9
524,8028,8883,13364,37718,512,3200,5683,10203,81,1091,5222,8081,13838,37777,1,5672,8095,65535,21,540,5
48,1102,27080,28017,34443,40007,6060,6542,8300,27888,4786,9443,2049,3050,5984,46823,12221,1352,6405,26
122,7210,41025,1103,1530,1883,8834,443,9100,45230,1234,3128,5432,12397,111,993,3780,5250,6112,524,524
7,20031,1211,1755,5985,6070,8880,1241,3690,6002,1035,4000,8080,9081,2362,23,587,921,8903,31001,143,259
8,3273,6101,8812,10628,25,113,513,1720,2533,6905,32764,38080,5040,20010,6001,6660,8471,82,2222,5093,62
62,6379,8545,384,5168,20222,7579,998,3057,3217,6106,9391,9,2380,5520,9060,19300,30718,49,84,161,5900,1
0001,8009,19,617,2100,5580,38292,85,6667,10443,42,2121,5986,23791,515,1199,10008,16993,631,2083,8443,9
527,13500,27017,30000,41523,554,5061,4659,8333,9855,5355,7001,1000,1220,5521,11234,20000,6988,3351,754
7,7,1900,7778,9160,31099,1030,10616,7902,8090,12174,1533,135,5631,623,5038,9300,19888,14330,109,1433,1
5672,1581,3790,5632,9999,80,2381,4840,7800,61616,1101,1128,1494,3311,9092,11000,110,995,1098,5800,523,
8087,10098,28784,407,7777,9090,19810,34963,50000,502,1100,8161,8180,9152,11099,2379,8023,88,1582,2010
1,16102,16992,1583,5814,5938,20111,11211,636,27000,1158,5400,5920,7443,9530,20171,8800,9099,7474,8222,
```

71,8303,62078,705,873,6000,7077,8503,9495,34964,5666,17200,5433,7801,11333,12401,12203,25025,264,2525,3628,9809,26000,50090,9000,2967,137,4730,5051,8899,10050,52302,8400,53,389,402,4443,7700,62514,1090,5353,6082,6661,40317,8089,17185,912,5405,28222,465,4445,6503,8014,57772,23472,1080,7021,8088,22222,5000,9084,18264,8888,6050,7144,41524,69,1811,44334,102,6502,1521,2809,9471,888,5351,5498,123,34962,8205,9042,2947,3389,5555,10080,10162,9200,500,1089,7510,15200,2207,3500,8008,9418,3460,6504,7770,25000,55553,179,783,8012,9595,46824,7580,5560,49152,83,903,1604,3632,4322,4567,445,8030,9390,3817,8901,10051,9002,27019,910,1099,3000,3299,222,7787,37,2000,3306,48899,7879,79,1723,3037,3312,8000,23943","tftp-ports":"69","ubnt-port":"10001","verbose":"true","wlan-list-poll-interval":"300","wsd-port":"3702"},"scan_targets":{"networks":["192.168.0.5/32"],"enable_dns":true,"enable_ip6":false,"inputs":["192.168.0.5"],"dns_timeout":2000000000,"concurrency":24},"version":"1.10.0 (build 20200804052508) [eae4e551f9f0ce5ab3bf0a1410b2ed5098db097e]"}
{"type":"status","ts":1597259758851278069,"level":"info","source":"connect","msg":"waiting on TCP probes to complete"}
{"type":"stats","ts":1597259769858733899,"stats":{"elapsed":31,"progress":94,"rateLimitTime":24246445895,"recv":546,"recvBytes":37404,"recvError":10,"recvRate":17,"resultCount":22,"secondsLeft":1,"sent":503,"sentBytes":36833,"sentError":0,"sentRate":16,"startTime":1597259738843056937}}
{"type":"result","ts":1597259738844372618,"host":"192.168.0.5","port":"0","proto":"icmp","probe":"echo","info":{"icmp.addrs":"192.168.0.5","icmp.rtts":"541214","ip.flags":"DF","ip.id":"0","ip.tos":"0","ip.ttl":"64"}}
{"type":"result","ts":1597259739077542717,"host":"192.168.0.5","port":"137","name":"MACMINI-EE7C7B","proto":"udp","probe":"netbios","info":{"netbios.domain":"WORKGROUP","netbios.mac":"f0:18:98:ee:7c:7b","netbios.macDateAdded":"2017-12-23","netbios.macVendor":"Apple, Inc."}}
{"type":"result","ts":1597259739219939211,"host":"192.168.0.5","port":"137","proto":"udp","probe":"netbios","info":{"netbios.addrs":"192.168.0.5"}}
{"type":"result","ts":1597259739471688048,"host":"192.168.0.5","port":"0","proto":"arp","probe":"arp","info":{"arp.mac":"f0:18:98:ee:7c:7b","arp.macDateAdded":"2017-12-23","arp.macVendor":"Apple, Inc.","source":"arp"}}
{"type":"result","ts":1597259739826114224,"host":"192.168.0.5","port":"5353","name":"Developers-Mac-mini","proto":"udp","probe":"mdns","info":{"mdns.replies":"5.0.168.192.in-addr.arpa.=PTR,Developers-Mac-mini.local."}}
{"type":"result","ts":1597259740197709484,"host":"192.168.0.5","port":"445","proto":"tcp","probe":"connect","info":{"ntlmssp.dnsComputer":"Developers-Mac-mini.local","ntlmssp.dnsDomain":"local","ntlmssp.negotiationFlags":"0x62898235","ntlmssp.netbiosComputer":"DEVELOPERS-MAC-MINI","ntlmssp.netbiosDomain":"MACMINI-EE7C7B","ntlmssp.ntlmRevision":"15","ntlmssp.targetName":"MACMINI-EE7C7B","ntlmssp.timestamp":"0x01d670dd06500880","ntlmssp.version":"6.1.7600","protocol":"smb1\tsmb2\tsmb3","smb.capabilities":"0x00000066","smb.dialect":"0x0302","smb.guid":"ff12583f-5ba1-53b8-8ff3-a48a394056f7","smb.nativeLM":"@(#)PROGRAM:smbd PROJECT:smbx-499.60.1","smb.nativeOS":"Darwin","smb.sessionID":"0x9551b7cb00000001","smb.signing":"required","source":"mdns"}}

The data contains four types of object:

- **Scan config:** The `{"type":"config"}` object holds the full set of scan parameters, the scan engine version, and on Windows, the installed npcap version. The analysis engine reads the scan targets from this record to decide whether a given IP address was in scope.
- **Scan status:** The `{"type":"status"}` object carries diagnostic output from the scan engine, which can point to problems that occurred while the scan was running.
- **Scan stats:** The `{"type":"stats"}` object holds point-in-time statistics for the scan, including the number of packets sent and received and the progress estimate.
- **Scan result:** The `{"type":"result"}` object is a target's response to a specific probe: a TCP SYN+ACK reply, an ICMP reply, or the result of an application-layer probe such as an

SNMP query response or an HTTP screenshot. runZero analyzes and correlates scan results to build the asset data format.

The scan stats object has these sub-fields:

Field	Description
cpu	CPU Core Percent * 100. 100% of one core would be 100000.
elapsed	The number of seconds since the scan started.
fdcount	The number of open file descriptors.
memory	The current memory usage in bytes.
progress	The estimated progress as a percentage (90 = 90%).
rateLimitTime	The number of Unix nanoseconds spent idling in the rate limiter.
recv	The number of packets received from the network.
recvBytes	The number of bytes received from the network.
recvError	The number of errors receiving from the network.
recvRate	The average packet receive rate for the scan.
resultCount	The total number of findings from the scan.
routines	The number of internal goroutines in the scan engine.
secondsLeft	The estimated seconds left to complete the scan.
sent	The number of packets sent to the network.
sentBytes	The number of bytes sent to the network.
sentError	The number of errors sending to the network.
sentRate	The average packet send rate for the scan.
startTime	The Unix timestamp in nanoseconds of when the scan started.

The scan result object has these fields in addition to `type` and `ts`:

Field	Description
host	The IP address associated with the response.
name	An optional hostname returned as part of this response.
port	The TCP or UDP port or zero for other protocols.
proto	The transport protocol, one of `arp`, `icmp`, `tcp`, or `udp`.
probe	The specific internal probe name that returned this response.
info	The result details object where all keys and values are strings.

The `info` object holds probe-specific response data. Key names are usually in the form `probe.subfield`, with a few exceptions, and values are always strings, even for numeric and array content. Multiple values for a key are a tab-delimited array. Empty values are never reported for info keys. A scan may return several result objects for one probe, sometimes with duplicate values; the next processing phase correlates, deduplicates, and merges them.

Asset data

Asset data is the set of correlated and fingerprinted assets shown in the web console Inventory view and written to the `assets.jsonl` file by the runZero CLI. It represents the state of each unique asset at a point in time, built up by processing one or more sets of scan data.

Asset data comes in a few variants: line-delimited JSON (JSONL), standard JSON documents, and a simplified CSV export. JSONL is the easiest to work with because you can process it incrementally without loading the entire response into memory. Exporting asset data covers the export options in the inventory view.

This example is the correlated asset data from a scan of a single Apple Mac Mini:


```
{"id":"b73f8e09-78a6-4d2b-979d-e63908f28251","created_at":1597259778,"updated_at":1597259778,"organiza
tion_id":"b7fb13a7-701d-4ca5-b0e6-6f28f06cc866","site_id":"52d60c51-8dee-4f09-94e5-2dee30050a25","aliv
e":true,"last_seen":1597259750,"first_seen":1597259738,"detected_by":"arp","type":"Desktop","os":"Appl
e macOS","os_version":"10.15","hw":"Apple Mac Mini (Late 2018)","addresses":["192.168.0.5"],"addresses
_extra":["fe80::1c9d:c567:8db1:d79b"],"macs":["f0:18:98:ee:7c:7b"],"mac_vendors":["Apple, Inc."],"name
s":["MACMINI-EE7C7B","DEVELOPERS-MAC-MINI","DEVELOPERS-MAC-MINI.LOCAL"],"tags":{},"domains":[],"servic
es":{"192.168.0.5/0/arp/":{"arp.mac":"f0:18:98:ee:7c:7b","arp.macDateAdded":"2017-12-23","arp.macVendo
r":"Apple, Inc.","source":"arp","ts":"1597259739"},"192.168.0.5/0/icmp/":{"icmp.addrs":"192.168.0.
5","icmp.rtts":"541214","ip.flags":"DF","ip.id":"0","ip.tos":"0","ip.ttl":"64","ts":"1597259738"},"19
2.168.0.5/137/udp/":{"netbios.addrs":"192.168.0.5","netbios.domain":"WORKGROUP","netbios.mac":"f0:18:9
8:ee:7c:7b","netbios.macDateAdded":"2017-12-23","netbios.macVendor":"Apple, Inc.","protocol":"netbio
s","ts":"1597259739"},"192.168.0.5/22/tcp/":{"banner":"SSH-2.0-OpenSSH_7.9","ip.flags":"DF","ip.i
d":"0","ip.tos":"0","ip.ttl":"64","protocol":"ssh","service.cpe23":"cpe:/a:openbsd:openssh:7.9","servi
ce.family":"OpenSSH","service.vendor":"OpenBSD","service.version":"7.9","source":"mdns","ssh.hostKey.d
ata":"AAAAB3NzaC1yc2EAAAADAQABAAABAQCjGYTFcSp2Fs/R8dboLYiQ6PPrulZYanYH3SCYYr5QgC1SIF3AURGYTMnUDAS+tTI/
PquwowkgiqJrtfsQMAsCrahbPahwiOLTupsuLNp3evXYYSf8ZQFyBN8iz5cys06u+yczqWG7Fu8mgpS8zwCwN7yRrbFWd8+Hp6GgfU
U4Z6jUQoZu7iajpbSXlTA9OYKXQIZOm8qc4mPLT/uHw9nxNmExWA1V/2ZeoS59NGSV8zFMKb52SOXKhkvHAIUVh5NJDAudxK4uP4eG
6dxr8btYtVKIOYKlsLdSBSfHvSCvVVlb7DKJBiMXG+qspt33Zd73o4S9ICh2OaSbVt7h/NZ3","ssh.hostKey.md5":"75:9b:a2:
e6:10:da:72:8a:11:91:3f:a1:43:14:7f:2e","ssh.hostKey.sha256":"SHA256:xVJfddKBJ9E5jstVCj0zY8763Rnxy2pqp
zaLZXO+cHc","ssh.hostKey.type":"ssh-rsa","syn.rtt":"542019","tcp.options":"MSS:05b4","tcp.ts":"2009546
838","tcp.urg":"0","tcp.win":"65535","ts":"1597259740"},"192.168.0.5/3031/tcp/":{"source":"mdns","t
s":"1597259740"},"192.168.0.5/3283/tcp/":{"source":"mdns","ts":"1597259740"},"192.168.0.5/445/tcp/":
{"ip.flags":"DF","ip.id":"0","ip.tos":"0","ip.ttl":"64","ntlmssp.dnsComputer":"Developers-Mac-mini.loc
al","ntlmssp.dnsDomain":"local","ntlmssp.negotiationFlags":"0x62898235","ntlmssp.netbiosComputer":"DEV
ELOPERS-MAC-MINI","ntlmssp.netbiosDomain":"MACMINI-EE7C7B","ntlmssp.ntlmRevision":"15","ntlmssp.target
Name":"MACMINI-EE7C7B","ntlmssp.timestamp":"0x01d670dd06500880","ntlmssp.version":"6.1.7600","protoco
l":"smb1\tsmb2\tsmb3","smb.capabilities":"0x00000066","smb.dialect":"0x0302","smb.guid":"ff12583f-5ba1
-53b8-8ff3-a48a394056f7","smb.nativeLM":"@(#)PROGRAM:smbd PROJECT:smbx-499.60.1","smb.nativeOS":"Darwi
n","smb.sessionID":"0x9551b7cb00000001","smb.signing":"required","source":"mdns","syn.rtt":"624414","t
cp.options":"MSS:05b4","tcp.ts":"2009544186","tcp.urg":"0","tcp.win":"65535","ts":"1597259740"},"192.1
68.0.5/5353/udp/":{"hw.device":"Desktop","hw.family":"Mac mini","hw.product":"Mac mini (Late 2018)","h
w.vendor":"Apple","mdns.addrs":"fe80::1c9d:c567:8db1:d79b\t192.168.0.5","mdns.device.model":"Macmini8,
```

```
1","mdns.device.osxvers":"19","mdns.ports":"eppc/tcp=3031\tnet-assistant/udp=3283\trfb/tcp=5900\tsftp-
ssh/tcp=22\tsmb/tcp=445\tssh/tcp=22","mdns.replies":"5.0.168.192.in-addr.arpa.=PTR,Developers-Mac-min
i.local.\tDeveloper\\226\\128\\153s\\ Mac\\ mini._device-info._tcp.local.=TXT,model=Macmini8,1 osxvers
=19\tDevelopers-Mac-mini.local.=A,192.168.0.5\tDevelopers-Mac-mini.local.=AAAA,fe80::1c9d:c567:8db1:d7
9b\t_eppc._tcp.local.=PTR,Developer\\226\\128\\153s\\ Mac\\ mini._eppc._tcp.local.\t_net-assistant._ud
p.local.=PTR,Developer\\226\\128\\153s\\ Mac\\ mini._net-assistant._udp.local.\t_rfb._tcp.local.=PTR,D
eveloper\\226\\128\\153s\\ Mac\\ mini._rfb._tcp.local.\t_sftp-ssh._tcp.local.=PTR,Developer\\226\\128
\\153s\\ Mac\\ mini._sftp-ssh._tcp.local.\t_smb._tcp.local.=PTR,Developer\\226\\128\\153s\\ Mac\\ min
i._smb._tcp.local.\t_ssh._tcp.local.=PTR,Developer\\226\\128\\153s\\ Mac\\ mini._ssh._tcp.local.","mdn
s.services":"ssh/tcp\tsftp-ssh/tcp\teppc/tcp\trfb/tcp\tsmb/tcp\tnet-assistant/udp","os.cpe23":"cpe:/o:
apple:mac_os_x:10.15","os.family":"Mac OS X","os.product":"Mac OS X","os.vendor":"Apple","os.versio
n":"10.15","protocol":"mdns","ts":"1597259740"},"192.168.0.5/5900/tcp/":{"ip.flags":"DF","ip.i
d":"0","ip.tos":"0","ip.ttl":"64","protocol":"vnc","source":"mdns","syn.rtt":"625419","tcp.options":"M
SS:05b4","tcp.ts":"2009549134","tcp.urg":"0","tcp.win":"65535","ts":"1597259749","vnc.version":"RFB 00
3.889"}},"credentials":{},"rtts":{"icmp/echo":[541214]},"attributes":{"_macs.ipmap":"f0:18:98:ee:7c:7b
=192.168.0.5","ip.ttl.hops":"0","ip.ttl.host":"192.168.0.5","ip.ttl.port":"22","ip.ttl.source":"64","i
p.ttl.source.icmp":"64","ip.ttl.win":"65535","match.db":"mdns-device-info-txt","match.score":"90","ntl
mssp.dnsComputer":"Developers-Mac-mini.local","ntlmssp.dnsDomain":"local","ntlmssp.version":"6.1.760
0","os.cpe23":"cpe:/o:apple:mac_os_x:10.15","os.family":"Mac OS X","os.product":"Mac OS X","os.vendo
r":"Apple","os.version":"10.15","smb.guid":"ff12583f-5ba1-53b8-8ff3-a48a394056f7","smb.nativeLM":"@(#)
PROGRAM:smbd PROJECT:smbx-499.60.1","smb.nativeOS":"Darwin"},"service_count":9,"service_count_tcp":
5,"service_count_udp":2,"service_count_arp":1,"service_count_icmp":1,"lowest_ttl":0,"lowest_rtt":54121
4,"last_agent_id":"ca811190-329c-4da3-8cbe-3fd2ddff2663","last_task_id":"de5a4176-3614-4b71-8939-95b91
08124aa","newest_mac":"f0:18:98:ee:7c:7b","newest_mac_vendor":"Apple, Inc.","newest_mac_age":151398720
0000000000,"comments":null,"service_ports_tcp":["22","445","3031","3283","5900"],"service_ports_udp":
["137","5353"],"service_protocols":["mdns","netbios","smb1","smb2","smb3","ssh","vnc"],"service_produc
ts":["openbsd openssh"],"org_name":"Test Lab","site_name":"MAC","agent_name":"TENTACULAR"}
```


Top-level asset data fields use several data types: string arrays, objects, strings, and integers. runZero tracks multiple IP addresses and MACs per asset, so those are arrays. Asset-level attributes and services are objects with additional structure. The `id` field (a V4 UUID) uniquely identifies an asset; nearly every other field can change between scans as assets move around the network, change IPs, and open and close services.

