ZendDiff: Differential Testing of PHP Interpreter

Yuancheng Jiang∗, Jianing Wang∗, Qiange Liu, Yeqi Fu,
Jian Mao, Roland H. C. Yap, Zhenkai Liang

ASE, November 2025
Seoul, South Korea

Background

- **PHP's Dominance in Web Development**
  - Powers the majority (**over 70%**) of today's websites.
  - ***Zend*** engine: >1M lines of *C* ⇒ *large, security-critical attack surface*.

Background

- **PHP's Dominance in Web Development**
  - Powers the majority (**over 70%**) of today's websites.
  - ***Zend*** engine: >1M lines of *C* ⇒ *large, security-critical attack surface*.
- **Existing Research Efforts: Fuzzing-Focus** (SEC'12, NDSS'19, S&P'21, SEC'25, AsiaCCS'25)
  - Grammar/semantic-guided test case generation.
  - *Oracle*: Crashes & Sanitizers
    - Great for memory errors, **weak for silent miscomputations**.

Background

- **PHP's Dominance in Web Development**
  - Powers the majority (**over 70%**) of today's websites.
  - ***Zend*** engine: >1M lines of *C* ⇒ *large, security-critical attack surface*.
- **Existing Research Efforts: Fuzzing-Focus** (SEC'12, NDSS'19, S&P'21, SEC'25, AsiaCCS'25)
  - Grammar/semantic-guided test case generation.
  - *Oracle*: Crashes & Sanitizers
    - Great for memory errors, **weak for silent miscomputations**.
- **Research Question:**

> Beyond explicit security issues, can we systematically detect ***logic bugs*** that silently lead to incorrect results in PHP?

The Missing Test Oracle for PHP

- **The Lack of An Effective Test Oracle.**
  - **Metamorphic Testing:** powerful but needs hand-crafted relations --- not scalable for PHP's vast semantics.
  - **Differential Testing:** no good alternative implementation of the PHP interpreter (HHVM dropped PHP in 2019).

The Missing Test Oracle for PHP

- **The Lack of An Effective Test Oracle.**
  - **Metamorphic Testing:** powerful but needs hand-crafted relations --- not scalable for PHP's vast semantics.
  - **Differential Testing:** no good alternative implementation of the PHP interpreter (HHVM dropped PHP in 2019).
- **Observation**:
  - Since PHP 8, Zend ships **JIT** that follows execution paths distinct from the interpreter.

The Missing Test Oracle for PHP

- **The Lack of An Effective Test Oracle.**
  - **Metamorphic Testing:** powerful but needs hand-crafted relations --- not scalable for PHP's vast semantics.
  - **Differential Testing:** no good alternative implementation of the PHP interpreter (HHVM dropped PHP in 2019).
- **Observation**:
  - Since PHP 8, Zend ships **JIT** that follows execution paths distinct from the interpreter.
- **Insight**:
  - Treat **non-JIT** vs **JIT** as two "implementations". (CCS'22, SOSP'23)
  - If the same input program produces *inconsistent* outputs, flag a **logic bug**.

Challenges

- **C1 Inconsistency detection in the presence of non-determinism.**
  - Two correct runs can legitimately differ, due to inherent non-deterministic factors at runtime, such as memory address, thread scheduling, or random number generation.

Challenges

- **C1 Inconsistency detection in the presence of non-determinism.**
  - Two correct runs can legitimately differ, due to inherent non-deterministic factors at runtime, such as memory address, thread scheduling, or random number generation.
- **C2 Exercising PHP's JIT functionality sufficiently.**
  - Only toggling a JIT configuration flag rarely exposes the full PHP's optimizations, compared with V8's automated JIT strategy.

Challenges

- **C1 Inconsistency detection in the presence of non-determinism.**
  - Two correct runs can legitimately differ, due to inherent non-deterministic factors at runtime, such as memory address, thread scheduling, or random number generation.
- **C2 Exercising PHP's JIT functionality sufficiently.**
  - Only toggling a JIT configuration flag rarely exposes the full PHP's optimizations, compared with V8's automated JIT strategy.
- **C3 Probing internal miscomputations.**
  - Logic errors can lurk behind correct-looking outputs, such as an incorrect intermediate value that never reaches final outputs.

