Machine Programming

Lecture 17 – PL Kaleidoscope III: Software Security and Testing

Logistics

- Assignment 3
  - Due today!
- Oral presentation
  - Schedule is out! Check on courselore
  - Contact the course staff immediately if there is any issue
- Final project
  - Group formation due today!
  - Final project proposal due next week!

PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70% of the Internet

Core PHP engine flaws enable remote code execution and Heartbleed-style memory disclosure across the web.

PHP2Shell: What you need to know | PHP2Shell: Technical Threat Briefing w/ Wiz Research



Sagi Tzadik, Shir Tamari, Nir Ohfeld, Tomer Schwartz

October 21, 2026 | 10 minute read

PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70% of the Internet

Core PHP engine flaws enable remote code

PHP2Shell: What you need to know



Sagi Tzadik, Shir Tamari, Nir Ohfeld, Tomer Schwartz

October 21, 2026 | 10 minute read

Executive Summary

- Wiz Research uncovered multiple critical vulnerabilities in core PHP, a popular general-purpose scripting language that powers 70% of all websites in the world. Successful exploitation could allow attackers to take control of affected servers, steal sensitive data, and deface the websites running on them.
- PHP is a backbone of the web: it's the programming language powering WordPress, Joomla, Drupal, Wikipedia's MediaWiki, and many others.
- These vulnerabilities include memory disclosure and unauthenticated remote code execution **in PHP itself, independent of the application running on top of it**, putting every affected website in immediate risk.
- Every PHP version released in the last decade is affected by at least one of these vulnerabilities, going back to version 5.4. Attackers can exploit them remotely, in some cases fairly easily. Immediate patching is strongly advised.
- On October 21, 2026, **PHP released the patch** fixing multiple memory-safety vulnerabilities in the core engine.
- These findings rank among the most severe vulnerabilities discovered in PHP, and arguably in software history. The underlying flaws had been present for decades, but demonstrating their exploitability required an exceptionally complex chain of steps, at the edge of what human researchers could practically achieve. Frontier AI models substantially reduced that barrier, turning a longstanding flaw into an exploit.

PHP2Shell: AI u
affecting 70% o

Core PHP engine

PHP2Shell: Wha

October 21, 2026 | 10

Executive

- Wiz Research unc
  powers 70% of all
  servers, steal sens
- PHP is a backbon
  MediaWiki, and m
- These vulnerabilit
  independent of t
- Every PHP version
  5.4. Attackers can
- On October 21, 20
- These findings ra
  underlying flaws h
  complex chain of
  substantially redu

The Vulnerabilities

All of the disclosed vulnerabilities are memory-safety issues that stem from bugs in how PHP handles fatal errors while processing requests, particularly out-of-memory errors. These bugs can leave a corrupted internal PHP state behind after a request ends unexpectedly. A later request or worker can reuse that stale state, leading to memory disclosure or corruption.

Below you can find a table that demonstrates what type of impact was demonstrated by each of the bugs we reported to PHP. The applications listed below are targets we chose for the RCE demonstrations. They are representative targets - exploitation does not require one of these CMSs, as the vulnerable code is in PHP itself.

Tracking ID	Bug	What an attacker can achieve	What we demonstrated
CVE-2026-109618	Upload-progress unserializer use-after-free	Corrupt process memory and execute code	Pre-authentication RCE on WordPress, Drupal and Joomla
	Remote process-memory disclosure	Unauthenticated disclosure of PHP worker memory	Application-agnostic remote memory disclosure
CVE-2026-109619	Fatal-error backtrace use-after-free	Corrupt process memory and execute code	Pre-authentication RCE on Drupal Pre-authentication RCE on WordPress
	OPcache type confusion	Corrupt shared PHP state and execute code	

PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70% of the Internet

Core PHP engine

PHP2Shell: Wha

October 21, 2026 | 10

Executive Summary

- Wiz Rese
  powers
  servers,
- PHP is a
  MediaW
- These v
  indeper
- Every PH
  5.4. Atta
- On Octo
- These fi
  underlyi
  complex
  substan