Every asset belongs to an organization and a site within that organization.

The core asset data fields are:

Field	Description
id	The unique ID of this asset defined as a v4 UUID.
created_at	The asset created time represented as a 64-bit Unix timestamp in seconds.
updated_at	The asset last update time represented as a 64-bit Unix timestamp in seconds.
organization_id	The organization identifier defined as a v4 UUID.
site_id	The site identifier defined as a v4 UUID.
alive	A boolean indicating whether this asset was found during the last scan of the site.

last_seen	The time the asset last responded represented as a 64-bit Unix timestamp in seconds.
first_seen	The time the asset first responded represented as a 64-bit Unix timestamp in seconds.
detected_by	The protocol used to first detect that this asset was alive during the last scan.
type	A classification that represents a guess of the asset's purpose.
os_vendor	The operating system vendor name as determined by the fingerprinting engine.
os_product	The operating system product name as determined by the fingerprinting engine.
os_version	The operating system version as determined by the fingerprinting engine.
os	The operating system name as determined by the fingerprinting engine.
hw_vendor	The hardware vendor name as determined by the fingerprinting engine.
hw_product	The hardware product name as determined by the fingerprinting engine.
hw_version	The hardware version as determined by the fingerprinting engine.
hw	The hardware definition as determined by the fingerprinting engine.
addresses	An array of IP (v4/v6) addresses for the asset that were within the scan scope.
addresses_extra	An array of IP (v4/v6) addresses for the asset that were outside the scan scope.
macs	An array of MAC addresses associated with this asset.
mac_vendors	An array of MAC address vendors associated with this asset.
names	An array of unique hostnames associated with this asset (uppercase).
domains	An array of unique domain names associated with this asset (uppercase).
tags	A text representation of the user-specified tags associated with this asset.
attributes	An object containing a map of key-value string attributes for this asset.
services	An object containing each associated service with the key representing the service description.
credentials	An object containing a map of any associated credentials (SNMP v2/v3).
rtts	An object containing a map of round-trip measurement times in milliseconds.

service_count	A count of TCP, UDP, ARP, and ICMP services.
service_count_tcp	A count of TCP services.
service_count_udp	A count of UDP services.
service_count_arp	A count of ARP services (0 or 1).
service_count_icmp	A count of ICMP services.
software_count	A count of software results.
vulnerability_count	A count of vulnerability results.
lowest_ttl	The lowest observed source TTL for this asset.
lowest_rtt	The lowest observed source RTT for this asset.
last_agent_id	The v4 UUID of the Explorer responsible for the last scan of this asset.
last_task_id	The v4 UUID of the task associated with the last scan of this asset.
last_task_id	The v4 UUID of the task associated with the last scan of this asset.
newest_mac	The “newest” MAC address by registration date.
newest_mac_vendor	The “newest” MAC address vendor by registration date.
newest_mac_age	The “newest” MAC address registration date as a Unix timestamp in nanoseconds.
comments	User-specified comments associated with this asset.
service_ports_tcp	An array of strings representing the unique TCP ports found on this asset.
service_ports_udp	An array of strings representing the unique UDP ports found on this asset.
service_protocols	An array of strings representing the unique protocols found on this asset.
service_products	An array of strings representing the unique products found on this asset.
scanned	A TRUE or FALSE value indicating whether runZero has scanned the asset.
source_ids	The ID of the data source, mapped to this table.
eol_os	The operating system End-of-Life time represented as a 64-bit Unix timestamp in seconds.
eol_os_ext	The operating system extended End-of-Life time represented as a 64-bit Unix timestamp in seconds.
outlier_score	The 0-5 score range indicating how unusual an asset is compared to the rest of the inventory.

outlier_raw	The heuristic score indicating how unusual an asset is compared to the rest of the inventory.
sources	The name of the data source, mapped to this table.
org_name	The name of the organization associated with this asset.
site_name	The name of the site associated with this asset.
agent_name	The name of the Explorer associated with this asset.
agent_external_ip	The external IP address of the Explorer associated with this asset.
hosted_zone_name	The name of the hosted zone associated with this asset.
subnets	The registered subnets associated with the site this asset is in.

The `services` field is keyed by unique service identifier, with values stored as strings. Multiple values may be stored as tab-delimited strings in the service values. A typical service key looks like `192.168.0.5/22/tcp/`. The key is made of `address`, `port`, `transport`, and `virtual host` (which can be blank).

Change reports

The runZero platform calculates a **change report** after processing each scan. It is a JSON document you can download from the Task Details page, with this structure:

```
{
  "assets":{
           "new":{ "<asset-UUID>": { "Asset Data Fields":"" } },
           "online":{ "<asset-UUID>": { "Asset Data Fields":"" } },
           "offline":{ "<asset-UUID>": { "Asset Data Fields":"" } },
           "changed":{ "<asset-UUID>": { "Asset Data Fields":"" } },
           "summary":{
             "changed":#,
             "new":#,
             "total":#,
             "unchanged":#
           } },
  "directory_users":{
           "new":{ "<user-UUID>": { "User Data Fields":"" }},
           "changed":{ "<user-UUID>": { "User Data Fields":"" }},
           "summary":{
             "changed":#,
             "new":#,
             "total":#,
             "unchanged":#
           } },
  "directory_groups":{
           "new":{ "<group-UUID>": { "Group Data Fields":"" } },
           "changed":{ "<group-UUID>": { "Group Data Fields":"" } },
           "summary":{
```

```
"new":#,
             "total":#,
             "unchanged":#
           } },
           "truncated": <true/false>
}
```

The `new`, `online`, `offline`, and `changed` objects are keyed by the modified asset IDs, with values in the **asset data** format. The `summary` field holds overall change statistics for this task. The `truncated` field is true when the change report is incomplete because it reached the maximum change threshold (1000 asset changes today).

runZero data dictionary

runZero discovers and catalogs these fields from known protocols and services:

Protocol	Attribute name	Data type	Single-value
acop	`acop.cellID`	string	true
acop	`acop.channelID`	string	true
acop	`acop.clientID`	string	true
acop	`acop.controllerName`	string	true
acop	`acop.controllerSerial`	string	true
acop	`acop.controllerTime`	string	true
acop	`acop.controllerVersion`	string	true
acop	`acop.error`	string	true
acop	`acop.opVersion`	string	true
acop	`acop.rbuType`	string	true
acop	`acop.revision`	string	true
acop	`acop.stationID`	string	true
acop	`acop.stationName`	string	true
acop	`acop.supplierCode`	string	true
acop	`acop.supportLinking`	numeric	true
acop	`acop.supportSeqNum`	numeric	true
acop	`acop.systemSubtype`	string	true
acop	`acop.systemType`	string	true
acop	`acop.toolData`	string	true
acop	`acop.toolVersion`	string	true

activemq	activemq.jvm.vendor	string	true
activemq	activemq.jvm.version	string	true
activemq	activemq.os	string	true
activemq	activemq.protocolVersion	string	true
activemq	activemq.wireformat.data	string	true
activemq	activemq.wireformat.version	string	true
adb	adb.access	string	true
ads	ads.deviceName	string	true
ads	ads.routeCount	numeric	true
ads	ads.routes	string	true
ads	ads.version	string	true
ads	ads.versionBuild	string	true
ads	ads.versionMajor	string	true
ads	ads.versionMinor	string	true
advantech-adam	advantech.firmware	string	true
advantech-adam	advantech.ipv4	string	true
advantech-adam	advantech.mac	string	true
advantech-adam	advantech.magic	string	true
advantech-adam	advantech.model	string	true
airplay	airplay.build	numeric	true
airplay	airplay.canRecordScreenStream	bool	true
airplay	airplay.deviceID	string	true
airplay	airplay.features	numeric	true
airplay	airplay.featuresEx	string	true
airplay	airplay.firmwareBuildDate	string	true
airplay	airplay.firmwareRevision	numeric	true
airplay	airplay.hardwareRevision	numeric	true
airplay	airplay.hasUDPMirroringSupport	bool	true
airplay	airplay.initialVolume	numeric	true
airplay	airplay.keepAliveLowPower	bool	true
airplay	airplay.keepAliveSendStatsAsBody	bool	true

airplay	airplay.macAddress	string	true
airplay	airplay.manufacturer	string	true
airplay	airplay.model	string	true
airplay	airplay.name	string	true
airplay	airplay.osBuildVersion	string	true
airplay	airplay.osinfo	string	true
airplay	airplay.pi	uuid	true
airplay	airplay.pk	string	true
airplay	airplay.playbackCapabilities	string	false
airplay	airplay.protocolVersion	numeric	true
airplay	airplay.ptpinfo	string	true
airplay	airplay.receiverHDRCapability	string	true
airplay	airplay.screenDemoMode	bool	true
airplay	airplay.sdk	string	true
airplay	airplay.senderAddress	string	true
airplay	airplay.sourceVersion	numeric	true
airplay	airplay.statusFlags	string	true
airplay	airplay.supportedFormats	string	true
airplay	airplay.volumeControlType	numeric	true
airplay	airplay.vv	numeric	true
airplay	apache.serverInfo	string	true
amqp	amqp.capabilities	string	true
amqp	amqp.clusterName	string	true
amqp	amqp.copyright	string	true
amqp	amqp.frameType	string	true
amqp	amqp.information	string	true
amqp	amqp.locales	string	true
amqp	amqp.mechanisms	string	true
amqp	amqp.platform	string	true
amqp	amqp.product	string	true
amqp	amqp.protocolID	string	true

amqp	amqp.protocolVersion	string	true
amqp	amqp.saslRequired	string	true
amqp	amqp.serverVersion	string	true
amqp	amqp.tlsRequired	string	true
amqp	amqp.version	string	true
atg	atg.cpe	string	true
atg	atg.functionCode	string	true
atg	atg.module	string	true
atg	atg.products	numeric	true
atg	atg.software	string	true
atg	atg.stationName	string	true
atg	atg.tankCount	numeric	true
atg	atg.version	string	true
bacnet	bacnet.address	string	true
bacnet	bacnet.bbmd	string	false
bacnet	bacnet.bbmdTable	string	true
bacnet	bacnet.databaseRevision	string	true
bacnet	bacnet.decodeWarning	string	false
bacnet	bacnet.deviceCount	numeric	true
bacnet	bacnet.deviceType	string	true
bacnet	bacnet.fdt	string	false
bacnet	bacnet.foreignDeviceRegistered	string	true
bacnet	bacnet.foreignDeviceTable	string	true
bacnet	bacnet.gateway	string	true
bacnet	bacnet.iHave	string	false
bacnet	bacnet.instance	string	true
bacnet	bacnet.instanceID	string	true
bacnet	bacnet.maxApdu	string	true
bacnet	bacnet.maxApduLengthAccepted	string	true
bacnet	bacnet.modelName	string	true
bacnet	bacnet.network	string	true

bacnet	bacnet.objectCount	numeric	true
bacnet	bacnet.pid	string	false
bacnet	bacnet.port	numeric	true
bacnet	bacnet.profileName	string	true
bacnet	bacnet.protocolRevision	string	true
bacnet	bacnet.readFailure	string	false
bacnet	bacnet.remote	string	true
bacnet	bacnet.remoteDevice	string	false
bacnet	bacnet.remoteDeviceCount	numeric	true
bacnet	bacnet.route	string	true
bacnet	bacnet.routerNetwork	string	false
bacnet	bacnet.routerNetworks	string	true
bacnet	bacnet.routingEntries	string	true
bacnet	bacnet.routingEntry	string	false
bacnet	bacnet.segmentation	string	true
bacnet	bacnet.segmentationSupported	numeric	true
bacnet	bacnet.serialNumber	string	true
bacnet	bacnet.systemStatus	string	true
bacnet	bacnet.target	string	true
bacnet	bacnet.vendorID	string	true
bacnet	bacnet.vendorIDLookup	string	true
bacnet	bacnet.vendorLookup	string	true
bedrock	bedrock.contents	string	true
bedrock	bedrock.guid	string	true
bedrock	bedrock.uptime	string	true
bgp	bgp.addressFamilies	string	true
bgp	bgp.asn	string	true
bgp	bgp.asn32	string	true
bgp	bgp.bgpID	string	true
bgp	bgp.capabilities	string	true
bgp	bgp.capabilityCount	numeric	true

bgp	bgp.error	string	true
bgp	bgp.holdTime	string	true
bgp	bgp.identifier	string	true
bgp	bgp.messageType	string	true
bgp	bgp.notificationCode	string	true
bgp	bgp.notificationCodeID	string	true
bgp	bgp.notificationSubcode	string	true
bgp	bgp.notificationSubcodeName	string	true
bgp	bgp.subCode	string	true
bgp	bgp.version	string	true
bitdefender-app	bitdefender.deviceID	string	true
bitdefender-app	bitdefender.hostname	string	true
bitdefender-app	bitdefender.model	string	true
bitdefender-app	bitdefender.os	string	true
bitdefender-app	bitdefender.type	string	true
bitdefender-app	service.product	string	true
bitdefender-app	service.vendor	string	true
bitdefender-app	service.version	string	true
bjnp	bjnp.ipv4	string	true
bjnp	bjnp.ipv6	string	true
bjnp	bjnp.macAddress	string	true
bjnp	bjnp.type	string	true
brother	brother.firmware.version	string	true
brother	brother.hostname	string	true
brother	brother.ip	string	true
brother	brother.mainFirmwareVersion	string	true
brother	brother.modelName	string	true
brother	brother.scanner	string	true
brother	brother.serialNbr	string	true
brother	brother.sub1FirmwareVersion	string	true
brother	brother.sub3FirmwareVersion	string	true

cacti	cacti.version	string	true
cassandra	cassandra.cluster	string	true
cassandra	cassandra.compression	string	true
cassandra	cassandra.confidence	string	true
cassandra	cassandra.cqlVersion	string	true
cassandra	cassandra.protocolVersions	string	true
cassandra	cassandra.version	string	true
cdp	cdp.addresses	string	true
cdp	cdp.capabilities	string	true
cdp	cdp.checksum	string	true
cdp	cdp.deviceID	string	true
cdp	cdp.isHost	string	true
cdp	cdp.isIGMPCapable	string	true
cdp	cdp.isL2Switch	string	true
cdp	cdp.isL3Router	string	true
cdp	cdp.isRepeater	string	true
cdp	cdp.isSourceRouteBridge	string	true
cdp	cdp.isTransparentBridge	string	true
cdp	cdp.mgmtAddresses	string	true
cdp	cdp.nativeVLAN	string	true
cdp	cdp.osVersion	string	true
cdp	cdp.platform	string	true
cdp	cdp.port	numeric	true
cdp	cdp.portID	numeric	true
cdp	cdp.powerConsumption	string	true
cdp	cdp.sysName	string	true
cdp	cdp.target	string	true
cdp	cdp.ttl	string	true
cdp	cdp.version	string	true
cdp	cdp.voipVLAN	string	true
chargen	chargen.osGuess	string	true

checkmk	banner	string	true
checkmk	checkmk.agentOS	string	true
checkmk	checkmk.arch	string	true
checkmk	checkmk.buildDate	string	true
checkmk	checkmk.hostname	string	true
checkmk	checkmk.osName	string	true
checkmk	checkmk.osType	string	true
checkmk	checkmk.osVersion	string	true
checkmk	checkmk.version	string	true
chromecast	chromecast.buildVersion	numeric	true
chromecast	chromecast.capabilities	string	true
chromecast	chromecast.castBuildRevision	string	true
chromecast	chromecast.connected	bool	true
chromecast	chromecast.ethernetConnected	bool	true
chromecast	chromecast.generation	numeric	true
chromecast	chromecast.hasUpdate	bool	true
chromecast	chromecast.hotspotBssid	string	true
chromecast	chromecast.ipAddress	string	true