Our Solutions

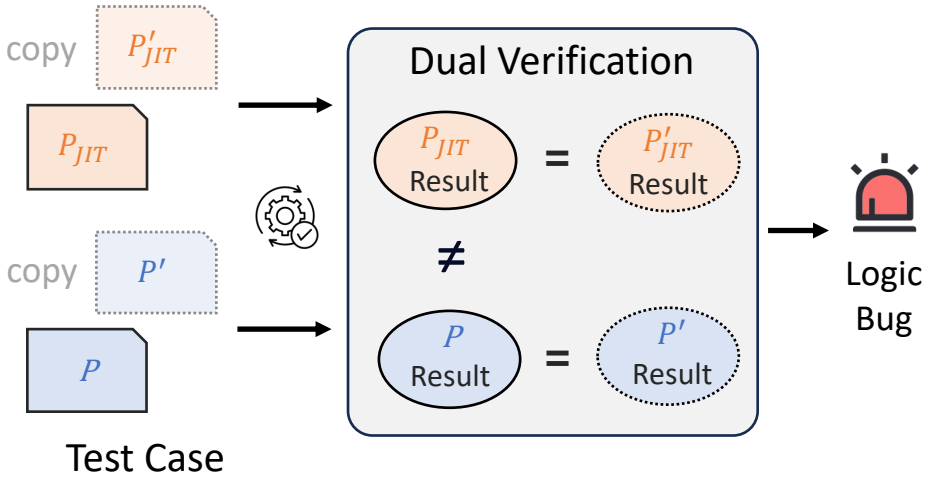
- **C1.** Inconsistency detection in the presence of non-determinism.
- **Solution: Dual Verification**
  - Executes each test case twice under both execution modes.
  - Program P (Interpreter), P_{JIT}, equivalent clones P' and P'_{JIT}
  - Original Oracle:

$$R(P) == R(P_{JIT})$$

  - Dual Verification Oracle:

$$R(P) == R(P') \;\&\&\; R(P_{JIT}) == R(P'_{JIT})$$

  - Discard flaky tests.



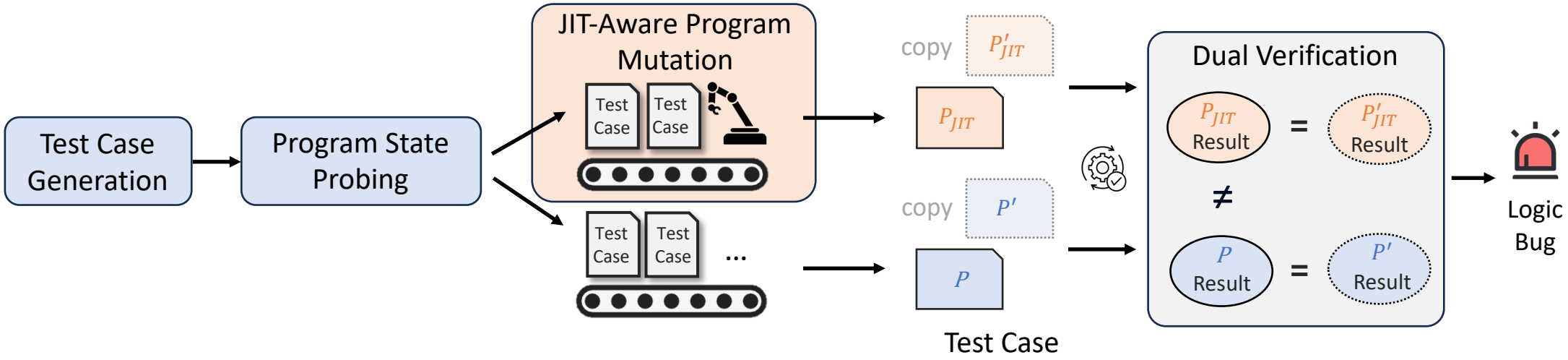
Our Solutions

- **C2.** Exercising PHP's JIT functionality sufficiently.
- **Solution: JIT-aware Program Mutation**
  - Proactively trigger compilation & optimization by three mutation strategies: fine-grained JIT trigger controls, threshold-dependent activation, and mutation of *Opcache* configurations.