cripting language that

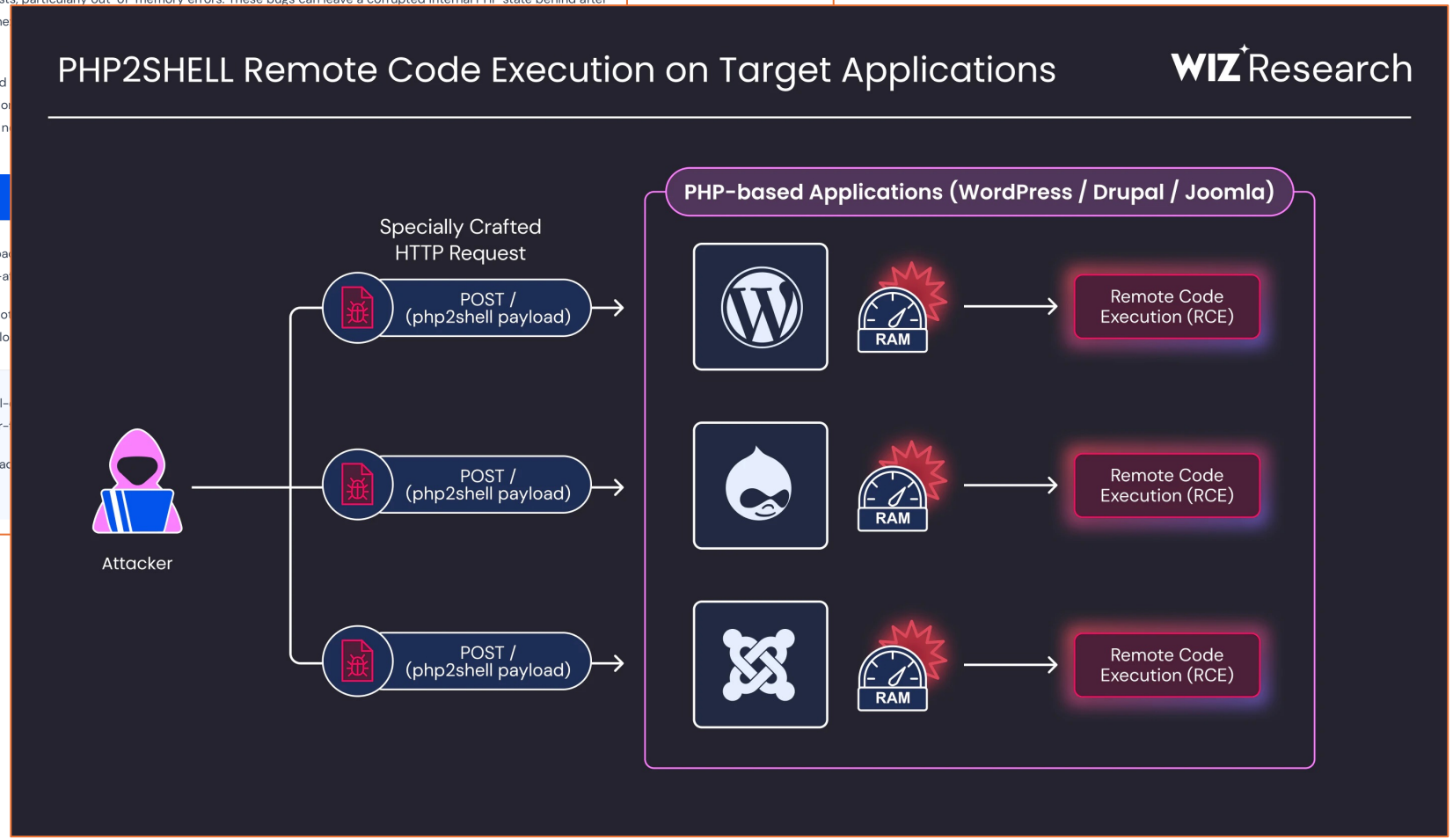
ntrol of affected

The Vulnerabilities

All of the disclosed vulnerabilities are memory-safety issues that stem from bugs in how PHP handles fatal errors while processing requests, particularly out-of-memory errors. These bugs can leave a corrupted internal PHP state behind after a request ends une
corruption.

Below you can find
PHP. The applicatio
exploitation does n

Tracking ID	Bug
CVE-2026-109618	Uploa use-a Remo disclo
CVE-2026-109619	Fatal- after- OPca



PHP2Shell: AI uncovers critical RCE vulnerabilities affecting 70%

Core PHP engine flaws enable remote code

PHP2Shell: What you need to know | PH



Sagi Tzadik, Shir Tamari, Nir Ohfeld, Tomer Schwartz

October 21, 2026 | 10 minute read

Information Technology Laboratory

NATIONAL VULNERABILITY DATABASE

NIST NATIONAL VULNERABILITY DATABASE NVD

VULNERABILITIES

CVE-2026-109618 Detail

UNDERGOING ENRICHMENT

The CVE is currently being enriched by the NVD Enrichment team. The enrichment process results in the association of reference link tags, CVSS, CWE, and CPE applicability statement data.