chromecast	chromecast.locale	string	true
chromecast	chromecast.location.countryCode	string	true
chromecast	chromecast.location.latitude	numeric	true
chromecast	chromecast.location.longitude	numeric	true
chromecast	chromecast.macAddress	string	true
chromecast	chromecast.name	string	true
chromecast	chromecast.optIn.crash	bool	true
chromecast	chromecast.optIn.opencast	bool	true
chromecast	chromecast.optIn.stats	bool	true
chromecast	chromecast.publicKey	string	true
chromecast	chromecast.releaseTrack	string	true
chromecast	chromecast.setupState	numeric	true
chromecast	chromecast.setupStats.historicallySucceeded	bool	true

chromecast	chromecast.setupStats.numCheckConnectivity	numeric	true
chromecast	chromecast.setupStats.numConnectWifi	numeric	true
chromecast	chromecast.setupStats.numConnectedWifiNotSaved	numeric	true
chromecast	chromecast.setupStats.numInitialEurekaInfo	numeric	true
chromecast	chromecast.setupStats.numObtainIp	numeric	true
chromecast	chromecast.ssdpUdn	string	true
chromecast	chromecast.ssid	string	true
chromecast	chromecast.timeFormat	numeric	true
chromecast	chromecast.timezone	string	true
chromecast	chromecast.tosAccepted	bool	true
chromecast	chromecast.umaClientId	string	true
chromecast	chromecast.uptime	numeric	true
chromecast	chromecast.version	numeric	true
chromecast	chromecast.wpaConfigured	bool	true
chromecast	chromecast.wpaId	numeric	true
chromecast	chromecast.wpaState	numeric	true
cip	cip.address	string	true
cip	cip.backplaneSlot	string	true
cip	cip.deviceType	string	true
cip	cip.deviceTypeID	string	true
cip	cip.discoveredPorts	numeric	true
cip	cip.ethernetLinkCount	numeric	true
cip	cip.gateway	string	true
cip	cip.gateway.ethernetLinkCount	numeric	true
cip	cip.gateway.ips	string	true
cip	cip.gateway.macs	string	true
cip	cip.gateway.tcpipInterfaceCount	numeric	true
cip	cip.id	string	true
cip	cip.identityPort	numeric	true
cip	cip.ips	string	true
cip	cip.macPairs	string	true

cip	cip.macs	string	true
cip	cip.moduleCount	numeric	true
cip	cip.moduleStatus	string	true
cip	cip.modules	string	false
cip	cip.port	numeric	true
cip	cip.product	string	true
cip	cip.productCode	string	true
cip	cip.productCodeName	string	true
cip	cip.productID	string	true
cip	cip.productName	string	true
cip	cip.revision	string	true
cip	cip.safetyEnumerationSuppressed	string	true
cip	cip.safetySupervisor	string	true
cip	cip.serialNumber	string	true
cip	cip.serialNumberHex	string	true
cip	cip.slotStatus	string	true
cip	cip.state	string	true
cip	cip.stateCode	string	true
cip	cip.status	string	true
cip	cip.statusCode	string	true
cip	cip.supportsCommunicationService	numeric	true
cip	cip.supportsTCPEncapsulation	numeric	true
cip	cip.supportsUDPEncapsulation	numeric	true
cip	cip.target	string	true
cip	cip.tcpipInterfaceCount	numeric	true
cip	cip.vendor	string	true
cip	cip.vendorID	string	true
cisco-phone	ciscocp.apploadid	string	true
cisco-phone	ciscocp.bootloadid	string	true
cisco-phone	ciscocp.hostname	string	true
cisco-phone	ciscocp.mac	string	true

cisco-phone	ciscocp.messagewaiting	string	true
cisco-phone	ciscocp.model	string	true
cisco-phone	ciscocp.phonedn	string	true
cisco-phone	ciscocp.revision	string	true
cisco-phone	ciscocp.serial	string	true
cisco-phone	ciscocp.series	string	true
cisco-phone	ciscocp.udi	string	true
cisco-phone	ciscocp.version	string	true
ciscosmi	ciscosmi.reply	string	true
coap	coap.contentFormat	string	true
coap	coap.contentUnknown	string	true
coap	coap.encodedPayload	string	true
coap	coap.jsonData.cid	string	true
coap	coap.jsonData.name	string	true
coap	coap.jsonData.type	string	true
coap	coap.messageID	string	true
coap	coap.options	string	true
coap	coap.payload	string	true
coap	coap.resources	string	true
coap	coap.responseCode	string	true
coap	coap.token	string	true
coap	coap.type	string	true
coap	coap.version	string	true
cockpit	cockpit.os	string	true
cockpit	cockpit.variant	string	true
cockpit	cockpit.variantID	string	true
cockpit	host.name	string	false
codesys	codesys.banner	string	true
codesys	codesys.cpe23	string	true
codesys	codesys.format	string	true
codesys	codesys.osName	string	true

codesys	codesys.osType	string	true
codesys	codesys.productType	string	true
codesys	codesys.version	string	true
codesys2	codesys2.byteOrder	string	true
codesys2	codesys2.osType	string	true
codesys2	codesys2.osVersion	string	true
codesys2	codesys2.vendor	string	true
comtrol	dnsServer	string	true
comtrol	gateway	string	true
comtrol	hostname	string	true
comtrol	ipAddress	string	true
comtrol	macAddress	string	true
comtrol	manufacturer	string	true
comtrol	modelName	string	true
comtrol	modelNumber	string	true
comtrol	netmask	string	true
comtrol	osVersion	string	true
comtrol	serialNumber	string	true
confluence	confluence.baseURL	string	true
confluence	confluence.build	string	true
confluence	confluence.version	string	true
consul	consul.config.version	string	true
couchdb	couchdb.version	string	true
crestron	crestron.banner	string	true
crestron	crestron.buildDate	string	true
crestron	crestron.hostname	string	true
crestron	crestron.id	string	true
crestron	crestron.mac	string	true
crestron	crestron.model	string	true
crestron	crestron.version	string	true

crimsonv3	crimsonv3.manufacturer	string	true
crimsonv3	crimsonv3.model	string	true
crowd	crowd.version	string	true
crowd	service.product	string	true
crowd	service.vendor	string	true
crowd	service.version	string	true
daytime	daytime.osGuess	string	true
daytime	daytime.timestamp	string	true
db2	db2.database	string	true
db2	db2.extnam	string	true
db2	db2.hostname	string	true
db2	db2.instance	string	true
db2	db2.nodeName	string	true
db2	db2.port	numeric	true
db2	db2.raw	string	true
db2	db2.serverType	string	true
db2	db2.serviceName	string	true
db2	db2.srvnam	string	true
db2	db2.srvrlslv	string	true
db2	db2.version	string	true
dcerpc	dcerpc.accepted	string	true
dcerpc	dcerpc.acceptedContexts	numeric	true
dcerpc	dcerpc.assocGroup	string	true
dcerpc	dcerpc.authLength	string	true
dcerpc	dcerpc.byteOrder	string	true
dcerpc	dcerpc.callID	string	true
dcerpc	dcerpc.dataRep	string	true
dcerpc	dcerpc.fragLength	string	true
dcerpc	dcerpc.interface	string	true
dcerpc	dcerpc.interfaceUUID	string	true
dcerpc	dcerpc.interfaceVersion	string	true

dcerpc	dcerpc.maxRecvFrag	string	true
dcerpc	dcerpc.maxXmitFrag	string	true
dcerpc	dcerpc.oxid.addresses	string	true
dcerpc	dcerpc.oxid.bindingCount	numeric	true
dcerpc	dcerpc.oxid.bindings	string	true
dcerpc	dcerpc.oxid.comVersion	string	true
dcerpc	dcerpc.oxid.error	string	true
dcerpc	dcerpc.oxid.machineName	string	true
dcerpc	dcerpc.oxid.security	string	true
dcerpc	dcerpc.pduType	string	true
dcerpc	dcerpc.port	numeric	true
dcerpc	dcerpc.rejectReason	string	true
dcerpc	dcerpc.rejectReasonCode	string	true
dcerpc	dcerpc.rejectedContexts	numeric	true
dcerpc	dcerpc.secondaryAddr	string	true
dcerpc	dcerpc.serviceCount	numeric	true
dcerpc	dcerpc.services	string	true
dcerpc	dcerpc.target	string	true
dcerpc	dcerpc.version	string	true
dcerpc	epm.addrLen	numeric	true
dcerpc	epm.address	string	true
dcerpc	epm.assocGroup	string	true
dcerpc	epm.authRequired	string	true
dcerpc	epm.dataRep	string	true
dcerpc	epm.maxRecvFrag	numeric	true
dcerpc	epm.maxSendFrag	numeric	true
dcerpc	epm.oxid.addresses	string	true
dcerpc	epm.oxid.security	string	true
dcerpc	epm.oxidVersion	string	true
dcerpc	epm.version	string	true
dhcp	dhcp.authentication	string	true

dhcp	dhcp.bootFile	string	true
dhcp	dhcp.broadcast	string	true
dhcp	dhcp.broadcastAddress	string	true
dhcp	dhcp.class	string	false
dhcp	dhcp.clientFqdn	string	true
dhcp	dhcp.clientIP	string	true
dhcp	dhcp.clientMAC	string	true
dhcp	dhcp.clientSystemArchitecture	string	false
dhcp	dhcp.clientSystemArchitectureCode	string	true
dhcp	dhcp.clientid.enterprise	string	false
dhcp	dhcp.domainName	string	true
dhcp	dhcp.domainNameServer	string	true
dhcp	dhcp.flags	string	true
dhcp	dhcp.hardwareAddressLength	string	true
dhcp	dhcp.hardwareType	string	true
dhcp	dhcp.hops	string	true
dhcp	dhcp.hostname	string	true
dhcp	dhcp.isRequest	string	true
dhcp	dhcp.leaseTime	string	true
dhcp	dhcp.leaseTimeFormatted	string	true
dhcp	dhcp.maximumMessageSize	string	true
dhcp	dhcp.message	string	true
dhcp	dhcp.messageType	string	true
dhcp	dhcp.messageTypeCode	string	true
dhcp	dhcp.nextServerIP	string	true
dhcp	dhcp.ntpServers	string	true
dhcp	dhcp.offeredIP	string	true
dhcp	dhcp.op	string	true
dhcp	dhcp.port	numeric	true
dhcp	dhcp.rebindingTime	string	true
dhcp	dhcp.rebindingTimeFormatted	string	true

dhcp	dhcp.relayAgentIP	string	true
dhcp	dhcp.relayAgentInformation	string	true
dhcp	dhcp.renewalTime	string	true
dhcp	dhcp.renewalTimeFormatted	string	true
dhcp	dhcp.router	string	true
dhcp	dhcp.seconds	string	true
dhcp	dhcp.serverIdentifier	string	true
dhcp	dhcp.serverName	string	true
dhcp	dhcp.subnetMask	string	true
dhcp	dhcp.target	string	true
dhcp	dhcp.tftpServerAddress	string	true
dhcp	dhcp.tftpServerName	string	true
dhcp	dhcp.transactionID	string	true
dhcp	dhcp.vendorClass	string	true
diameter	diameter.cpe	string	true
diameter	diameter.firmwareRevision	string	true
diameter	diameter.originHost	string	true
diameter	diameter.originRealm	string	true
diameter	diameter.productName	string	true
diameter	diameter.resultCode	string	true
diameter	diameter.vendor	string	true
diameter	diameter.vendorID	string	true
diametersctp	diameter.cpe	string	true
diametersctp	diameter.firmwareRevision	string	true
diametersctp	diameter.originHost	string	true
diametersctp	diameter.originRealm	string	true
diametersctp	diameter.productName	string	true
diametersctp	diameter.vendor	string	true
diametersctp	diameter.vendorID	string	true
digi-realport	realport.banner	string	true
digi-realport	realport.baud	numeric	true

digi-realport	realport.name	string	true
digi-realport	realport.portCount	numeric	true
digi-realport	realport.serialPort	numeric	true
digi-realport	realport.version	string	true
dnp3	dnp3.destination	string	true
dnp3	dnp3.deviceID	string	true
dnp3	dnp3.deviceName	string	true
dnp3	dnp3.hardwareVersion	string	true
dnp3	dnp3.location	string	true
dnp3	dnp3.manufacturer	string	true
dnp3	dnp3.model	string	true
dnp3	dnp3.owner	string	true
dnp3	dnp3.serialNumber	string	true
dnp3	dnp3.softwareVersion	string	true
dnp3	dnp3.source	string	true
dns	dns.addrs	string	true
dns	dns.ancount	numeric	true
dns	dns.answerAddrs	string	true
dns	dns.answerNames	string	true
dns	dns.answerTXTs	string	true
dns	dns.answerTypes	string	true
dns	dns.authors.bind	string	true
dns	dns.edns0	string	true
dns	dns.edns0.cookie	string	true
dns	dns.edns0.nsid	string	true
dns	dns.edns0.subnet	string	true
dns	dns.edns0.udpSize	string	true
dns	dns.edns0.unknown	string	true
dns	dns.edns0.version	string	true
dns	dns.flags	string	true
dns	dns.hostname.bind	string	true

dns	dns.hostnameBind	string	true
dns	dns.id	string	true
dns	dns.id.server	string	true
dns	dns.idServer	string	true
dns	dns.isRequest	string	true
dns	dns.meraki	string	true
dns	dns.opcode	string	true
dns	dns.ptr	string	false
dns	dns.qdcount	numeric	true
dns	dns.rcode	string	true
dns	dns.recursion	string	true
dns	dns.resolvers	string	true
dns	dns.resolves	string	true
dns	dns.rtts	numeric	true
dns	dns.subnets	numeric	true
dns	dns.telco.epc	string	false
dns	dns.tracer	string	false
dns	dns.tracer.edns0	string	false
dns	dns.transport	string	true
dns	dns.version	string	true
dns	dns.version.bind	string	true
dns	dns.version.server	string	true
dns	dns.versionBind	string	true
docker	docker.architecture	string	true
docker	docker.dockerRootDir	string	true
docker	docker.kernelVersion	string	true
docker	docker.memTotal	string	true
docker	docker.name	string	true
docker	docker.ncpu	string	true
docker	docker.operatingSystem	string	true
docker	docker.ostype	string	true

docker	docker.systemTime	string	true
docker	docker.version	string	true
dtls	dtls.alert	string	true
dtls	dtls.contentType	string	true
dtls	dtls.cookieLength	string	true
dtls	dtls.encapsulation	string	true
dtls	dtls.handshakeFragmentLength	string	true
dtls	dtls.handshakeFragmentOffset	string	true
dtls	dtls.handshakeLength	string	true
dtls	dtls.handshakeMessageSeq	string	true
dtls	dtls.handshakeType	string	true
dtls	dtls.recordEpoch	string	true
dtls	dtls.recordLength	string	true
dtls	dtls.recordSequence	string	true
dtls	dtls.serverVersion	string	true
dtls	dtls.unknown	string	true
dtls	dtls.verifyCookie	string	true
dtls	dtls.version	string	true
echo	echo.data	string	true
echo	echo.match	string	true
echo	echo.probeHex	string	true
echo	echo.probeLength	string	true
echo	echo.responseHex	string	true
echo	echo.responseLength	string	true
elasticsearch	elasticsearch.version.number	string	true
epo	epo.guid	string	true
epo	epo.hostname	string	true
epo	epo.server	string	true
epo	epo.version	string	true
epo	mcafeeAgent.hostname	string	true
epo	mcafeeAgent.version	string	true

erlangdp	epmd.details	string	false
erlangdp	epmd.names	string	false
erlangdp	erldp.challenge	string	true
erlangdp	erldp.flags	string	true
erlangdp	erldp.hostname	string	true
erlangdp	erldp.node	string	true
erlangdp	erldp.status	string	true
erlangdp	erldp.version	string	true
etcd2	etcd2.access	string	true
etcd2	etcd2.keys	string	true
ethercat	ethercat.ado	string	true
ethercat	ethercat.adp	string	true
ethercat	ethercat.command	string	true
ethercat	ethercat.commandType	string	true
ethercat	ethercat.datagramCount	numeric	true
ethercat	ethercat.detected	bool	true
ethercat	ethercat.index	string	true
ethercat	ethercat.workingCounter	numeric	true
fgfm	fgfm.deviceName	string	true
fgfm	fgfm.firmwareVersion	string	true
fgfm	fgfm.mgmtIP	string	true
fgfm	fgfm.platform	string	true
fgfm	fgfm.serialNumber	string	true
firebird	firebird.opcode	string	true
firebird	firebird.protocolVersion	string	true
firebird	firebird.version	string	true
fox	fox.appName	string	true
fox	fox.appVersion	string	true
fox	fox.authAgent	string	true
fox	fox.brandId	string	true
fox	fox.cpe	string	true

fox	fox.hostAddress	string	true
fox	fox.hostId	string	true
fox	fox.hostname	string	true
fox	fox.language	string	true
fox	fox.osName	string	true
fox	fox.osVersion	string	true
fox	fox.stationName	string	true
fox	fox.sysInfo	string	true
fox	fox.timeZone	string	true
fox	fox.vendor	string	true
fox	fox.version	string	true
fox	fox.vmName	string	true
fox	fox.vmUuid	string	true
fox	fox.vmVersion	string	true
gesrtp	gesrtp.cpe	string	true
gesrtp	gesrtp.deviceIndicator	string	true
gesrtp	gesrtp.plcName	string	true
git	git.branches	string	true
git	git.capabilities	string	true
git	git.error	string	true
git	git.headRef	string	true
git	git.protocolVersion	string	true
git	git.refs	string	true
git	git.serverSoftware	string	true
git	git.tags	string	true
git-http	gitlab.manifest.hash	string	true
googlewifi	googleWifi.software.version	string	true
googlewifi	googleWifi.system.countryCode	string	true
googlewifi	googleWifi.system.hardwareID	string	true
googlewifi	googleWifi.system.modelID	string	true
googlewifi	googleWifi.wan.gateway	string	true

googlewifi	`googleWifi.wan.localIP`	string	true
gtpc	`gtpc.restartCounter`	numeric	true
gtpc	`gtpc.teid`	string	true
gtpc	`gtpc.teidPresent`	string	true
gtpc	`gtpc.version`	string	true
gtpprime	`gtpprime.detected`	bool	true
gtpprime	`gtpprime.pt`	string	true
gtpprime	`gtpprime.restartCounter`	numeric	true
gtpprime	`gtpprime.teid`	string	true
gtpprime	`gtpprime.version`	string	true