Our Solutions

- **C2.** Exercising PHP's JIT functionality sufficiently.
- **Solution: JIT-aware Program Mutation**
  - Proactively trigger compilation & optimization by three mutation strategies: fine-grained JIT trigger controls, threshold-dependent activation, and mutation of *Opcache* configurations.
- **C3.** Detecting internal miscomputations.
- **Solution: Program State Probing**
  - Inject lightweight probes to capture fine-grained internal program states (e.g., variable values, class attributes).

ZendDiff Overview



- **Three practical techniques tailored for PHP.**
  - Dual Verification (C1), JIT-aware Program Mutation (C2), Program State Probing (C3).
- **Implementation**: ~2K LOC Python; No invasive engine patches required.

Evaluation: Effectiveness on Logic Bug Detection

- **Key Findings**
  - 51 total reports: **37 fixed**, **3 confirmed**, 5 dup, 6 expected.
  - Of the 40 confirmed new issues: **34 JIT**, **3 non-JIT**, **3 both**.
  - Touches **30 files**; **1,958 LOC changes** (1,802 additions, 156 deletions).

Evaluation: Effectiveness on Logic Bug Detection

- **Key Findings**
  - 51 total reports: **37 fixed**, **3 confirmed**, 5 dup, 6 expected.
  - Of the 40 confirmed new issues: **34 JIT**, **3 non-JIT**, **3 both**.
  - Touches **30 files**; **1,958 LOC changes** (1,802 additions, 156 deletions).
- **Case Study**

```php
<?php
$root = simplexml_load_string('../></root>');
$spattr = $root->child->attributes('special-ns');
var_dump(Dom\import_simplexml($spattr));
?>
```

```
Dom\import_simplexml(): Return value must be of type
Dom\Element, Dom\Attr returned in Unknown on line 0
```

Non-JIT

```
object(Dom\Attr)#2 (22) {["namespaceURI"] … }
```

JIT

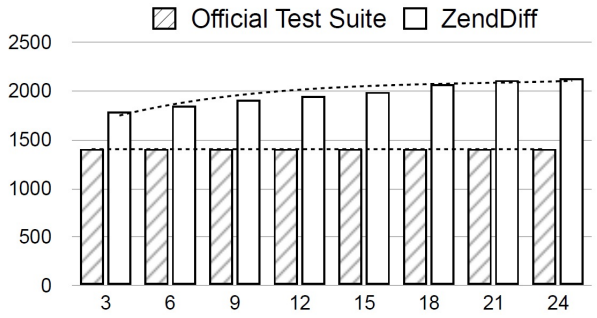
  - **Root Cause**: missed type annotation in non-JIT mode.
  - **Impact**: persisted >4 years

Evaluation: Comparison with Baselines

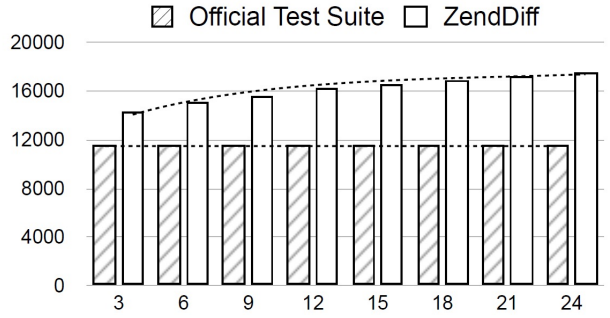
- **Complementary** with FlowFusion (SOTA fuzzer, USENIX Sec'25).
  - ZendDiff finds logic bugs FlowFusion typically won't.
  - FlowFusion still superior for memory-safety faults.

Evaluation: Comparison with Baselines

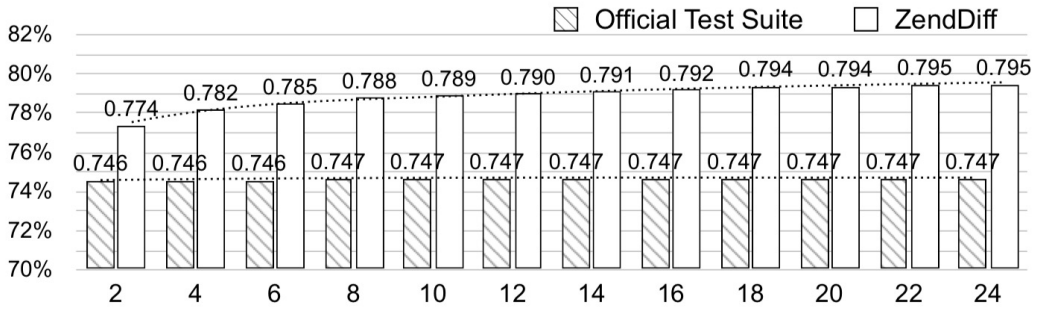
- **Complementary** with FlowFusion (SOTA fuzzer, USENIX Sec'25).
  - ZendDiff finds logic bugs FlowFusion typically won't.
  - FlowFusion still superior for memory-safety faults.
- Comparison with the **Official PHP Test Suite** (18,000 distinct cases).