Description

PHP uses longjmp() for out-of-memory (OOM) errors to exit the current context and abort handling of the request. Some contexts may not properly account for this scenario and leave request memory in an uninitialized state before performing an allocation. The subsequent request cleanup procedure may read or branch on this memory. Multiple OOM strategies can consistently trigger this condition and lead to remote code execution (RCE).

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0 | SSVC

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

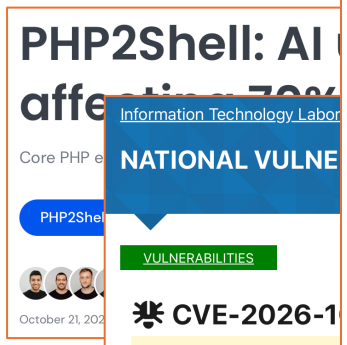
CVSS 4.0 Severity and Vector Strings:

NIST: NVD — N/A — NVD assessment not yet provided.

CNA: PHP Group — **CVSS-B** 9.2 CRITICAL — **Vector:** CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

QUICK INFO

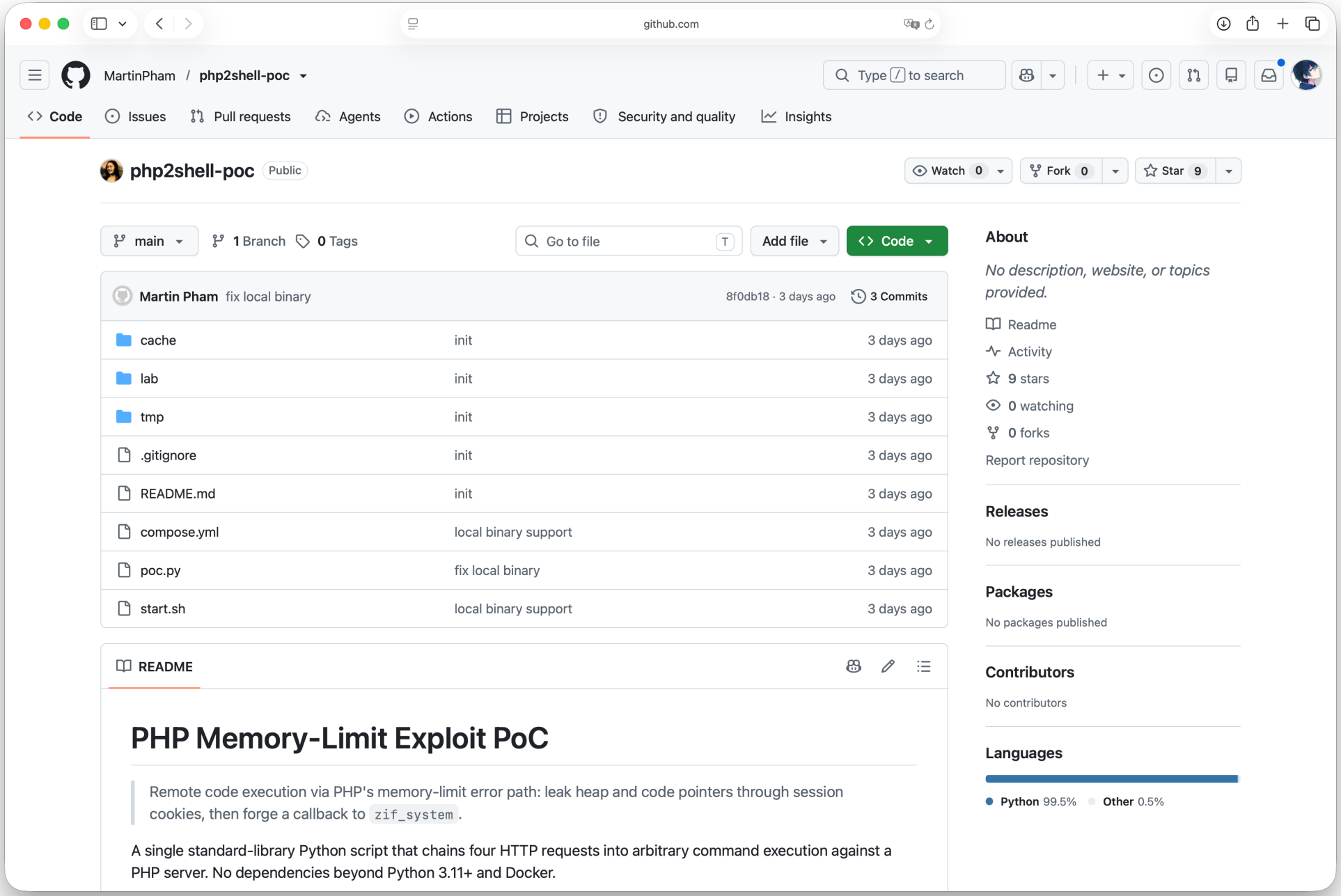
CVE Dictionary Entry:
CVE-2026-109618
NVD Published Date:
Oct 21, 2026
NVD Last Modified:
Oct 29, 2026
Source:
PHP Group