gtpu	`gtpu.detected`	bool	true
gtpu	`gtpu.pt`	string	true
gtpu	`gtpu.restartCounter`	numeric	true
gtpu	`gtpu.sequenceNumber`	string	true
gtpu	`gtpu.teid`	string	true
gtpu	`gtpu.version`	string	true
hartip	`hartip.inactivityTimeout`	string	true
hartip	`hartip.masterType`	string	true
hartip	`hartip.messageType`	string	true
hartip	`hartip.status`	string	true
hartip	`hartip.statusDescription`	string	true
hartip	`hartip.transactionID`	string	true
hartip	`hartip.version`	string	true
hiddiscoveryd	`hiddiscoveryd.address`	string	true
hiddiscoveryd	`hiddiscoveryd.hostname`	string	true
hiddiscoveryd	`hiddiscoveryd.mac`	string	true
hiddiscoveryd	`hiddiscoveryd.model`	string	true
hiddiscoveryd	`hiddiscoveryd.unpatchedVertXploit`	string	true
hiddiscoveryd	`hiddiscoveryd.version`	string	true
hiddiscoveryd	`hiddiscoveryd.versionDate`	string	true
hikvision	`hikvision.codebase.lastModified`	string	true

hikvision	hikvision.codebase.platform	string	true
hikvision	hikvision.codebase.plugins	string	false
hikvision	hikvision.deviceCode	string	true
hikvision	hikvision.deviceType	string	true
hikvision	hikvision.firmwareVersion	string	true
hikvision	hikvision.firmwareVersionEmbedded	string	true
hikvision	hikvision.loginPageLastModified	string	true
hikvision	hikvision.modelFamily	string	true
hikvision	hikvision.pluginVersion	string	true
hikvision	hikvision.realm	string	true
hikvision	hikvision.webVersion	string	true
http	html.copyright	string	true
http	html.favicon	string	true
http	html.generator	string	true
http	html.title	string	true
http	http.body	string	true
http	http.body.mmh3	string	true
http	http.code	numeric	true
http	http.contentLength	string	true
http	http.contentType	string	true
http	http.head.acceptRanges	string	true
http	http.head.cacheControl	string	true
http	http.head.contentLength	numeric	true
http	http.head.contentSecurityPolicy	string	true
http	http.head.contentType	string	true
http	http.head.cookieNames	string	true
http	http.head.date	string	true
http	http.head.etag	string	true
http	http.head.expires	string	true
http	http.head.faviconPath	string	true
http	http.head.lastModified	string	true

http	http.head.location	string	true
http	http.head.microsoftsharepointteamservices	string	true
http	http.head.server	string	true
http	http.head.setCookie	string	true
http	http.head.spiislatency	string	true
http	http.head.vary	string	true
http	http.head.wwwAuthenticate	string	true
http	http.head.xConfluenceRequestTime	string	true
http	http.head.xInfluxdbBuild	string	true
http	http.head.xInfluxdbVersion	string	true
http	http.head.xJenkins	string	true
http	http.head.xNtnxEnv	string	true
http	http.head.xOracleDmsEcid	string	true
http	http.head.xOracleDmsRid	string	true
http	http.head.xPoweredBy	string	true
http	http.head.xSharepointhealthscore	string	true
http	http.head.xconfluencerequesttime	string	true
http	http.head.xframeoptions	string	true
http	http.location	string	true
http	http.message	string	true
http	http.method	string	true
http	http.ntlm.challenge	string	true
http	http.ntlm.path	string	true
http	http.owa.version	string	true
http	http.owa.version.full	string	true
http	http.path	string	true
http	http.port	numeric	true
http	http.server	string	true
http	http.setCookie	string	true
http	http.sonicwall.hardware	string	true
http	http.sonicwall.softwareVersion	string	true

http	http.status	string	true
http	http.statusCode	string	true
http	http.target	string	true
http	http.technologies	string	true
http	http.title	string	true
http	http.uri	string	true
http	http.url	string	true
http	http.version	string	true
http	http.vmware.thumbprint	string	true
http	http.vmware.username	string	true
http	landesk.configPath	string	true
http	landesk.providerVersion	string	true
http	landesk.serverVersion	string	true
iax2	iax2.destCallNumber	string	true
iax2	iax2.detected	bool	true
iax2	iax2.iseqno	string	true
iax2	iax2.oseqno	string	true
iax2	iax2.response	string	true
iax2	iax2.sourceCallNumber	string	true
iax2	iax2.subclass	string	true
iax2	iax2.subclassName	string	true
iax2	iax2.ts	numeric	true
icmp	icmp.addrs	string	true
icmp	icmp.rtts	string	true
icmp	icmp.typeCode	string	true
icmp	icmp.typeCodeN	string	true
icmp	icmp6.addrs	string	true
icmp	icmp6.rtts	string	true
icmp	ip.tos	string	false
icmp	ip.ttl	string	false
ident	ident.error	string	true

ident	ident.opSys	string	true
ident	ident.osGuess	string	true
ident	ident.username	string	true
idrac	idrac.adEnabled	bool	true
idrac	idrac.fwVersion	string	true
idrac	idrac.gui.titleBar	string	true
idrac	idrac.hostname	string	true
idrac	idrac.isOEMBranded	bool	true
idrac	idrac.license	string	true
idrac	idrac.name	string	true
idrac	idrac.oem.hostname	string	true
idrac	idrac.serverGen	string	true
idrac	idrac.ssoEnabled	bool	true
idrac	idrac.status	string	true
idrac	idrac.sysDesc	string	true
idrac	idrac.systemLockdown	string	true
idrac	idrac.systemModelName	string	true
idrac	idrac.tfaEnabled	bool	true
idrac	idrac.version	numeric	true
idrac	service.product	string	true
idrac	service.vendor	string	true
idrac	service.version	string	true
iec104	iec104.asduCOT	string	true
iec104	iec104.asduCommonAddr	string	true
iec104	iec104.asduCommonAddrs	string	true
iec104	iec104.asduInfoObjAddr	string	true
iec104	iec104.asduNegative	string	true
iec104	iec104.asduTest	string	true
iec104	iec104.asduTypes	string	true
iec104	iec104.command	string	true
iec104	iec104.commands	string	true

iec104	iec104.frameCount	numeric	true
iec104	iec104.frameType	string	true
iec104	iec104.frameTypes	string	true
iec104	iec104.initCause	string	true
iec104	iec104.initLocalFlag	string	true
iec104	iec104.startDTConfirmed	string	true
iec104	iec104.testFRConfirmed	string	true
igel-discovery	igel.firmwareVersion	string	true
igel-discovery	igel.hostname	string	true
igel-discovery	igel.ipAddress	string	true
igel-discovery	igel.macAddress	string	true
igel-discovery	igel.osType	string	true
igel-discovery	igel.productID	string	true
igel-discovery	igel.productName	string	true
iis	http.owa.version	string	true
iis	http.owa.version.full	string	true
iis	rdg.authScheme	string	true
iis	rdg.transport	string	true
ike	ike.authMethod	string	true
ike	ike.dhGroup	string	true
ike	ike.encryptionAlgo	string	true
ike	ike.esn	string	true
ike	ike.exchangeType	string	true
ike	ike.flags	string	true
ike	ike.hashAlgo	string	true
ike	ike.initiatorSPI	string	true
ike	ike.integrityAlgo	string	true
ike	ike.keyLength	string	true
ike	ike.lifeDuration	string	true
ike	ike.lifeType	string	true
ike	ike.messageID	string	true

ike	ike.messageLength	string	true
ike	ike.nextPayload	string	true
ike	ike.nonESPMarker	string	true
ike	ike.notifyTypes	string	true
ike	ike.payload	string	true
ike	ike.payloadCount	numeric	true
ike	ike.prfAlgo	string	true
ike	ike.reply	string	true
ike	ike.responderSPI	string	true
ike	ike.saProtocol	string	true
ike	ike.sha1	string	true
ike	ike.vendorID	string	true
ike	ike.vendorIDCount	numeric	true
ike	ike.version	string	true
influxdb	influxdb.build	string	true
influxdb	influxdb.databases	string	false
influxdb	influxdb.version	string	true
ipmi	ipmi.additionalSupport	numeric	true
ipmi	ipmi.additionalSupportRaw	numeric	true
ipmi	ipmi.authAnon	string	true
ipmi	ipmi.authNull	string	true
ipmi	ipmi.authTypes	string	true
ipmi	ipmi.auxFirmwareRev	string	true
ipmi	ipmi.channel	string	true
ipmi	ipmi.cipherZero	string	true
ipmi	ipmi.ciphers	string	true
ipmi	ipmi.ciphersPageError	string	true
ipmi	ipmi.ciphersRaw	string	true
ipmi	ipmi.ciphersText	string	true
ipmi	ipmi.command	string	true
ipmi	ipmi.completionCode	string	true

ipmi	ipmi.completionCodeText	string	true
ipmi	ipmi.connVersions	string	true
ipmi	ipmi.connVersionsRaw	string	true
ipmi	ipmi.deviceID	string	true
ipmi	ipmi.deviceRevision	string	true
ipmi	ipmi.deviceRevisionRaw	string	true
ipmi	ipmi.fakeUsername	string	true
ipmi	ipmi.firmware	string	true
ipmi	ipmi.firmwareRevision	string	true
ipmi	ipmi.guid.mac	string	true
ipmi	ipmi.guid.mac.vendor	string	true
ipmi	ipmi.guid.partID	string	true
ipmi	ipmi.guid.serial	string	true
ipmi	ipmi.ipmiVersion	string	true
ipmi	ipmi.manufacturer	string	true
ipmi	ipmi.manufacturerID	string	true
ipmi	ipmi.oemData	string	true
ipmi	ipmi.oemID	string	true
ipmi	ipmi.oemName	string	true
ipmi	ipmi.passAuth	string	true
ipmi	ipmi.passAuthRaw	string	true
ipmi	ipmi.productID	string	true
ipmi	ipmi.rakp.authCiphers	string	true
ipmi	ipmi.rakp.count	numeric	true
ipmi	ipmi.rakp.cracked	string	true
ipmi	ipmi.rakp.crackedCount	numeric	true
ipmi	ipmi.rakp.guids	string	true
ipmi	ipmi.rakp.hashes	string	true
ipmi	ipmi.rakp.passwords	string	true
ipmi	ipmi.rakp.sessionIDs	string	true
ipmi	ipmi.rakp.usernames	string	true

ipmi	ipmi.response.commandMismatch	string	true
ipmi	ipmi.response.data	string	true
ipmi	ipmi.response.dataLen	string	true
ipmi	ipmi.response.parseError	string	true
ipmi	ipmi.response.raw	string	true
ipmi	ipmi.response.text	string	true
ipmi	ipmi.rmcpPlus	string	true
ipmi	ipmi.session.ID	string	true
ipmi	ipmi.session.activateAuthType	string	true
ipmi	ipmi.session.activateAuthTypeName	string	true
ipmi	ipmi.session.activateID	string	true
ipmi	ipmi.session.activateInboundSeq	string	true
ipmi	ipmi.session.activateMaxPriv	string	true
ipmi	ipmi.session.activateMaxPrivName	string	true
ipmi	ipmi.session.activateStatus	string	true
ipmi	ipmi.session.challenge	string	true
ipmi	ipmi.session.challengeAuthType	string	true
ipmi	ipmi.session.challengeAuthTypeName	string	true
ipmi	ipmi.session.challengeID	string	true
ipmi	ipmi.session.challengeStatus	string	true
ipmi	ipmi.session.consoleID	string	true
ipmi	ipmi.session.status	string	true
ipmi	ipmi.session.statusCode	string	true
ipmi	ipmi.session.username	string	true
ipmi	ipmi.srcAddr	string	true
ipmi	ipmi.srcLun	string	true
ipmi	ipmi.system.guid	string	true
ipmi	ipmi.tgtAddr	string	true
ipmi	ipmi.tgtLun	string	true
ipmi	ipmi.userAuth	string	true
ipmi	ipmi.userAuthRaw	string	true

ipmi	ipmi.v15.authMethods	string	true
ipmi	ipmi.v15.authNonePrivilege	string	true
ipmi	ipmi.v15.authNoneProof	string	true
ipmi	ipmi.v15.authNoneSession	string	true
ipmi	ipmi.v15.authNoneSessionID	string	true
ipmi	ipmi.v2.0	string	true
ipmi	ipmi.weakCiphers	string	true
ippbrowse	ippbrowse.info	string	true
ippbrowse	ippbrowse.location	string	true
ippbrowse	ippbrowse.makeModel	string	true
ippbrowse	ippbrowse.state	string	true
ippbrowse	ippbrowse.type	string	true
ippbrowse	ippbrowse.uri	string	true
irc	irc.banner	string	true
irc	irc.channelCount	numeric	true
irc	irc.channelModes	string	true
irc	irc.core.sim	string	true
irc	irc.cpe23	string	true
irc	irc.createdDate	string	true
irc	irc.error	string	true
irc	irc.networkName	string	true
irc	irc.product	string	true
irc	irc.serverName	string	true
irc	irc.serverSoftware	string	true
irc	irc.userCount	numeric	true
irc	irc.userModes	string	true
irc	irc.vendor	string	true
irc	irc.version	string	true
iscsi	iscsi.acknowledge	string	true
iscsi	iscsi.authMethod	string	true
iscsi	iscsi.bidiReadResidualCount	numeric	true

iscsi	iscsi.bufferOffset	string	true
iscsi	iscsi.continue	string	true
iscsi	iscsi.currentStage	string	true
iscsi	iscsi.dataDigest	string	true
iscsi	iscsi.dataSN	string	true
iscsi	iscsi.expCmdSN	string	true
iscsi	iscsi.expDataSN	string	true
iscsi	iscsi.final	string	true
iscsi	iscsi.flags.readResidualOverflow	string	true
iscsi	iscsi.flags.readResidualUnderflow	string	true
iscsi	iscsi.flags.residualOverflow	string	true
iscsi	iscsi.flags.residualUnderflow	string	true
iscsi	iscsi.flags.status	string	true
iscsi	iscsi.headerDigest	string	true
iscsi	iscsi.immediateDelivery	string	true
iscsi	iscsi.initiatorTaskTag	string	true
iscsi	iscsi.isid	string	true
iscsi	iscsi.keyvalue.MaxRecvDataSegmentLength	string	true
iscsi	iscsi.keyvalue.authMethod	string	true
iscsi	iscsi.keyvalue.dataDigest	string	true
iscsi	iscsi.keyvalue.defaultTime2Retain	string	true
iscsi	iscsi.keyvalue.defaultTime2Wait	string	true
iscsi	iscsi.keyvalue.errorRecoveryLevel	string	true
iscsi	iscsi.keyvalue.headerDigest	string	true
iscsi	iscsi.keyvalue.ifmarker	string	true
iscsi	iscsi.keyvalue.maxConnections	numeric	true
iscsi	iscsi.keyvalue.ofmarker	string	true
iscsi	iscsi.keyvalue.targetAddress	string	true
iscsi	iscsi.keyvalue.targetName	string	true
iscsi	iscsi.keyvalue.xCom.cisco.pingTimeout	string	true
iscsi	iscsi.keyvalue.xCom.cisco.protocol	string	true

iscsi	iscsi.keyvalue.xCom.cisco.sendAsyncText	string	true
iscsi	iscsi.logout.response	string	true
iscsi	iscsi.lun	string	true
iscsi	iscsi.lunAddressMode	string	true
iscsi	iscsi.maxCmdSN	string	true
iscsi	iscsi.nextStage	string	true
iscsi	iscsi.opcode	string	true
iscsi	iscsi.residualCount	numeric	true
iscsi	iscsi.response	string	true
iscsi	iscsi.statSN	string	true
iscsi	iscsi.status	string	true
iscsi	iscsi.statusClass	string	true
iscsi	iscsi.statusDetail	string	true
iscsi	iscsi.targetAddress	string	true
iscsi	iscsi.targetAddresses	string	true
iscsi	iscsi.targetCount	string	true
iscsi	iscsi.targetName	string	true
iscsi	iscsi.targetNames	string	true
iscsi	iscsi.time2Retain	string	true
iscsi	iscsi.time2Wait	string	true
iscsi	iscsi.transferTaskTag	string	true
iscsi	iscsi.transit	string	true
iscsi	iscsi.tsih	string	true
iscsi	iscsi.unknownData	string	true
iscsi	iscsi.versionActive	string	true
iscsi	iscsi.versionMax	string	true
iscsi	iscsi.versionMin	string	true
iua	iua.errorCode	string	true
iua	iua.infoString	string	true
iua	iua.messageClass	string	true
iua	iua.messageType	string	true

jdwp	jdwp.banner	string	true
jdwp	jdwp.description	string	true
jdwp	jdwp.major	string	true
jdwp	jdwp.minor	string	true
jdwp	jdwp.packetID	string	true
jdwp	jdwp.version	string	true
jdwp	jdwp.vmName	string	true
jdwp	jdwp.vmVersion	string	true
jetdirect	jetdirect.banner	string	true
jetdirect	jetdirect.cpe23	string	true
jetdirect	jetdirect.filesystemAccess	string	true
jetdirect	jetdirect.firmware	string	true
jetdirect	jetdirect.id	string	true
jetdirect	jetdirect.model	string	true
jetdirect	jetdirect.status	string	true
jetdirect	jetdirect.vendor	string	true
jetdirect	pjl.id	string	true
jira	jira.baseURL	string	true
jira	jira.build	string	true
jira	jira.version	string	true
jms	jms.admin	string	true
jms	jms.cluster	string	true
jms	jms.jms	string	true
jms	jms.jmsdirect	string	true
jms	jms.jmxrmi	string	true
jms	jms.jmxrmi.env.url	string	true
jms	jms.jmxrmi.env.version	string	true
jms	jms.mqdirect2	string	true
jms	jms.portmapper	numeric	true
jms	jms.portmapper.env.imqhome	numeric	true
jms	jms.portmapper.env.imqvarhome	numeric	true

jms	jms.portmapper.env.sessionid	numeric	true
jms	jms.services	string	false
jms	jms.tcp.ports	numeric	false
jms	jms.version	string	true
kafka	kafka.correlationID	string	true
kafka	kafka.version	string	true
kasa	kasa.deviceAlias	string	true
kasa	kasa.deviceId	string	true
kasa	kasa.deviceType	string	true
kasa	kasa.hardwareAddress	string	true
kasa	kasa.hardwareId	string	true
kasa	kasa.hardwareVersion	string	true
kasa	kasa.model	string	true
kasa	kasa.obdSource	string	true
kasa	kasa.oemId	string	true
kasa	kasa.softwareVersion	string	true
kerberos	kerberos.error	string	true
kerberos	kerberos.errorCode	string	true
kerberos	kerberos.errorCodeName	string	true
kerberos	kerberos.messageType	string	true
kerberos	kerberos.messageTypeCode	string	true
kerberos	kerberos.microseconds	string	true
kerberos	kerberos.port	numeric	true