Unique DynASM Opcode



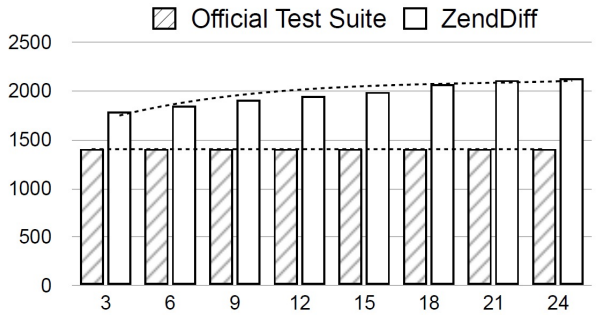
Unique DynASM Opcode Pair



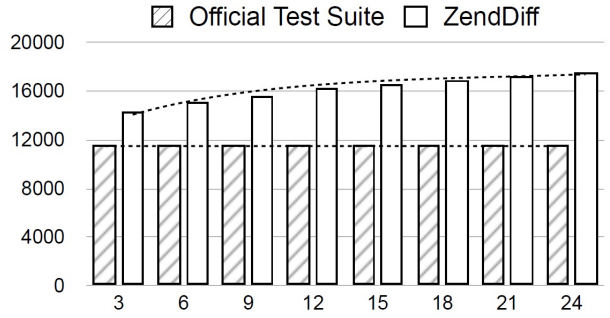
Code Coverage

Evaluation: Comparison with Baselines

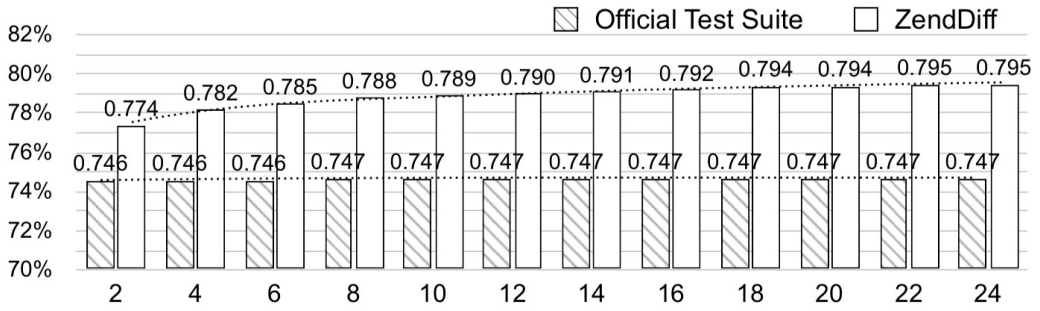
- **Complementary** with FlowFusion (SOTA fuzzer, USENIX Sec'25).
  - ZendDiff finds logic bugs FlowFusion typically won't.
  - FlowFusion still superior for memory-safety faults.
- Comparison with the **Official PHP Test Suite** (18,000 distinct cases).



Unique DynASM Opcode



Unique DynASM Opcode Pair



Code Coverage

ZendDiff increases **Zend opcode diversity**, and achieves **higher code coverage.**

Summary

- **A new oracle for discovering logic bugs in PHP.**
  - Identifying execution inconsistencies between non-JIT and JIT modes.
- **Three practical techniques tailored for PHP.**
  - Dual verification, JIT-aware program mutation, program state probing.
- **Impact:**
  - **51** previously unknow logic bugs; 37 fixed, 3 confirmed.
  - Our bug reports were positively received and acknowledged by the PHP developers.
- **ZendDiff** is available at:
  - https://github.com/YuanchengJiang/ZendDiff
- Meet us at the **ZendDiff poster**.
- **Q&A**