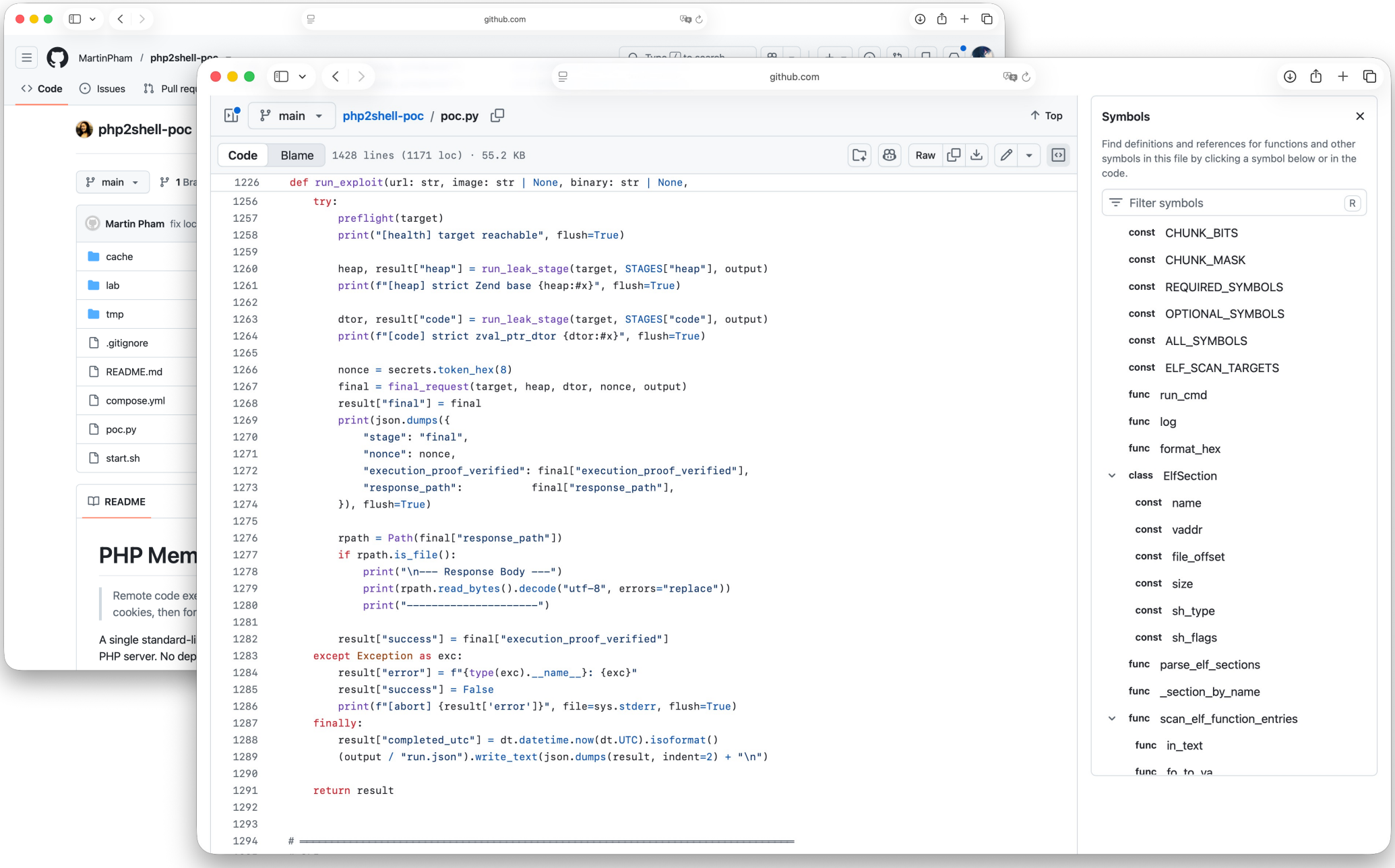


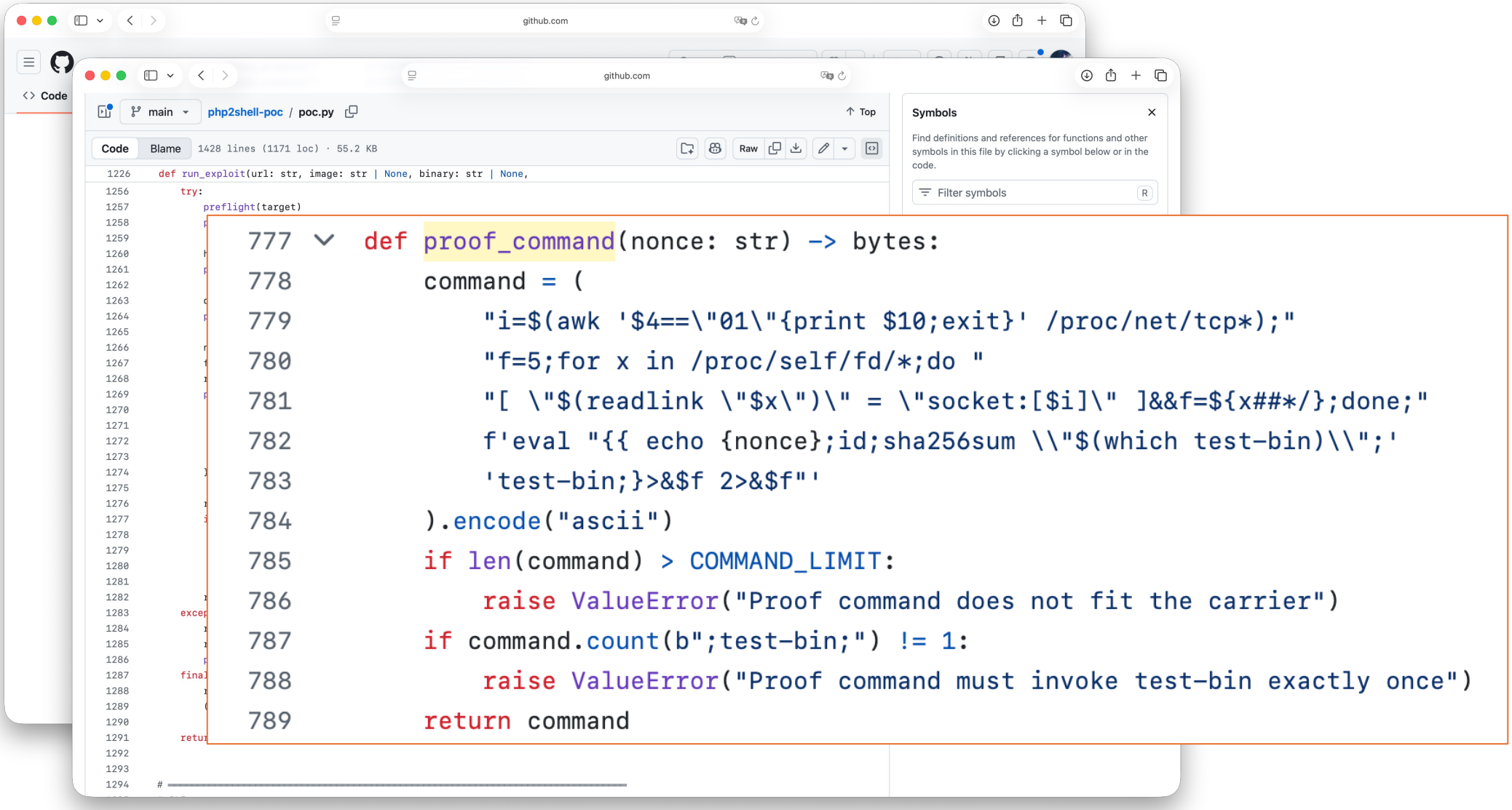
References to Advisories, Solutions, and Tools