kerberos	kerberos.realm	string	true
kerberos	kerberos.serverTS	numeric	true
kerberos	kerberos.servicePrincipal	string	true
kerberos	kerberos.target	string	true
kerberos	kerberos.ticket.realm	string	true
kerberos	kerberos.version	string	true
knxnet	knxnet.address	string	true
knxnet	knxnet.bus.address	string	true

knxnet	knxnet.bus.applicationID	string	true
knxnet	knxnet.bus.descriptor	string	true
knxnet	knxnet.bus.firmwareRevision	string	true
knxnet	knxnet.bus.hardwareType	string	true
knxnet	knxnet.bus.manufacturerID	string	true
knxnet	knxnet.bus.name	string	true
knxnet	knxnet.bus.orderInfo	string	true
knxnet	knxnet.bus.programVersion	string	true
knxnet	knxnet.bus.serialNumber	string	true
knxnet	knxnet.busDevice	string	false
knxnet	knxnet.busDeviceCount	numeric	true
knxnet	knxnet.channel	string	true
knxnet	knxnet.decodeErrors	string	true
knxnet	knxnet.decodeWarning	string	false
knxnet	knxnet.descriptor	string	true
knxnet	knxnet.firmwareRevision	string	true
knxnet	knxnet.hardwareType	string	true
knxnet	knxnet.hpaiDataEP	string	false
knxnet	knxnet.mac	string	true
knxnet	knxnet.manufacturerData	string	true
knxnet	knxnet.manufacturerID	string	true
knxnet	knxnet.medium	string	true
knxnet	knxnet.multicast	string	true
knxnet	knxnet.multicastAddress	string	true
knxnet	knxnet.name	string	true
knxnet	knxnet.orderInfo	string	true
knxnet	knxnet.remoteDevice	string	false
knxnet	knxnet.remoteDeviceCount	numeric	true
knxnet	knxnet.serial	string	true
knxnet	knxnet.serialNumber	string	true
knxnet	knxnet.service.Core	string	true

knxnet	knxnet.service.Tunnelling	string	true
knxnet	knxnet.serviceFamilies	string	true
knxnet	knxnet.status	string	true
knxnet	knxnet.tunnelData	string	true
knxnet	knxnet.tunnelEP	string	false
knxnet	knxnet.type	string	true
l2t	l2t.assignedTunnelID	string	true
l2t	l2t.bearerCaps	string	true
l2t	l2t.firmwareRevision	string	true
l2t	l2t.framingCaps	string	true
l2t	l2t.hostName	string	true
l2t	l2t.messageType	string	true
l2t	l2t.protocolVersion	string	true
l2t	l2t.vendorName	string	true
l2tp	l2tp.assignedTunnelID	string	true
l2tp	l2tp.bearerCaps	string	true
l2tp	l2tp.controlCode	string	true
l2tp	l2tp.errorCode	string	true
l2tp	l2tp.errorMsg	string	true
l2tp	l2tp.firmwareRevision	string	true
l2tp	l2tp.framingCaps	string	true
l2tp	l2tp.hostName	string	true
l2tp	l2tp.nr	string	true
l2tp	l2tp.ns	string	true
l2tp	l2tp.protoVersion	string	true
l2tp	l2tp.recvWindowSize	string	true
l2tp	l2tp.resultCode	string	true
l2tp	l2tp.sessionID	string	true
l2tp	l2tp.tunnelID	string	true
l2tp	l2tp.unknownType	string	true
l2tp	l2tp.vendorName	string	true

landesk	landesk.configPath	string	true
landesk	landesk.providerVersion	string	true
landesk	landesk.serverVersion	string	true
langflow	service.product	string	true
langflow	service.version	string	true
lantronix	lantronix.fwinfo	string	true
lantronix	lantronix.mac	string	true
lantronix	lantronix.macVendor	string	true
lantronix	lantronix.serial	string	true
lantronix	lantronix.type	string	true
lantronix	lantronix.version	string	true
ldap	ldap.auth	string	true
ldap	ldap.configurationNamingContext	string	true
ldap	ldap.currentTime	string	true
ldap	ldap.defaultNamingContext	string	true
ldap	ldap.diagMessage	string	true
ldap	ldap.dnsHostName	string	true
ldap	ldap.domainControllerFunctionality	string	true
ldap	ldap.domainFunctionality	string	true
ldap	ldap.dsServiceName	string	true
ldap	ldap.error	string	true
ldap	ldap.forestFunctionality	string	true
ldap	ldap.isGlobalCatalogReady	string	true
ldap	ldap.isSynchronized	string	true
ldap	ldap.matchedDN	string	true
ldap	ldap.namingContexts	numeric	true
ldap	ldap.notes	string	false
ldap	ldap.resultCode	string	true
ldap	ldap.resultCodeName	string	true
ldap	ldap.rootDomainNamingContext	string	true
ldap	ldap.schemaNamingContext	string	true

ldap	ldap.searchresult	string	true
ldap	ldap.serverName	string	true
ldap	ldap.serverType	string	true
ldap	ldap.startTLS	string	true
ldap	ldap.subschemaSubentry	string	true
ldap	ldap.supportedControl	string	true
ldap	ldap.supportedControls	numeric	true
ldap	ldap.supportedExtension	string	true
ldap	ldap.supportedExtensions	numeric	true
ldap	ldap.supportedLDAPVersion	string	true
ldap	ldap.supportedSASLMechanisms	numeric	true
ldap	ldap.supportsStartTLS	numeric	true
ldap	ldap.vendorName	string	true
ldap	ldap.vendorVersion	string	true
llmnr	llmnr.ipv4	string	true
llmnr	llmnr.ipv6	string	true
llmnr	llmnr.name	string	true
lockdownd	lockdownd.cpuArchitecture	string	true
lockdownd	lockdownd.deviceName	string	true
lockdownd	lockdownd.hardwareModel	string	true
lockdownd	lockdownd.productName	string	true
lockdownd	lockdownd.productType	string	true
lockdownd	lockdownd.productVersion	string	true
lockdownd	lockdownd.supportedDeviceFamilies	string	true
lpd	lpd.banner	string	true
lwm2m	lwm2m.bootstrapServer	string	true
lwm2m	lwm2m.payload	string	true
lwm2m	lwm2m.registrationDirectory	string	true
lwm2m	lwm2m.serverImpl	string	true
m2pa	m2pa.errorCode	string	true
m2pa	m2pa.infoString	string	true

m2pa	m2pa.linkState	string	true
m2pa	m2pa.linkStateName	string	true
m2pa	m2pa.messageClass	string	true
m2pa	m2pa.messageType	string	true
m2ua	m2ua.errorCode	string	true
m2ua	m2ua.infoString	string	true
m2ua	m2ua.messageClass	string	true
m2ua	m2ua.messageType	string	true
m3ua	m3ua.errorCode	string	true
m3ua	m3ua.infoString	string	true
m3ua	m3ua.messageClass	string	true
m3ua	m3ua.messageType	string	true
managesieve	managesieve.banner	string	true
managesieve	managesieve.extensions	string	true
managesieve	managesieve.implementation	string	true
managesieve	managesieve.language	string	true
managesieve	managesieve.sasl	string	true
managesieve	managesieve.startTLS	string	true
managesieve	managesieve.version	string	true
mdns	mdns.addrs	string	false
mdns	mdns.device.model	string	true
mdns	mdns.device.osxvers	numeric	true
mdns	mdns.hostname	string	false
mdns	mdns.knowsAbout	string	false
mdns	mdns.ports	string	false
mdns	mdns.replies	string	false
mdns	mdns.services	string	false
megaco	megaco.banner	string	true
megaco	megaco.error	string	true
megaco	megaco.errorCode	string	true
megaco	megaco.format	string	true

megaco	megaco.mid	string	true
megaco	megaco.port	numeric	true
megaco	megaco.profile	string	true
megaco	megaco.replyID	string	true
megaco	megaco.serviceChange	string	true
megaco	megaco.target	string	true
megaco	megaco.transactionID	string	true
megaco	megaco.version	string	true
melsecq	melsecq.cpe	string	true
melsecq	melsecq.cpuModel	string	true
memcache	memcache.binaryProtocol	string	true
memcache	memcache.bytes	string	true
memcache	memcache.currConnections	numeric	true
memcache	memcache.currItems	string	true
memcache	memcache.error	string	true
memcache	memcache.libevent	string	true
memcache	memcache.limitMaxbytes	string	true
memcache	memcache.maxConnections	numeric	true
memcache	memcache.pid	string	true
memcache	memcache.pointerSize	string	true
memcache	memcache.threads	string	true
memcache	memcache.totalConnections	string	true
memcache	memcache.totalItems	string	true
memcache	memcache.uptime	numeric	true
memcache	memcache.version	string	true
mgcp	mgcp.endpoint	string	true
mgcp	mgcp.endpoints	numeric	true
mgcp	mgcp.localOptions	string	true
mgcp	mgcp.mode	string	true
mgcp	mgcp.packages	string	true
mgcp	mgcp.responseCode	string	true

mgcp	mgcp.signals	string	true
mikrotikwinbox	mikrotikwinbox.banner	string	true
mikrotikwinbox	mikrotikwinbox.subProtocol	string	true
minecraft	minecraft.currentplayers	numeric	true
minecraft	minecraft.maxplayers	numeric	true
minecraft	minecraft.motd	string	true
minecraft	minecraft.version	string	true
modbus	modbus.applicationName	string	true
modbus	modbus.deviceCount	numeric	true
modbus	modbus.exception	string	true
modbus	modbus.function	string	true
modbus	modbus.gateway	string	true
modbus	modbus.gatewayPort	numeric	true
modbus	modbus.identifier	string	true
modbus	modbus.modelName	string	true
modbus	modbus.port	numeric	true
modbus	modbus.productCode	string	true
modbus	modbus.productName	string	true
modbus	modbus.registers	string	true
modbus	modbus.remote	string	true
modbus	modbus.revision	string	true
modbus	modbus.subDeviceCount	numeric	true
modbus	modbus.target	string	true
modbus	modbus.unitID	string	true
modbus	modbus.vendor	string	true
modbus	modbus.vendorURL	string	true
mongodb	mongodb.allocator	string	true
mongodb	mongodb.auth	string	true
mongodb	mongodb.bits	numeric	true
mongodb	mongodb.buildEnvironment.cc	string	true
mongodb	mongodb.buildEnvironment.ccflags	string	true

mongodb	mongodb.buildEnvironment.cxx	string	true
mongodb	mongodb.buildEnvironment.cxxflags	string	true
mongodb	mongodb.buildEnvironment.distarch	string	true
mongodb	mongodb.buildEnvironment.distmod	string	true
mongodb	mongodb.buildEnvironment.linkflags	string	true
mongodb	mongodb.buildEnvironment.target_arch	string	true
mongodb	mongodb.buildEnvironment.target_os	string	true
mongodb	mongodb.cpuArch	string	true
mongodb	mongodb.debug	bool	true
mongodb	mongodb.error	string	true
mongodb	mongodb.gitVersion	string	true
mongodb	mongodb.hostname	string	true
mongodb	mongodb.isMaster	string	true
mongodb	mongodb.javascriptEngine	string	true
mongodb	mongodb.jsEngine	string	true
mongodb	mongodb.legacyWireVersion	string	true
mongodb	mongodb.maxBsonObjectSize	numeric	true
mongodb	mongodb.maxWireVersion	string	true
mongodb	mongodb.memSizeMB	string	true
mongodb	mongodb.minWireVersion	string	true
mongodb	mongodb.modules	string	false
mongodb	mongodb.mongos	string	true
mongodb	mongodb.numCores	string	true
mongodb	mongodb.ok	string	true
mongodb	mongodb.opcode	string	true
mongodb	mongodb.openssl.compiled	string	true
mongodb	mongodb.openssl.running	string	true
mongodb	mongodb.os	string	true
mongodb	mongodb.osVersion	string	true
mongodb	mongodb.readOnly	string	true
mongodb	mongodb.setParameter.md5	string	true

mongodb	mongodb.storageEngines	string	false
mongodb	mongodb.sysInfo	string	true
mongodb	mongodb.version	string	true
mongodb	mongodb.versionArray	string	false
moxa	moxa.ip	string	true
moxa	moxa.mac	string	true
moxa	moxa.model	string	true
moxa	moxa.productLine	string	true
moxa	moxa.vendor	string	true
mqtt	mqtt.broker.authRequired	bool	true
mqtt	mqtt.broker.supportedProtocols	numeric	true
mqtt	mqtt.broker.sys.buildTime	string	true
mqtt	mqtt.broker.sys.clients.connected	string	true
mqtt	mqtt.broker.sys.clients.maximum	string	true
mqtt	mqtt.broker.sys.clients.total	string	true
mqtt	mqtt.broker.sys.messages.retained.count	numeric	true
mqtt	mqtt.broker.sys.messages.stored	string	true
mqtt	mqtt.broker.sys.subscriptions.count	numeric	true
mqtt	mqtt.broker.sys.time	string	true
mqtt	mqtt.broker.sys.uptime	numeric	true
mqtt	mqtt.broker.sys.version	string	true
mqtt	mqtt.connAckCode	string	true
mqtt	mqtt.connAckMessage	string	true
mqtt	mqtt.isV5	string	true
mqtt	mqtt.port	numeric	true
mqtt	mqtt.protocolVersion	string	true
mqtt	mqtt.sessionPresent	string	true
mqtt	mqtt.target	string	true
mssql	mssql.browser.response	string	false
mssql	mssql.encryption	string	true
mssql	mssql.encryptionCode	string	true

mssql	mssql.instanceName	string	true
mssql	mssql.instanceNames	string	true
mssql	mssql.isClustered	string	true
mssql	mssql.mars	string	true
mssql	mssql.nps	string	true
mssql	mssql.ntlm.dnsComputer	string	true
mssql	mssql.ntlm.dnsDomain	string	true
mssql	mssql.ntlm.dnsTree	string	true
mssql	mssql.ntlm.negotiateFlags	string	true
mssql	mssql.ntlm.netbiosComputer	string	true
mssql	mssql.ntlm.netbiosDomain	string	true
mssql	mssql.ntlm.osBuild	string	true
mssql	mssql.ntlm.osVersion	string	true
mssql	mssql.ntlm.targetName	string	true
mssql	mssql.ntlm.timestamp	numeric	true
mssql	mssql.packetType	string	true
mssql	mssql.packetTypeCode	string	true
mssql	mssql.port	numeric	true
mssql	mssql.product	string	true
mssql	mssql.requiresEncryption	string	true
mssql	mssql.requiresTDS8Strict	string	true
mssql	mssql.serverName	string	true
mssql	mssql.serverNames	string	true
mssql	mssql.sessionEncryption	string	true
mssql	mssql.target	string	true
mssql	mssql.tcp	string	true
mssql	mssql.tcp.ports	numeric	false
mssql	mssql.version	string	true
mssql	mssql.versionBuild	string	true
mssql	mssql.versionMajor	string	true
mssql	mssql.versionMinor	string	true

mssql	mssql.versionName	string	true
mssql	mssql.versionSubBuild	string	true
mssql	mssql.versions	string	true
munin	munin.capabilities	string	false
munin	munin.hostname	string	true
munin	munin.metrics	string	false
munin	munin.nodes	string	false
munin	munin.rawBanner	string	true
munin	munin.tlsRequired	bool	true
munin	munin.version	string	true
mysql	mysql.authPluginDataLength	string	true
mysql	mysql.authPluginName	string	true
mysql	mysql.capabilities	string	true
mysql	mysql.capabilityFlags	string	true
mysql	mysql.characterSet	string	true
mysql	mysql.characterSetName	string	true
mysql	mysql.connectionID	string	true
mysql	mysql.error	string	true
mysql	mysql.errorCode	string	true
mysql	mysql.errorMessage	string	true
mysql	mysql.errorSQLState	string	true
mysql	mysql.family	string	true
mysql	mysql.normalizedVersion	string	true
mysql	mysql.packetType	string	true
mysql	mysql.product	string	true
mysql	mysql.protocolVersion	string	true
mysql	mysql.ssl	string	true
mysql	mysql.sslSupported	numeric	true
mysql	mysql.status	string	true
mysql	mysql.statusFlags	string	true
mysql	mysql.version	string	true

mysqlx	mysqlx.authentication.mechanisms	string	true
mysqlx	mysqlx.client.interactive	string	true
mysqlx	mysqlx.client.pwdExpireOk	string	true
mysqlx	mysqlx.compression.algorithm	string	true
mysqlx	mysqlx.doc.formats	numeric	true
mysqlx	mysqlx.error	string	true
mysqlx	mysqlx.errorCode	string	true
mysqlx	mysqlx.errorSeverity	string	true
mysqlx	mysqlx.messageType	string	true
mysqlx	mysqlx.nodeType	string	true
mysqlx	mysqlx.tls	string	true
natpmp	natpmp.externalIP	string	true
natpmp	natpmp.lastChange	string	true
natpmp	natpmp.responseCode	string	true
nats	nats.authRequired	bool	true
nats	nats.banner	string	true
nats	nats.clientID	string	true
nats	nats.clientIP	string	true
nats	nats.cluster	string	true
nats	nats.connectURLs	string	true
nats	nats.domain	string	true
nats	nats.gitCommit	string	true
nats	nats.goVersion	string	true
nats	nats.headers	string	true
nats	nats.host	string	true
nats	nats.info	string	true
nats	nats.jetStream	string	true
nats	nats.ldm	string	true
nats	nats.maxPayload	string	true
nats	nats.nonce	string	true
nats	nats.ok	string	true

nats	nats.pong	string	true
nats	nats.port	numeric	true
nats	nats.proto	string	true
nats	nats.serverID	string	true
nats	nats.serverName	string	true
nats	nats.tlsAvailable	string	true
nats	nats.tlsRequired	bool	true
nats	nats.tlsVerify	string	true
nats	nats.version	string	true
ndmp	ndmp.reason	string	true
ndmp	ndmp.status	string	true
ndmp	ndmp.statusName	string	true
ndmp	ndmp.timestampTS	numeric	true
ndmp	ndmp.version	string	true
neo4j	neo4j.boltVersion	string	true
neo4j	neo4j.neo4jVersion	string	true
neo4j	neo4j.server	string	true
neo4j	neo4j.serverVersion	string	true
netbios	netbios.addrs	string	true
netbios	netbios.domain	string	true
netbios	netbios.domainController	string	true
netbios	netbios.mac	string	true
netbios	netbios.macDateAdded	string	true
netbios	netbios.macVendor	string	true
netbios	netbios.name	string	true
netbios	netbios.names	string	true
netbios	netbios.nodeType	string	true
netbios	netbios.primarydomainController	string	true