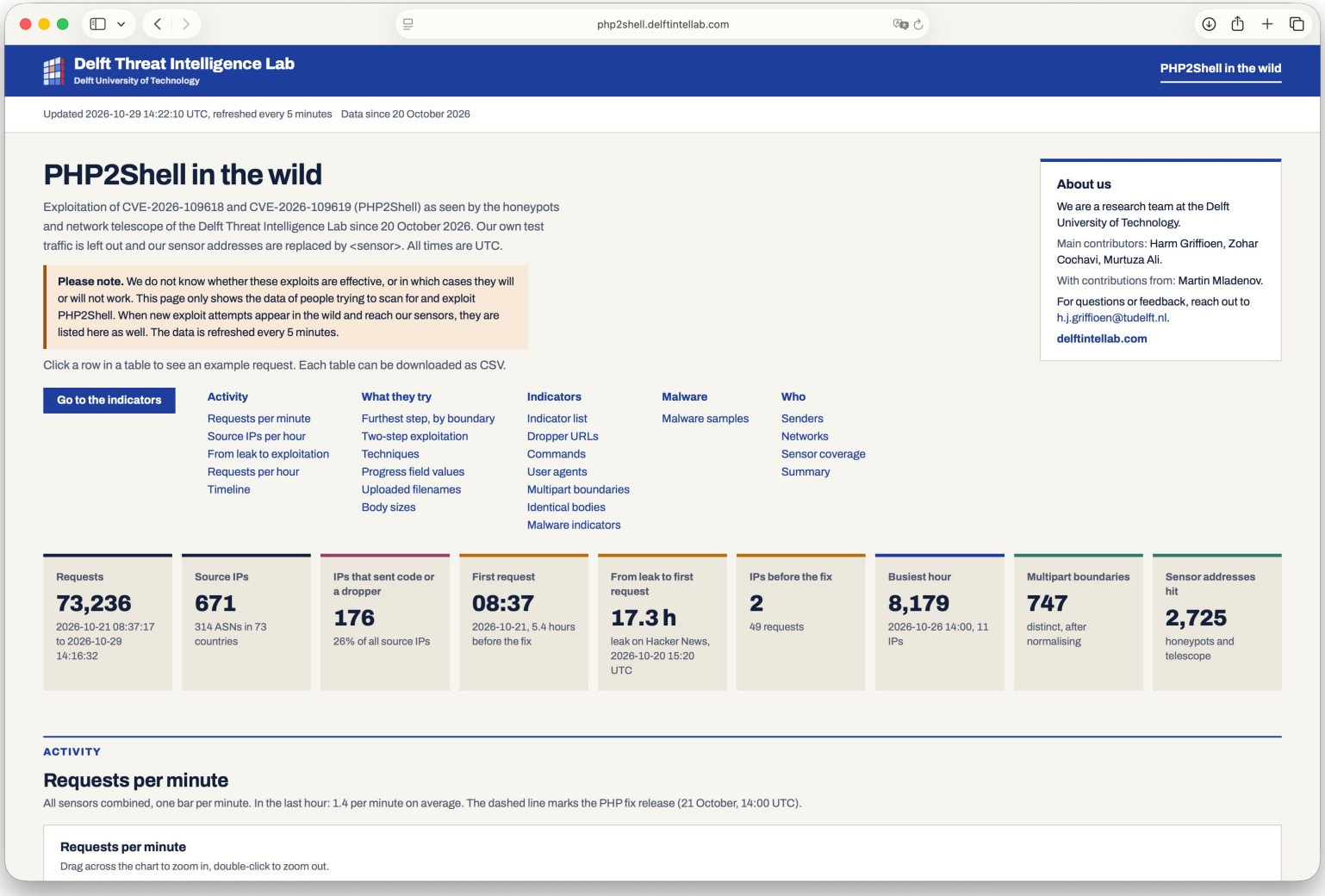
By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

URL	Source(s)	Tag(s)
http://seclists.org/fulldisclosure/2026/Oct/51	CVE	
http://www.openwall.com/lists/oss-security/2026/10/21/6	CVE	Mailing List, Third Party Advisory
http://www.openwall.com/lists/oss-security/2026/10/23/5	CVE	Mailing List, Third Party Advisory
https://arcticwolf.com/resources/blog/php2shell-critical-php-vulnerabilities-enable-unauthenticated-remote-code-execution/	CISA-ADP	Third Party Advisory
https://cert-portal.siemens.com/productcert/html/ssa-566512.html	siemens-SADP	Third Party Advisory
https://github.com/php/php-src/security/advisories/GHSA-w8hw-w9fx-j79w	PHP Group	Mitigation, Vendor Advisory
https://php2shell.delftintellab.com	CISA-ADP	Third Party Advisory
https://www.cisa.gov/known-exploited-vulnerabilities-catalog?field_cve=CVE-2026-109618	CISA-ADP	US Government Resource
https://www.wiz.io/blog/php2shell-php-rce	CISA-ADP	Third Party Advisory


github.com
MartinPham / php2shell-poc
Type / to search
Code
Issues
Pull requests
Agents
Actions
Projects
Security and quality
Insights
php2shell-poc
Public
Watch 0
Fork 0
Star 9
main
1 Branch
0 Tags
Go to file
T
Add file
Code
Martin Pham fix local binary
8f0db18 · 3 days ago
3 Commits
cache init 3 days ago
lab init 3 days ago
tmp init 3 days ago
.gitignore init 3 days ago
README.md init 3 days ago
compose.yml local binary support 3 days ago
poc.py fix local binary 3 days ago
start.sh local binary support 3 days ago
README
PHP Memory-Limit Exploit PoC
Remote code execution via PHP's memory-limit error path: leak heap and code pointers through session cookies, then forge a callback to zif_system.
A single standard-library Python script that chains four HTTP requests into arbitrary command execution against a PHP server. No dependencies beyond Python 3.11+ and Docker.
About
No description, website, or topics provided.
Readme
Activity
9 stars
0 watching
0 forks
Report repository
Releases
No releases published
Packages
No packages published
Contributors
No contributors
Languages
Python 99.5%
Other 0.5%


github.com
MartinPham / php2shell-poc
Code
Issues
Pull req
php2shell-poc
main
1 Bra
Martin Pham fix loc
cache
lab
tmp
.gitignore
README.md
compose.yml
poc.py
start.sh
README
PHP Mem
Remote code exe
cookies, then for
A single standard-li
PHP server. No dep
github.com
main
php2shell-poc / poc.py
Top
Code
Blame
1428 lines (1171 loc) · 55.2 KB
Raw
1226 def run_exploit(url: str, image: str | None, binary: str | None,
1256 try:
1257 preflight(target)
1258 print("[health] target reachable", flush=True)
1259
1260 heap, result["heap"] = run_leak_stage(target, STAGES["heap"], output)
1261 print(f"[heap] strict Zend base {heap:#x}", flush=True)
1262
1263 dtor, result["code"] = run_leak_stage(target, STAGES["code"], output)
1264 print(f"[code] strict zval_ptr_dtor {dtor:#x}", flush=True)
1265
1266 nonce = secrets.token_hex(8)
1267 final = final_request(target, heap, dtor, nonce, output)
1268 result["final"] = final
1269 print(json.dumps({
1270 "stage": "final",
1271 "nonce": nonce,
1272 "execution_proof_verified": final["execution_proof_verified"],
1273 "response_path": final["response_path"],
1274 }), flush=True)
1275
1276 rpath = Path(final["response_path"])
1277 if rpath.is_file():
1278 print("\n--- Response Body ---")
1279 print(rpath.read_bytes().decode("utf-8", errors="replace"))
1280 print("----------")
1281
1282 result["success"] = final["execution_proof_verified"]
1283 except Exception as exc:
1284 result["error"] = f"{type(exc).__name__}: {exc}"
1285 result["success"] = False
1286 print(f"[abort] {result['error']}", file=sys.stderr, flush=True)
1287 finally:
1288 result["completed_utc"] = dt.datetime.now(dt.UTC).isoformat()
1289 (output / "run.json").write_text(json.dumps(result, indent=2) + "\n")
1290
1291 return result
1292
1293
1294 # ==========
Symbols
Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.
Filter symbols
R
const CHUNK_BITS
const CHUNK_MASK
const REQUIRED_SYMBOLS
const OPTIONAL_SYMBOLS
const ALL_SYMBOLS
const ELF_SCAN_TARGETS
func run_cmd
func log
func format_hex
class ElfSection
const name
const vaddr
const file_offset
const size
const sh_type
const sh_flags
func parse_elf_sections
func _section_by_name
func scan_elf_function_entries
func in_text
func fo_to_va


github.com
Code
github.com
main
php2shell-poc / poc.py
Top
Code
Blame
1428 lines (1171 loc) · 55.2 KB
Raw
Symbols
Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.
Filter symbols
R
1226 def run_exploit(url: str, image: str | None, binary: str | None,
1256 try:
1257 preflight(target)
1283 excep
1287 final
1291 retu
1294 # ═══
777 def proof_command(nonce: str) -> bytes:
778 command = (
779 "i=$(awk '$4==\"01\"{print $10;exit}' /proc/net/tcp*);"
780 "f=5;for x in /proc/self/fd/*;do "
781 "[\"$(readlink \"$x\")\" = \"socket:[$i]\"]&&f=${x##*/};done;"
782 f'eval "{{ echo {nonce};id;sha256sum \\"$(which test-bin)\\";'
783 'test-bin;}>&$f 2>&$f"'
784).encode("ascii")
785 if len(command) > COMMAND_LIMIT:
786 raise ValueError("Proof command does not fit the carrier")
787 if command.count(b";test-bin;") != 1:
788 raise ValueError("Proof command must invoke test-bin exactly once")
789 return command