netbios	netbios.responseType	string	true
netbios	netbios.username	string	true
nfs	nfs.allowed	string	false

nfs	nfs.detected	bool	true
nfs	nfs.exports	string	true
nfs	nfs.maxVersion	string	true
nfs	nfs.minVersion	string	true
nfs	nfs.unknownReply	string	true
nfs	nfs.verifierFlavor	string	true
nfs	nfs.versions	string	true
nfs	nfs.xid	string	true
nrpe	nrpe.commandArgsEnabled	bool	true
nrpe	nrpe.output	string	true
nrpe	nrpe.resultCode	string	true
nrpe	nrpe.version	string	true
ntp	ntp.controlAssociationID	string	true
ntp	ntp.controlOpcode	string	true
ntp	ntp.controlSequence	string	true
ntp	ntp.controlStatus	string	true
ntp	ntp.controlVersion	string	true
ntp	ntp.interval	string	true
ntp	ntp.leapIndicator	string	true
ntp	ntp.leapIndicatorCode	string	true
ntp	ntp.mode	string	true
ntp	ntp.modeCode	string	true
ntp	ntp.monlistItemSize	string	true
ntp	ntp.monlistNumItems	string	true
ntp	ntp.monlistSupported	numeric	true
ntp	ntp.originTimestamp	numeric	true
ntp	ntp.poll	string	true
ntp	ntp.pollSeconds	string	true
ntp	ntp.precision	string	true
ntp	ntp.precisionSeconds	string	true
ntp	ntp.processor	string	true

ntp	ntp.readVar	string	true
ntp	ntp.receiveTimestamp	numeric	true
ntp	ntp.referenceID	string	true
ntp	ntp.referenceIDDescription	string	true
ntp	ntp.referenceTimestamp	numeric	true
ntp	ntp.rootDelay	string	true
ntp	ntp.rootDispersion	string	true
ntp	ntp.serverTime	string	true
ntp	ntp.skew	string	true
ntp	ntp.skewMS	string	true
ntp	ntp.stratum	string	true
ntp	ntp.stratumDescription	string	true
ntp	ntp.swVersion	string	true
ntp	ntp.system	string	true
ntp	ntp.timestamp	string	true
ntp	ntp.transmitTimestamp	numeric	true
ntp	ntp.version	string	true
omronfins	omronfins.banner	string	true
omronfins	omronfins.model	string	true
omronfins	omronfins.transport	numeric	true
omronfins	omronfins.version	string	true
opcua	opcua.applications	string	false
opcua	opcua.channelCertSHA256	string	true
opcua	opcua.clientThumbprint	string	true
opcua	opcua.endpointCerts	numeric	true
opcua	opcua.endpointCertsSuppressed	string	true
opcua	opcua.endpointCount	numeric	true
opcua	opcua.endpointUniqueCerts	numeric	true
opcua	opcua.endpoints	numeric	true
opcua	opcua.error	string	true
opcua	opcua.errorReason	string	true

opcua	opcua.errorText	string	true
opcua	opcua.maxChunkCount	numeric	true
opcua	opcua.maxMessageSize	string	true
opcua	opcua.messageType	string	true
opcua	opcua.port	numeric	true
opcua	opcua.protocol	string	true
opcua	opcua.receiveBufferSize	string	true
opcua	opcua.revisedLifetime	string	true
opcua	opcua.secureChannelID	string	true
opcua	opcua.securityPolicy	string	true
opcua	opcua.sendBufferSize	string	true
opcua	opcua.supportedMessageSecurityModes	numeric	true
opcua	opcua.supportedSecurityPolicies	numeric	true
opcua	opcua.supportedUserTokenTypes	numeric	true
opcua	opcua.target	string	true
opcua	opcua.tokenID	string	true
opcua	opcua.unknownNodeId	string	true
openvpn	openvpn.detected	bool	true
openvpn	openvpn.keyID	string	true
openvpn	openvpn.messagePacketID	string	true
openvpn	openvpn.opcode	string	true
openvpn	openvpn.packetID	string	true
openvpn	openvpn.packetIDArrayLength	string	true
openvpn	openvpn.remoteSessionID	string	true
openvpn	openvpn.reply	string	true
openvpn	openvpn.service	string	true
openvpn	openvpn.sessionID	string	true
oracle	oracle.connectFlags0	string	true
oracle	oracle.connectFlags1	string	true
oracle	oracle.didResend	string	true
oracle	oracle.globalServiceOptions	string	true

oracle	oracle.nsn.Authentication	string	true
oracle	oracle.nsn.Encryption	string	true
oracle	oracle.nsnVersion	string	true
oracle	oracle.packetType	string	true
oracle	oracle.packetTypeCode	string	true
oracle	oracle.redirectData	string	true
oracle	oracle.redirectDescriptor	string	true
oracle	oracle.refuseData	string	true
oracle	oracle.refuseDescriptor	string	true
oracle	oracle.refuseReason	string	true
oracle	oracle.tnsVersion	string	true
oracle	oracle.version	string	true
oracledb	oracledb.errCode	string	true
oracledb	oracledb.packetType	string	true
oracledb	oracledb.tns.error	string	true
oracledb	oracledb.tns.version	string	true
oracledb	oracledb.tns.vsn	string	true
oracledb	oracledb.version	string	true
oracledb	oracledb.vsnnum	string	true
orion	orion.components	string	true
orion	orion.version	string	true
panxmlapi	host.mac	string	false
panxmlapi	host.name	string	false
panxmlapi	pan.api.advancedRouting	string	true
panxmlapi	pan.api.appVersion	string	true
panxmlapi	pan.api.arpcache.ports	string	true
panxmlapi	pan.api.avDatabaseVersion	string	true
panxmlapi	pan.api.deviceCertificateStatus	string	true
panxmlapi	pan.api.globalProtectVersion	string	true
panxmlapi	pan.api.logdbVersion	string	true
panxmlapi	pan.api.model	string	true

panxmlapi	pan.api.multiVsys	string	true
panxmlapi	pan.api.operationalMode	string	true
panxmlapi	pan.api.osVersion	string	true
panxmlapi	pan.api.plugins	string	true
panxmlapi	pan.api.serialNumber	string	true
panxmlapi	pan.api.threatDatabaseVersion	string	true
panxmlapi	pan.api.urlDatabaseVersion	string	true
panxmlapi	pan.api.vmMode	string	true
panxmlapi	pan.api.vpnDisableMode	string	true
panxmlapi	pan.api.wildfireDatabaseVersion	string	true
panxmlapi	pan.api.wildfireRt	string	true
panxmlapi	snmp.arpcache	string	false
pca	pca.caps	string	true
pca	pca.isRequest	string	true
pca	pca.name	string	true
pca	pca.status	string	true
pcworx	pcworx.firmwareDate	string	true
pcworx	pcworx.firmwareTime	string	true
pcworx	pcworx.firmwareVersion	string	true
pcworx	pcworx.modelNumber	string	true
pcworx	pcworx.plcType	string	true
pega	pega.version	string	true
pfcp	pfcp.nodeID	string	true
pfcp	pfcp.nodeIDType	string	true
pfcp	pfcp.recoveryTimestamp	numeric	true
pfcp	pfcp.seidPresent	string	true
pfcp	pfcp.sequenceNumber	string	true
pfcp	pfcp.version	string	true
pop3	pop3.apop	string	true
pop3	pop3.apopBanner	string	true
pop3	pop3.authMethods	string	true

pop3	pop3.banner	string	true
pop3	pop3.capa.message	string	true
pop3	pop3.capa.status	string	true
pop3	pop3.capabilities	string	true
pop3	pop3.capabilityCount	numeric	true
pop3	pop3.expire	string	true
pop3	pop3.implementation	string	true
pop3	pop3.loginDelay	string	true
pop3	pop3.message	string	true
pop3	pop3.pipelining	string	true
pop3	pop3.responseCodes	string	true
pop3	pop3.startTLS	string	true
pop3	pop3.status	string	true
pop3	pop3.top	string	true
pop3	pop3.uidl	string	true
pop3	pop3.userAuth	string	true
postgresql	postgresql.auth.details	string	true
postgresql	postgresql.auth.method	string	true
postgresql	postgresql.backendPID	string	true
postgresql	postgresql.clientEncoding	string	true
postgresql	postgresql.dateStyle	string	true
postgresql	postgresql.error.code	numeric	true
postgresql	postgresql.error.file	string	true
postgresql	postgresql.error.line	string	true
postgresql	postgresql.error.message	string	true
postgresql	postgresql.error.routine	string	true
postgresql	postgresql.error.severity	string	true
postgresql	postgresql.error.text	string	true
postgresql	postgresql.error.unknown	string	true
postgresql	postgresql.integerDatetimes	string	true
postgresql	postgresql.isSuperuser	string	true

postgresql	postgresql.notice.message	string	true
postgresql	postgresql.notice.severity	string	true
postgresql	postgresql.packetType	string	true
postgresql	postgresql.routine	string	true
postgresql	postgresql.serverEncoding	string	true
postgresql	postgresql.serverVersion	string	true
postgresql	postgresql.sessionAuthorization	string	true
postgresql	postgresql.sslSupported	string	true
postgresql	postgresql.standardConformingStrings	string	true
postgresql	postgresql.timeZone	string	true
postgresql	postgresql.txStatus	string	true
pptp	pptp.bearerCapabilities	string	true
pptp	pptp.errorCode	string	true
pptp	pptp.errorText	string	true
pptp	pptp.firmwareRevision	string	true
pptp	pptp.framingCapabilities	string	true
pptp	pptp.fwRevision	string	true
pptp	pptp.hostname	string	true
pptp	pptp.maxChannels	string	true
pptp	pptp.port	numeric	true
pptp	pptp.protocolVersion	string	true
pptp	pptp.protocolVersionRaw	string	true
pptp	pptp.resultCode	string	true
pptp	pptp.resultText	string	true
pptp	pptp.target	string	true
pptp	pptp.vendor	string	true
proconos	proconos.bootProject	string	true
proconos	proconos.ladderLogicRuntime	string	true
proconos	proconos.plcType	string	true
proconos	proconos.projectName	string	true
proconos	proconos.projectSourceCode	string	true

proconos	prosoft.moduleName	string	true
proconos	prosoft.operatingSystemRevision	string	true
proconos	prosoft.softwareRevision	string	true
profinet	profinet.annotation	string	true
profinet	profinet.cpe	string	true
profinet	profinet.deviceName	string	true
profinet	profinet.deviceType	string	true
profinet	profinet.vendor	string	true
prosoft	firmwareDate	string	true
prosoft	moduleName	string	true
prosoft	moduleRevision	string	true
prosoft	moduleSerial	string	true
prosoft	moduleStatus	string	true
psdisco	psdisco.appName	string	true
psdisco	psdisco.appTitleID	string	true
psdisco	psdisco.code	string	true
psdisco	psdisco.id	string	true
psdisco	psdisco.name	string	true
psdisco	psdisco.protoVersion	string	true
psdisco	psdisco.requestPort	string	true
psdisco	psdisco.status	string	true
psdisco	psdisco.sysVersion	string	true
psdisco	psdisco.type	string	true
qotd	qotd.quote	string	true
qualys	qualys.correlationID	string	true
radius	radius.attributes	string	true
radius	radius.calledStationId	string	true
radius	radius.code	numeric	true
radius	radius.codeID	string	true
radius	radius.identifier	string	true
radius	radius.nasIdentifier	string	true

radius	radius.replyMessage	string	true
rdp	rdp.auth.nla	string	true
rdp	rdp.auth.sspeua	string	true
rdp	rdp.auth.supportsNegotiation	numeric	true
rdp	rdp.auth.tls	string	true
rdp	rdp.failureCode	string	true
rdp	rdp.failureCodeName	string	true
rdp	rdp.features	string	true
rdp	rdp.flags	string	true
rdp	rdp.negotiation	string	true
rdp	rdp.ntlm.dnsComputer	string	true
rdp	rdp.ntlm.dnsDomain	string	true
rdp	rdp.ntlm.dnsTree	string	true
rdp	rdp.ntlm.netbiosComputer	string	true
rdp	rdp.ntlm.netbiosDomain	string	true
rdp	rdp.ntlm.osVersion	string	true
rdp	rdp.ntlm.targetName	string	true
rdp	rdp.securityLevel	string	true
rdp	rdp.selectedProtocol	string	true
rdp	rdp.selectedProtocolName	string	true
rdp	rdp.tls.fingerprintSHA256	string	true
rdp	rdp.tls.issuer	string	true
rdp	rdp.tls.notAfter	string	true
rdp	rdp.tls.notBefore	string	true
rdp	rdp.tls.subject	string	true
redfish	redfish.name	string	true
redfish	redfish.product	string	true
redfish	redfish.uuid	string	true
redfish	redfish.vendor	string	true
redfish	redfish.version	string	true
redis	redis.archBits	numeric	true

redis	redis.authRequired	bool	true
redis	redis.buildID	string	true
redis	redis.clusterEnabled	bool	true
redis	redis.connectedClients	numeric	true
redis	redis.error	string	true
redis	redis.gccVersion	string	true
redis	redis.gitSHA1	string	true
redis	redis.maxMemory	string	true
redis	redis.memAllocator	string	true
redis	redis.mode	string	true
redis	redis.os	string	true
redis	redis.ping	string	true
redis	redis.port	numeric	true
redis	redis.protectedMode	string	true
redis	redis.redisVersion	string	true
redis	redis.role	string	true
redis	redis.runID	string	true
redis	redis.totalCommands	string	true
redis	redis.totalConnections	string	true
redis	redis.uptime	numeric	true
redis	redis.usedMemory	string	true
redis	redis.valkeyVersion	string	true
redis	redis.version	string	true
riak	riak.nodename	string	true
riak	riak.version	string	true
riak-http	riak-http.version	string	true
rip	rip.auth	string	true
rip	rip.nextHop	string	false
rip	rip.route	string	false
rip	rip.routeCidr	string	false
rip	rip.routeCount	string	true

rip	rip.routingDomain	string	true
rip	rip.version	string	true
roomalert	roomalert.ipAddress	string	false
roomalert	roomalert.macAddress	string	false
roomalert	roomalert.model	string	false
roomalert	roomalert.osVersion	string	false
roomalert	roomalert.serialNumber	string	true
rpcbind	rpcbind.acceptState	string	true
rpcbind	rpcbind.acceptStatus	string	true
rpcbind	rpcbind.addrs	string	true
rpcbind	rpcbind.authStatus	string	true
rpcbind	rpcbind.error	string	true
rpcbind	rpcbind.mappingCount	numeric	true
rpcbind	rpcbind.mountd	string	true
rpcbind	rpcbind.nfs	string	true
rpcbind	rpcbind.nlockmgr	string	true
rpcbind	rpcbind.programEntries	string	true
rpcbind	rpcbind.programs	string	true
rpcbind	rpcbind.rejectStatus	string	true
rpcbind	rpcbind.replyStatus	string	true
rpcbind	rpcbind.transport	numeric	true
rpcbind	rpcbind.versionRange	string	true
rsync	rsync.banner	string	true
rsync	rsync.modules	string	true
rsync	rsync.service	string	true
rsync	rsync.version	string	true
rtmp	rtmp.version	string	true
rtsp	rtsp.banner	string	true
rtsp	rtsp.contentType	string	true
rtsp	rtsp.cseq	string	true
rtsp	rtsp.head.*	string	true

rtsp	rtsp.methodCount	numeric	true
rtsp	rtsp.methods	string	true
rtsp	rtsp.port	numeric	true
rtsp	rtsp.public	string	true
rtsp	rtsp.reasonPhrase	string	true
rtsp	rtsp.request.cseq	string	true
rtsp	rtsp.request.method	string	true
rtsp	rtsp.request.uri	string	true
rtsp	rtsp.server	string	true
rtsp	rtsp.session	string	true
rtsp	rtsp.statusClass	string	true
rtsp	rtsp.statusCode	string	true
rtsp	rtsp.statusName	string	true
rtsp	rtsp.target	string	true
rtsp	rtsp.transport	numeric	true
rtsp	rtsp.version	string	true
rtsp	rtsp.wwwAuthenticate	string	true
s7comm	s7comm.componentCount	numeric	true
s7comm	s7comm.components	string	true
s7comm	s7comm.copyright	string	true
s7comm	s7comm.cpuType	string	true
s7comm	s7comm.firmwareExtension	string	true
s7comm	s7comm.firmwareExtensionVersion	string	true
s7comm	s7comm.firmwareName	string	true
s7comm	s7comm.firmwareVersion	string	true
s7comm	s7comm.hardwareName	string	true
s7comm	s7comm.hardwareVersion	string	true
s7comm	s7comm.locationDesignation	string	true
s7comm	s7comm.memoryCardSerial	string	true
s7comm	s7comm.moduleName	string	true
s7comm	s7comm.moduleType	string	true

s7comm	s7comm.moduleTypeName	string	true
s7comm	s7comm.moduleVersion	string	true
s7comm	s7comm.plantDesignation	string	true
s7comm	s7comm.protocolVersion	string	true
s7comm	s7comm.serialNumber	string	true
s7comm	s7comm.systemName	string	true
sadp	sadp.activated	string	true
sadp	sadp.analogChannelNum	string	true
sadp	sadp.bootTime	string	true
sadp	sadp.cmdPort	string	true
sadp	sadp.deviceDesc	string	true
sadp	sadp.deviceSerial	string	true
sadp	sadp.deviceType	string	true
sadp	sadp.dhcp	string	true
sadp	sadp.digitalChannelNum	string	true
sadp	sadp.dspVersion	string	true
sadp	sadp.httpPort	string	true
sadp	sadp.ipv4.address	string	true
sadp	sadp.ipv4.gateway	string	true
sadp	sadp.ipv4.subnet	string	true
sadp	sadp.ipv6.address	string	true
sadp	sadp.ipv6.gateway	string	true
sadp	sadp.ipv6.maskLen	string	true
sadp	sadp.mac	string	true
sadp	sadp.oemInfo	string	true
sadp	sadp.passwordResetAbility	string	true
sadp	sadp.softwareVersion	string	true
sadp	sadp.uuid	string	true
sccp	sccp.deviceType	string	true
sccp	sccp.protocolVersion	string	true
sccp	sccp.rejected	string	true

securemote	securemote.hostname	string	true
securemote	securemote.server	string	true
servicetag	serviceTag.agentURN	string	true
servicetag	serviceTag.agentVersion	string	true
servicetag	serviceTag.registryVersion	string	true
servicetag	serviceTag.servicePort	string	true
servicetag	serviceTag.sysinfo.cpuInfo.name	string	true
servicetag	serviceTag.sysinfo.cpuMfg	string	true
servicetag	serviceTag.sysinfo.hostID	string	true
servicetag	serviceTag.sysinfo.hostname	string	true
servicetag	serviceTag.sysinfo.mfg	string	true
servicetag	serviceTag.sysinfo.platform	string	true
servicetag	serviceTag.sysinfo.release	string	true
servicetag	serviceTag.sysinfo.serialNumber	string	true
servicetag	serviceTag.sysinfo.system	string	true
sgsap	sgsap.messageType	string	true
sgsap	sgsap.sgsCause	string	true
sip	sip.accept	string	true
sip	sip.allow	string	true
sip	sip.contact	string	true
sip	sip.head.*	string	true
sip	sip.local	string	true
sip	sip.port	numeric	true
sip	sip.reasonPhrase	string	true
sip	sip.server	string	true
sip	sip.statusCode	string	true
sip	sip.supported	numeric	true
sip	sip.target	string	true
sip	sip.userAgent	string	true
sip	sip.via	string	true
sip	sip.wwwAuthenticate	string	true

sixnet-udr	redlion.authRequired	string	true
sixnet-udr	redlion.crc	string	true
sixnet-udr	redlion.exposure	string	true
sixnet-udr	redlion.model	string	true
sixnet-udr	redlion.sync	string	true
slp	slp.attributes	string	true
slp	slp.da.attributes	string	true
slp	slp.da.bootTimestamp	numeric	true
slp	slp.da.scopes	string	true
slp	slp.da.spi	string	true
slp	slp.da.url	string	true
slp	slp.error	string	true
slp	slp.errorCode	string	true
slp	slp.flags	string	true
slp	slp.function	string	true
slp	slp.functionID	string	true
slp	slp.language	string	true
slp	slp.length	numeric	true
slp	slp.nextExtensionOffset	string	true
slp	slp.port	numeric	true
slp	slp.sa.attributes	string	true
slp	slp.sa.scopes	string	true
slp	slp.sa.url	string	true
slp	slp.serviceAgentURL	string	true
slp	slp.serviceTypeCount	numeric	true
slp	slp.serviceTypes	string	true
slp	slp.serviceTypesRaw	string	true
slp	slp.target	string	true
slp	slp.urlCount	numeric	true
slp	slp.urls	string	true
slp	slp.version	string	true

slp	slp.xid	string	true
smb	ntlmssp.authTargetDomain	bool	true
smb	ntlmssp.authTargetServer	bool	true
smb	ntlmssp.dnsComputer	string	true
smb	ntlmssp.dnsDomain	string	true
smb	ntlmssp.dnsTree	string	true
smb	ntlmssp.negFlags	string	true
smb	ntlmssp.netbiosComputer	string	true
smb	ntlmssp.netbiosDomain	string	true
smb	ntlmssp.ntlmRevision	numeric	true
smb	ntlmssp.serverChallenge	string	false
smb	ntlmssp.targetName	string	true
smb	ntlmssp.timestamp	string	true
smb	ntlmssp.timestampTS	numeric	true
smb	ntlmssp.version	string	true
smb	ntlmssp.versionInvalid	bool	true
smb	smb.accessControl	string	true
smb	smb.bootTime	string	true
smb	smb.bootTimeTS	string	true
smb	smb.capabilities	string	true
smb	smb.cipherAlg	string	false
smb	smb.cipherAlgCnt	numeric	true
smb	smb.compAlg	string	false
smb	smb.compAlgCnt	numeric	true
smb	smb.compFlags	string	true
smb	smb.dialect	string	true
smb	smb.dialectIndex	string	true
smb	smb.dialectName	string	true
smb	smb.dialectRevision	string	true
smb	smb.dnsComputer	string	true
smb	smb.dnsDomain	string	true

smb	smb.dnsTree	string	true
smb	smb.guid	string	true
smb	smb.hashAlg	string	false
smb	smb.hashAlgCnt	numeric	true
smb	smb.hashSaltLen	string	true
smb	smb.maxReadSize	string	true
smb	smb.maxTransactSize	string	true
smb	smb.maxWriteSize	string	true
smb	smb.nativeLM	string	true
smb	smb.nativeOS	string	true
smb	smb.negCtxCnt	numeric	true
smb	smb.netName	string	true
smb	smb.netbiosComputer	string	true
smb	smb.netbiosDomain	string	true
smb	smb.ntStatus	string	true
smb	smb.ntlmssp	string	true
smb	smb.ntlmsspNegFlags	string	true
smb	smb.ntlmsspVersion	string	true
smb	smb.posixExtensions	string	true
smb	smb.rdmaTransformCnt	numeric	true
smb	smb.rdmaTransforms	string	false
smb	smb.securityMode	string	true
smb	smb.serverGUID	string	true
smb	smb.sessionID	string	true
smb	smb.signing	string	true
smb	smb.signingAlg	string	false
smb	smb.signingAlgCnt	numeric	true
smb	smb.signingEnabled	bool	true
smb	smb.signingRequired	bool	true
smb	smb.supportsEncryptedPasswords	numeric	true
smb	smb.timestamp	numeric	true

smb	smb.timestampTS	numeric	true
smb	smb.transportSecurity	numeric	true
smb	smb.version	string	true
smb3	ntlmssp.versionInvalid	string	true
smb3	smb.accessControl	string	true
smb3	smb.capabilities	string	true
smb3	smb.cipherAlg	string	false
smb3	smb.cipherAlgCnt	string	true
smb3	smb.compAlg	string	false
smb3	smb.compAlgCnt	string	true
smb3	smb.compFlags	numeric	true
smb3	smb.dialect	string	true
smb3	smb.guid	string	true
smb3	smb.hashAlg	string	false
smb3	smb.hashAlgCnt	string	true
smb3	smb.hashSaltLen	string	true
smb3	smb.nativeLM	string	true
smb3	smb.nativeOS	string	true
smb3	smb.netName	string	true
smb3	smb.netbiosComputer	string	true
smb3	smb.netbiosDomain	string	true
smb3	smb.posixExtensions	string	true
smb3	smb.rdmaTransformCnt	string	true
smb3	smb.rdmaTransforms	string	false
smb3	smb.sessionID	string	true
smb3	smb.signing	string	true
smb3	smb.signingAlg	string	false
smb3	smb.signingAlgCnt	string	true
smb3	smb.supportsEncryptedPasswords	string	true
smb3	smb.transportSecurity	string	true
snmp	snmp._secrets	string	true

snmp	snmp.a10.softwareVersion	string	true
snmp	snmp.arpCount	numeric	true
snmp	snmp.arpcache	string	true
snmp	snmp.arpcache.oids	string	false
snmp	snmp.arpcache.ports	numeric	false
snmp	snmp.cdpCapabilities	string	true
snmp	snmp.cdpDeviceIDs	string	true
snmp	snmp.cdpIPs	string	true
snmp	snmp.cdpNeighborCount	numeric	true
snmp	snmp.cdpPlatforms	string	true
snmp	snmp.cdpPortIDs	string	true
snmp	snmp.cisco.hardware	string	true
snmp	snmp.cisco.hardwareRev	string	true
snmp	snmp.cisco.softwareVersion	string	true
snmp	snmp.community	string	true
snmp	snmp.credentials	string	true
snmp	snmp.defaultCommunities	string	true
snmp	snmp.engineBoots	numeric	true
snmp	snmp.engineID.enterpriseID	string	true
snmp	snmp.engineID.raw	string	true
snmp	snmp.engineID.vendor	string	true
snmp	snmp.engineTime	string	true
snmp	snmp.errorIndex	string	true
snmp	snmp.errorStatus	string	true
snmp	snmp.errorStatusName	string	true
snmp	snmp.extendOsReleaseFull	string	true
snmp	snmp.f5.hardware	string	true
snmp	snmp.f5.softwareBuild	string	true
snmp	snmp.f5.softwareVersion	string	true
snmp	snmp.failedAuth	string	true
snmp	snmp.fortinet.fgSysVersion	string	true

snmp	snmp.fortinet.fnSysSerial	string	true
snmp	snmp.fortinet.fnSysVersion	string	true
snmp	snmp.hostname	string	true
snmp	snmp.ifCount	numeric	true
snmp	snmp.inauthentic	string	true
snmp	snmp.installedSWErr	string	true
snmp	snmp.installedSWHWHint	string	true
snmp	snmp.installedSWOSHint	string	true
snmp	snmp.installedSWProtoVer	string	true
snmp	snmp.installedSoftware	string	true
snmp	snmp.installedSoftwareCount	numeric	true
snmp	snmp.interfaceAddrs	string	true
snmp	snmp.interfaceAddrsMap	string	true
snmp	snmp.interfaceAliases	string	true
snmp	snmp.interfaceAliasesMap	string	true
snmp	snmp.interfaceCount	numeric	true
snmp	snmp.interfaceMacs	string	true
snmp	snmp.interfaceMacsMap	string	true
snmp	snmp.interfaceNames	string	true
snmp	snmp.interfaceNamesMap	string	true
snmp	snmp.interfaceTypesMap	string	true
snmp	snmp.interfaces	string	true
snmp	snmp.ivanti.esapVersion	string	true
snmp	snmp.ivanti.productName	string	true
snmp	snmp.ivanti.productVersion	string	true
snmp	snmp.juniper.serialNumbers	string	true
snmp	snmp.juniper.softwareVersion	string	true
snmp	snmp.juniperModel	string	true
snmp	snmp.lldp.neighbor	string	false
snmp	snmp.lldpChassisIDTypes	string	true
snmp	snmp.lldpChassisIDs	string	true

snmp	snmp.lldpNeighborCount	numeric	true
snmp	snmp.lldpPortDescs	numeric	true
snmp	snmp.lldpPortIDs	numeric	true
snmp	snmp.lldpPortNums	numeric	true
snmp	snmp.lldpSysDescrs	string	true
snmp	snmp.lldpSysNames	string	true
snmp	snmp.macs.ports	numeric	true
snmp	snmp.macs.vlans	string	true
snmp	snmp.modelNames	string	true
snmp	snmp.msgMaxSize	string	true
snmp	snmp.pan.appDatabaseVersion	string	true
snmp	snmp.pan.avDatabaseVersion	string	true
snmp	snmp.pan.hardwareVersion	string	true
snmp	snmp.pan.model	string	true
snmp	snmp.pan.serialNumber	string	true
snmp	snmp.pan.softwareVersion	string	true
snmp	snmp.pan.threatDatabaseVersion	string	true
snmp	snmp.pan.urlDatabaseVersion	string	true
snmp	snmp.pan.vpnClientVersion	string	true
snmp	snmp.port	numeric	true
snmp	snmp.routeCount	numeric	true
snmp	snmp.routes	string	true
snmp	snmp.secretCommunities	string	true
snmp	snmp.serialNumbers	string	true
snmp	snmp.sonicwall.hardware	string	true
snmp	snmp.sonicwall.serialNumber	string	true
snmp	snmp.sonicwall.softwareVersion	string	true
snmp	snmp.sophos.ha.peer	string	true
snmp	snmp.sophos.ha.status	string	true
snmp	snmp.sophos.hardware	string	true
snmp	snmp.sophos.softwareVersion	string	true

snmp	snmp.sysContact	string	true
snmp	snmp.sysDesc	string	true
snmp	snmp.sysDescr	string	true
snmp	snmp.sysLocation	string	true
snmp	snmp.sysName	string	true
snmp	snmp.sysObjectID	string	true
snmp	snmp.sysServices	string	true
snmp	snmp.sysUptime	string	true
snmp	snmp.sysUptimeFormatted	string	true
snmp	snmp.target	string	true
snmp	snmp.username	string	true
snmp	snmp.usmStatsDecryptionErrors	string	true
snmp	snmp.usmStatsNotInTimeWindows	string	true
snmp	snmp.usmStatsUnknownEngineIDs	string	true
snmp	snmp.usmStatsUnknownUserNames	string	true
snmp	snmp.usmStatsUnsupportedSecLevels	numeric	true
snmp	snmp.usmStatsWrongDigests	numeric	true
snmp	snmp.varbindCount	numeric	true
snmp	snmp.version	string	true
snmp	snmp.vlanCount	numeric	true
snmp	snmp.vlans	string	true
socks4	socks4.anonymousAccess	string	true
socks4	socks4.status	string	true
socks5	socks5.authMethod	string	true
socks5	socks5.authRequired	bool	true
socks5	socks5.noAcceptable	string	true
socks5	socks5.open	string	true
socks5	socks5.socks4	string	true
socks5	socks5.version	string	true
solr	service.product	string	true
solr	service.version	string	true

solr	solr.jvm.name	string	true
solr	solr.jvm.version	string	true
solr	solr.lucene.version	string	true
solr	solr.system.arch	string	true
solr	solr.system.name	string	true
solr	solr.system.version	string	true
solr	solr.version	string	true
splunk	splunk.build	string	true
splunk	splunk.version	string	true
spotify-connect	brandDisplayName	string	true
spotify-connect	clientID	string	true
spotify-connect	deviceID	string	true
spotify-connect	deviceType	string	true
spotify-connect	modelDisplayName	string	true
spotify-connect	remoteName	string	true
spotify-connect	scope	string	true
spotify-connect	status	string	true
spotify-connect	statusString	string	true
ssh	ssh.authMethods	string	false
ssh	ssh.banner	string	true
ssh	ssh.comments	numeric	true
ssh	ssh.error	string	true
ssh	ssh.exitStatus	string	true
ssh	ssh.extensionNames	string	true
ssh	ssh.extensions	string	false
ssh	ssh.firstKexFollows	string	true
ssh	ssh.hostKey	string	true
ssh	ssh.hostKey.bkhash	string	true
ssh	ssh.hostKey.data	string	true
ssh	ssh.hostKey.md5	string	true
ssh	ssh.hostKey.sha256	string	true

ssh	ssh.hostKeyAlgs	string	true
ssh	ssh.hostKeyCount	numeric	true
ssh	ssh.hostKeyTypes	string	false
ssh	ssh.kbdInstructions	string	true
ssh	ssh.kbdName	string	true
ssh	ssh.kbdQuestion	string	true
ssh	ssh.kexAlgs	string	true
ssh	ssh.kexCookie	string	true
ssh	ssh.kexPadding	string	true
ssh	ssh.preAuthBanner	string	true
ssh	ssh.protoVersion	string	true
ssh	ssh.service	string	true
ssh	ssh.sessionMethod	string	true
ssh	ssh.sessionOutput	string	true
ssh	ssh.softwareVersion	string	true
ssh	ssh.stage	string	true
ssh	ssh.toClientCiphers	string	true
ssh	ssh.toClientCompression	string	true
ssh	ssh.toClientComps	string	true
ssh	ssh.toClientLangs	string	true
ssh	ssh.toClientMACs	string	true
ssh	ssh.toServerCiphers	string	true
ssh	ssh.toServerCompression	string	true
ssh	ssh.toServerComps	string	true
ssh	ssh.toServerLangs	string	true
ssh	ssh.toServerMACs	string	true
ssh	ssh.version	string	true
steam	steam.authKeyIDs	string	true
steam	steam.broadcastingActive	string	true
steam	steam.clientID	string	true
steam	steam.connectPort	string	true

steam	steam.contentCachePort	string	true
steam	steam.deviceID	string	true
steam	steam.downloadLANPeerGroup	string	true
steam	steam.eUniverse	string	true
steam	steam.enabledServices	string	true
steam	steam.gamesRunning	string	true
steam	steam.hostname	string	true
steam	steam.instanceID	string	true
steam	steam.ipAddresses	string	true
steam	steam.is64Bit	string	true
steam	steam.isSteamDeck	string	true
steam	steam.macAddresses	string	true
steam	steam.minVersion	string	true
steam	steam.osType	string	true
steam	steam.publicIPAddress	string	true
steam	steam.remotePlayActive	string	true
steam	steam.screenLocked	string	true
steam	steam.steamIDs	string	true
steam	steam.steamVersion	string	true
steam	steam.supportedServices	string	true
steam	steam.timestamp	string	true
steam	steam.version	string	true
steam	steam.vrActive	string	true
stun	stun.attributeCount	numeric	true
stun	stun.attributes	string	true
stun	stun.errorCode	string	true
stun	stun.errorReason	string	true
stun	stun.fingerprint	string	true
stun	stun.mappedAddress	string	true
stun	stun.mappedIP	string	true
stun	stun.mappedPort	numeric	true

stun	stun.messageClass	string	true
stun	stun.messageLength	string	true
stun	stun.messageMethod	string	true
stun	stun.otherAddress	string	true
stun	stun.responseOrigin	string	true
stun	stun.software	string	true
stun	stun.transactionID	string	true
stun	stun.type	string	true
stun	stun.xorMappedAddress	string	true
sua	sua.errorCode	string	true
sua	sua.infoString	string	true
sua	sua.messageClass	string	true
sua	sua.messageType	string	true
sunrpc	rpcbind.mappingCount	numeric	true
sunrpc	rpcbind.mountd	string	true
sunrpc	rpcbind.nfs	string	true
sunrpc	rpcbind.programs	string	true
svn	svn.authMechanisms	string	true
svn	svn.banner	string	true
svn	svn.capabilities	string	true
svn	svn.maxVersion	string	true
svn	svn.minVersion	string	true
svn	svn.repoURL	string	true
sybase	sybase.target	string	true
sybase	sybase.version	string	true
syslog	syslog.appName	string	true
syslog	syslog.facility	string	true
syslog	syslog.facilityName	string	true
syslog	syslog.hostname	string	true
syslog	syslog.priority	string	true
syslog	syslog.rfc	string	true

syslog	syslog.severity	string	true
syslog	syslog.severityName	string	true
syslog	syslog.version	string	true
teamviewer	teamviewer.command	string	true
teamviewer	teamviewer.magic	string	true
teamviewer	teamviewer.response	string	true
tftp	tftp.banner	string	true
tftp	tftp.error	string	true
tftp	tftp.message	string	true
tftp	tftp.mode	string	true
tftp	tftp.opcode	string	true
tftp	tftp.rrq.filename	string	true
tftp	tftp.wrq.filename	string	true
time	time.skewMS	string	true
time	time.timestamp	string	true
time	time.timestampTS	numeric	true
time	time.value	numeric	true
tls	tls.alert	string	true
tls	tls.alertDescription	string	true
tls	tls.alertLevel	string	true
tls	tls.alpn	string	true
tls	tls.authorityKeyID	string	true
tls	tls.caUnknown	string	true
tls	tls.certificate	string	true
tls	tls.certificateCount	numeric	true
tls	tls.certificateParseError	string	true
tls	tls.certificates	string	true
tls	tls.chainFingerprints	numeric	true
tls	tls.cipher	string	true
tls	tls.cipherName	string	true
tls	tls.cipherSuiteNames	string	true

tls	tls.cipherSuites	string	true
tls	tls.cn	string	true
tls	tls.compression	string	true
tls	tls.crl	string	false
tls	tls.emails	string	false
tls	tls.errors	string	true
tls	tls.expired	string	true
tls	tls.extensionNames	string	true
tls	tls.extensions	string	true
tls	tls.fp.bkhash	string	true
tls	tls.fp.caSha1	string	false
tls	tls.fp.sha1	string	true
tls	tls.fp.sha1Hex	string	true
tls	tls.fp.sha256	string	true
tls	tls.helloVersion	string	true
tls	tls.hostTime	string	true
tls	tls.hostTimeTS	numeric	true
tls	tls.issuer	string	true
tls	tls.issuingURL	string	false
tls	tls.names	string	false
tls	tls.negotiationError	string	true
tls	tls.notAfter	string	true
tls	tls.notAfterTS	numeric	true
tls	tls.notBefore	string	true
tls	tls.notBeforeTS	numeric	true
tls	tls.ocsp	string	false
tls	tls.ocspResponse	string	true
tls	tls.port	numeric	true
tls	tls.pqHybridKeyExchangeCodes	string	false
tls	tls.pqHybridKeyExchangeNames	string	false
tls	tls.pqc	string	false

tls	tls.publicKeyAlgorithm	string	true
tls	tls.random	string	true
tls	tls.requiresClientCertificate	string	true
tls	tls.rzfp	string	true
tls	tls.rzfp0	string	true
tls	tls.sans	string	true
tls	tls.selfSigned	string	true
tls	tls.serial	string	true
tls	tls.sessionID	string	true
tls	tls.signatureAlgorithm	string	true
tls	tls.stack	string	true
tls	tls.subject	string	true
tls	tls.subjectKeyID	string	true
tls	tls.supportedVersionCodes	string	true
tls	tls.supportedVersionNames	string	false
tls	tls.supportedVersions	string	false
tls	tls.target	string	true
tls	tls.uri	string	false
tls	tls.version	string	true
tls	tls.versionName	string	true
turn	turn.attributeCount	numeric	true
turn	turn.attributes	string	true
turn	turn.errorCode	string	true
turn	turn.errorReason	string	true
turn	turn.fingerprint	string	true
turn	turn.lifetime	string	true
turn	turn.mappedAddress	string	true
turn	turn.mappedIP	string	true
turn	turn.mappedPort	numeric	true
turn	turn.messageClass	string	true
turn	turn.messageMethod	string	true

turn	turn.nonce	string	true
turn	turn.realm	string	true
turn	turn.relayedIP	string	true
turn	turn.relayedPort	numeric	true
turn	turn.requestedTransport	numeric	true
turn	turn.software	string	true
turn	turn.type	string	true
turn	turn.xorMappedAddress	string	true
turn	turn.xorRelayedAddress	string	true
ubnt	ubnt.addrs	string	false
ubnt	ubnt.configStatus	string	true
ubnt	ubnt.directConnectDomain	string	true
ubnt	ubnt.essid	string	true
ubnt	ubnt.firmware	string	true
ubnt	ubnt.hostName	string	true
ubnt	ubnt.interfaceMap	string	false
ubnt	ubnt.macs	string	false
ubnt	ubnt.modelFull	string	true
ubnt	ubnt.modelShort	string	true
ubnt	ubnt.protoVersion	string	true
ubnt	ubnt.sourceMAC	string	false
ubnt	ubnt.unifiVersion	string	true
ubnt	ubnt.uptime	string	true
ubnt	ubnt.webMgmtPort	string	true
ubnt	ubnt.webMgmtTLS	string	true
ubnt	ubnt.wmode	string	false
unitronics	unitronics.cpe	string	true
unitronics	unitronics.hwVersion	string	true
unitronics	unitronics.model	string	true
unitronics	unitronics.osVersion	string	true
unitronics	unitronics.unitID	string	true

upnp	upnp.controlURL	string	true
upnp	upnp.deviceType	string	true
upnp	upnp.eventSubURL	string	true
upnp	upnp.friendlyName	string	true
upnp	upnp.manufacturer	string	true
upnp	upnp.manufacturerURL	string	true
upnp	upnp.modelDescription	string	true
upnp	upnp.modelName	string	true
upnp	upnp.modelNumber	string	true
upnp	upnp.modelURL	string	true
upnp	upnp.presentationURL	string	true
upnp	upnp.scpdURL	string	true
upnp	upnp.serialNumber	string	true
upnp	upnp.udn	string	true
upnp	upnp.upc	string	true
upnp	upnp.url	string	true
upnp	upnp.urlBase	string	true
upnp	upnp.wifiMac	string	true
upnp	upnp.wiredMac	string	true
uscan	uscan.makeAndModel	string	true
uscan	uscan.manufacturer	string	true
uscan	uscan.serialNumber	string	true
uscan	uscan.version	string	true
vault	vault.version	string	true
vertiv_liebert	liebert.hostname	string	true
vertiv_liebert	liebert.mac	string	true
vertiv_liebert	liebert.model	string	true
vertiv_liebert	liebert.state	string	true
vertiv_liebert	liebert.sysname	string	true
vertiv_liebert	liebert.version	string	true
vmware	http.vmware.thumbprint	string	true

vmware	service.product	string	true
vmware	service.vendor	string	true
vmware	service.version	string	true
vnc	vnc.banner	string	true
vnc	vnc.desktopName	string	true
vnc	vnc.error	string	true
vnc	vnc.framebufferHeight	string	true
vnc	vnc.framebufferWidth	string	true
vnc	vnc.majorVersion	string	true
vnc	vnc.minorVersion	string	true
vnc	vnc.noAuth	string	true
vnc	vnc.securityTypeCount	numeric	true
vnc	vnc.securityTypes	string	true
vnc	vnc.version	string	true
wbsm	wbsm.active	string	true
webmin	webmin.port	string	true
webmin	webmin.scheme	string	true
webmin	webmin.server	string	true
wiznet	wiznet.fields	string	true
wsd	wsd.addrs	string	false
wsd	wsd.types	string	true
wsman	wsman.body	string	true
x11	x11.accessGranted	string	true
x11	x11.anonymousAccess	string	true
x11	x11.displayNumber	string	true
x11	x11.failureReason	string	true
x11	x11.status	string	true
x11	x11.vendor	string	true
x11	x11.version	string	true
x2ap	x2ap.criticality	string	true

x2ap	x2ap.messageType	string	true
x2ap	x2ap.procedureCode	string	true
xdmcp	xdmcp.authName	string	true
xdmcp	xdmcp.hostname	string	true
xdmcp	xdmcp.opcode	string	true
xdmcp	xdmcp.status	string	true
xdmcp	xdmcp.version	string	true
zabbix	zabbix.agentData	string	true
zabbix	zabbix.agentVersion	string	true
zabbix	zabbix.cpe23	string	true
zabbix	zabbix.remoteCommandsEnabled	bool	true
zabbix	zabbix.version	string	true
zabbix-agent	zabbix.agentVersion	string	true
zebra	host.name	string	true
zebra	zebra.deviceName	string	false
zebra	zebra.firmware	string	false
zebra	zebra.ip	string	false
zebra	zebra.model	string	false
zebra	zebra.response	string	false
zebra	zebra.serial	string	false
zookeeper	zk.access	string	true
zookeeper	zookeeper.access	string	true
zookeeper	zookeeper.banner	string	true
zookeeper	zookeeper.connections	numeric	true
zookeeper	zookeeper.imok	string	true
zookeeper	zookeeper.mode	string	true
zookeeper	zookeeper.nodeCount	numeric	true
zookeeper	zookeeper.port	numeric	true
zookeeper	zookeeper.restricted	string	true
zookeeper	zookeeper.restrictionReason	string	true
zookeeper	zookeeper.ruok	string	true

zookeeper	zookeeper.srvr	string	true
zookeeper	zookeeper.target	string	true
zookeeper	zookeeper.version	string	true
zookeeper	zookeeper.versionRaw	string	true
zyxel	zyxel.builddate	string	true
zyxel	zyxel.dhcpstate	string	true
zyxel	zyxel.firmware	string	true
zyxel	zyxel.firstlogin	string	true
zyxel	zyxel.gateway	string	true
zyxel	zyxel.hostname	string	true
zyxel	zyxel.ip	string	true
zyxel	zyxel.mac	string	true
zyxel	zyxel.maxport	string	true
zyxel	zyxel.model	string	true
zyxel	zyxel.subnetmask	string	true
zyxel	zyxel.uptime	string	true

Release notes

No release notes available.

Index

2

2FA: 96

A

Account API: 97 622
Action: 474
Activating your account: 3
Active scan: 128
Active scanning: 124
Adaptive Cards: 477
Add multiple users: 108
Address ranges: 129
Administrators: 92
AI threads and artifacts: 4
Alert messages: 476
Alerts: 8
Alerts: 473 473 578
Alive: 437
Amazon Web Services: 214
Analysis reports: 412
Annotators: 92
API client: 623 629
API client credentials: 622
API key: 622
Area: 98
ARP: 437
Asset category: 435
Asset data: 683
Asset functions: 436
Asset geolocation: 445
Asset risk report: 442
Asset route pathing: 463
Asset route pathing report: 617
Assets: 426 434
Assets back online: 409
Assets changed: 409
Assets ignored: 409
Assets marked offline: 409
Assets unchanged: 409
Atlassian Insight: 357
Authenticated SNMP v3: 148
Authentication passphrase: 148
Authentication protocol: 148
AuthNoPriv: 148

AuthPriv: 148
Automated installers: 30
Automatic queries: 579
Autonomous discovery: 134
AWS: 214 223
Azure AD: 117 296

B

Billing: 92
Billing information: 122
Binary signature: 36
Block Kit: 477
Bridge report: 463
Browser versions: 678
Bulk asset update: 431
Bulk import users: 108

C

CA: 444
Caret: 494
CDE: 463
CEE: 67
Censys Search: 225
Certificate authority: 444
Certificates: 9
Change report: 687
Channel: 475
Channels: 473
CIDR allow list: 148
Cisco ASA: 671
Cisco Catalyst: 613
Cisco Meraki Dashboard: 227
Client token: 623
CMDB: 369
Colors: 618
Common Name: 443
Community strings: 147
Comparison: 606
Compliance: 439
Connect: 432
Connector task: 210
Container: 20
Coverage reports: 602
Credentials: 146
Criticality: 441
CrowdStrike: 229
Cryptographic bill of materials: 620
CSV: 580
Curly brackets: 479

Custom integration: 235
Custom integration: 235
Custom integration script versions: 4
Custom role: 98

D

Dashboard: 8
Dashboards: 414
Data deletion after account termination: 4
Data formats: 679
Data retention: 87
Debug SNMP v3: 150
Defender: 281
Department: 49
Deploy: 20
Deployment plan: 40
Detected by: 437
Device type: 435
DHCP: 426
Diagnostics collection: 85
Differences: 606
Directory services: 433
Distinguished name: 443
DN: 443
DNS: 434
Docker: 20
Domain membership: 412
Download: 20
Download token: 622
Dragos: 273
Duplicate assets: 51

E

EC2: 223
ElasticSearch: 67
Email: 473
Email runZero Users: 473
Embedded devices: 20
End-of-Life: 439
Entra: 117
Entra ID: 296
ESSID: 553
ESXi: 352
Event records: 4
Events: 473
Example queries: 506
Explorer: 20 33 128
Export API: 622
Export asset data: 580

Export token: 622
External Asset Report: 604
External network discovery: 20
External reporting: 609
External users: 110

F

Filter: 504
Finding code: 561
Findings: 439
Findings: 561
Fingerprint results: 426
Firewall: 671
First scans: 140
Flat networks: 51

G

GCP: 276
GeoNames: 447 449
Global credential: 146
Glossary: 669
Goal tracking: 458
Goals: 458
Google Chrome: 674
Google Cloud Platform: 276
Google Workspace: 278
Grace period: 6
Grant: 629
Graphviz: 618
Group directory: 433
Group mapping search: 572
Group search: 571
Groups: 9
Groups changed: 409
Groups unchanged: 409
Groups updated by task: 410

H

Hardware: 437
High-availability: 81
Historical changes: 606
Hop: 617
Host ping: 133
Hosted External Explorer: 128
Hostnames: 129
Hostnames: 426 434 434
HP iLO: 581

I

IAM: 223
ICMP: 437
ICMP ping: 134
Insights: 8
Insights: 416
InsightVM: 325 326
InsightVM Cloud: 326
Installation: 3
IntegrationHub ETL: 369
Internet Exposure: 439
Inventory: 426
Inventory views: 8
Invoices: 123
IoT: 205
IP addresses: 426 434
IPv4: 602
IRE: 369
Issues: 406
ITOM: 369

J

Jira: 362
Jira Service Management: 357
Journal: 67
Journalctl: 67
JSON: 580
JSONL: 680

K

Knowledgebase: 671

L

LDAP: 287
Licensing: 122
Logging: 67

M

MAC addresses: 426 434
Magnifying glass: 603
Mattermost: 473
Max group size: 131
Max group size: 672
MFA: 5
Microsoft 365: 281
Microsoft 365 Defender: 281

Microsoft Azure: 290
Microsoft Endpoint Configuration Manager: 294
Microsoft Identity Platform: 118
Microsoft Intune: 303
Microsoft Teams: 477
MID Server: 369
Miradore: 309
Missing subnets: 602
MSI package: 30
Multi-factor authentication: 96
Multi-homed: 614
Mustache: 477

N

NameID: 112
NAT: 672
Nessus: 348
Nessus Professional: 345
NetBox: 311
Network bridge report: 614
Network bridges: 411
Network examples: 45
Network Map: 143 583
Network paths: 617
Network segmentation: 463
Newly discovered assets: 409
Newly discovered groups: 409
Newly discovered users: 409
Nexpose: 325 330
No access: 93
NoAuthNoPriv: 148
Npcap: 36

O

Offline: 434
Offline asset expiration: 4
Offline installation: 78
Okta: 119
On-premises: 53
Open Access: 439
Operating system: 426 435
Organization: 684
Organization API: 622
Organization API token: 623
Organization Overview Report: 601
Organization search: 574
Organizations: 128
OT: 205
Outlier score: 438

Outliers: 412
Ownership: 452

P

Packet rate: 130
Panther: 367
Passive: 124
Passive sampling: 156
Password rules: 19
PCI DSS: 463
Planning: 3
Ports: 131
PostgreSQL: 79
Prebuilt searches: 579
Prescan modes: 133
Primary address: 434
Prisma Cloud: 323
Privacy passphrase: 149
Privacy protocol: 149
Private IP addresses: 602
Probes: 131
Profile images: 19
Project: 50
Project assets: 122
Promiscuous mode: 156
Protocol gateway: 465
Protocols: 688
Public key algorithm: 443

Q

Qualys VMDR: 317
Queries: 503
Queries: 579
Query language: 579
Query library: 579

R

Rapid Response: 439
Rapid7 InsightVM: 326
Rapid7 Nexpose: 330
Raspberry Pi: 31
Recent assets: 122
Recurring scan: 456
Red border: 602
Reinstall: 34
Remote logging: 67
Report: 606
Reports: 8
REST: 624 624

RFC 1918: 140 142 412
RFC1918: 51 602
Risk: 441
Risk Management: 439
Risk rank: 561
Role: 98
Roles: 91
Router: 671
RPMs: 79
RTT: 437
Rules: 473 473
Rules: 476
Rules submenu: 476

S

SAML: 112 115
Sample Queries: 410
Sample rate: 133
Scan completed: 484
Scan data: 680
Scan data expiration: 4
Scan probe: 210
Scan result: 682
Scan scope: 434
Scan speed: 6
Scan speed: 672
Scan stats: 682
Scan template: 143
Scanner: 153
Scans: 455
Scheduled scans: 6
Scope: 129
Screenshot: 33
Screenshots: 23 8 635
Search: 426
Search assets: 512
Search certificates: 548
Search credentials: 575
Search explorers: 568
Search groups: 558
Search queries: 577
Search query rule: 484
Search reports: 567
Search scan templates: 560
Search services: 529
Search software: 537
Search syntax: 504
Search users: 556
Search vulnerabilities: 540
Search WiFi networks: 553

Searching tasks: 563
Secondary addresses: 434
SecurityGate.io: 368
Segmentation: 463
Self-hosted: 53 153
Self-signed: 444
Sendgrid: 473
SentinelOne: 332
Service Graph connector: 369
ServiceNow: 369
Services: 426 8 688
Shodan: 335
Side-by-side: 606
Sign up: 15 19
Signature algorithm: 443
SIP: 671
Site: 426 434 684
Site search: 574
Sites: 128
Slack: 473 477 502
SNMP: 147
SNMP credentials: 148
SNMP v2 Communities: 148
Software: 9
Software groups: 536
SPAN: 156 156
Splunk: 372 372
SSID: 553
SSO: 5 96 112 120
SSO group mapping: 115
Stale asset expiration: 4
Stale integration attribute expiration: 4
Stale vulnerability expiration: 4
Starlark: 235
Stateful firewalls: 128
Statistics: 603
Sub-asset: 465
Subject: 443
Subject Alternative Name: 443
Subnet: 585
Subnet analysis: 603
Subnet sampling: 133
Subnet size: 133
Subnet utilization: 142
Subnet utilization: 411
Subnets: 426 602
Sumo Logic: 373 373 379
Superuser: 92
Suppress: 461
Suppressing: 461

Suppression: 461
Switch topology: 143 612
Switch topology: 411 412
Syslog: 67
System events: 578
System requirements: 3

T

Tailscale: 337
Tanium: 339
TAP: 156
Task details: 409
Tasks: 455
Tasks overview page: 455
TCP ping: 134
TCP services: 437
Team member: 94
Templates: 476
Tenable Nessus: 342 348
Tenable Nessus Professional: 345
Tenable Security Center: 342
Tenable Vulnerability Management: 342
Tenable.io: 342 342
Tenable.sc: 342 350
Thinkst Canary: 394
Time To Live: 131
Tines: 385
TLS certificates: 443
Token: 622
Topology: 411
Total assets seen by task: 409
Traceroute: 617
Traffic sampling: 124
Two-factor authentication: 96
Type of Service: 131 131

U

UAC: 21
UDP ping: 134
UDP services: 437
Unmapped: 411
Unmapped assets: 612
Unmapped MACs: 143
Upgrade: 34
User directory: 433
User groups: 105
User search: 569
Users: 9
Users changed: 409

Users unchanged: 409
Users updated by task: 409
UUID: 684

V

VCenter: 352
Verb: 98
Verifier: 36
Viewers: 92
Virtual machine: 352
Virtual machines: 672
Virtualized system: 20
VMware: 20 352 672
VPN gateways: 128
Vulnerabilities: 9 439
Vulnerability groups: 537
Vulnerability instances: 540

W

Web screenshots: 3
Webhook: 473
Webhooks: 476
Widget library: 415
WiFi: 433
Wireless networks: 9
Wiz: 353
World Map: 598

X

X.509: 443