php2shell.delftintellab.com
Delft Threat Intelligence Lab
Delft University of Technology
PHP2Shell in the wild
Updated 2026-10-29 14:22:10 UTC, refreshed every 5 minutes Data since 20 October 2026
PHP2Shell in the wild
Exploitation of CVE-2026-109618 and CVE-2026-109619 (PHP2Shell) as seen by the honeypots and network telescope of the Delft Threat Intelligence Lab since 20 October 2026. Our own test traffic is left out and our sensor addresses are replaced by <sensor>. All times are UTC.
Please note. We do not know whether these exploits are effective, or in which cases they will or will not work. This page only shows the data of people trying to scan for and exploit PHP2Shell. When new exploit attempts appear in the wild and reach our sensors, they are listed here as well. The data is refreshed every 5 minutes.
About us
We are a research team at the Delft University of Technology.
Main contributors: Harm Griffioen, Zohar Cochavi, Murtuza Ali.
With contributions from: Martin Mladenov.
For questions or feedback, reach out to h.j.griffioen@tudelft.nl.
delftintellab.com
Click a row in a table to see an example request. Each table can be downloaded as CSV.
Go to the indicators
Activity
Requests per minute
Source IPs per hour
From leak to exploitation
Requests per hour
Timeline
What they try
Furthest step, by boundary
Two-step exploitation
Techniques
Progress field values
Uploaded filenames
Body sizes
Indicators
Indicator list
Dropper URLs
Commands
User agents
Multipart boundaries
Identical bodies
Malware indicators
Malware
Malware samples
Who
Senders
Networks
Sensor coverage
Summary
Requests
73,236
2026-10-21 08:37:17 to 2026-10-29 14:16:32
Source IPs
671
314 ASNs in 73 countries
IPs that sent code or a dropper
176
26% of all source IPs
First request
08:37
2026-10-21, 5.4 hours before the fix
From leak to first request
17.3 h
leak on Hacker News, 2026-10-20 15:20 UTC
IPs before the fix
2
49 requests
Busiest hour
8,179
2026-10-26 14:00, 11 IPs
Multipart boundaries
747
distinct, after normalising
Sensor addresses hit
2,725
honeypots and telescope
ACTIVITY
Requests per minute
All sensors combined, one bar per minute. In the last hour: 1.4 per minute on average. The dashed line marks the PHP fix release (21 October, 14:00 UTC).
Requests per minute
Drag across the chart to zoom in, double-click to zoom out.

Today's PHP Security Release, and How We're Strengthening PHP Security

Published on Oct 21, 2026 The PHP Foundation News Security

Earlier today, the PHP project released PHP 8.5.12, 8.4.27, 8.3.36, and 8.2.35. These releases fix two critical vulnerabilities that, under the right conditions, allow an unauthenticated attacker to execute code on servers running PHP. If you run PHP, it is imperative that you update now. The announcement on php.net contains and references technical details on the vulnerabilities, attack vectors, and instructions on how to update on which systems.

This post isn't about how to update. It's about how these vulnerabilities were found and fixed, what has changed in how PHP security is handled over the past two years, and what we need from the organizations that depend on PHP to keep PHP's security work going.

Two related critical remote code execution vulnerabilities

- CVE-2026-109618 / GHSA-w8hw-w9fx-j79w: **RCE via OOM corruption** (out-of-memory): If a request reached PHP's memory_limit at exactly the wrong moment, PHP could leave partially initialized internal data behind and later treat it as valid, corrupting memory. On Apache with mod_php, an unauthenticated request can trigger this before any application code runs, and full remote code execution was demonstrated with default settings. Remote code execution was also confirmed on PHP-FPM through application-level code paths.
- CVE-2026-109619 / GHSA-6wmf-4rv5-p3rp: **RCE via stale global state**: A fatal error at the wrong point could leave internal state pointing to memory that belonged to a previous request. A later request handled by the same worker could then use that freed memory. A use-after-free and worker crash were confirmed, and the reporter demonstrated remote code execution against a real-world application running on Apache with mod_php.

Today's PHP Security Release, and How We're Strengthening PHP Se

Published on Oct 21, 2026 The PHP Foundation News

Earlier today, the PHP project released PHP 8.5.1
critical vulnerabilities that, under the right conditi
on servers running PHP. If you run PHP, it is impe
php.net contains and references technical details
on how to update on which systems.

This post isn't about how to update. It's about ho
changed in how PHP security is handled over the
organizations that depend on PHP to keep PHP's

Two related critical remot vulnerabilities

- CVE-2026-109618 / GHSA-w8hw-w9fx-j79w:
  reached PHP's memory_limit at exactly the wr
  data behind and later treat it as valid, corruptir
  unauthenticated request can trigger this befor
  execution was demonstrated with default setti
  FPM through application-level code paths.
- CVE-2026-109619 / GHSA-6wmf-4rv5-p3rp:
  point could leave internal state pointing to mer
  handled by the same worker could then use th
  confirmed, and the reporter demonstrated rem
  running on Apache with mod_php.

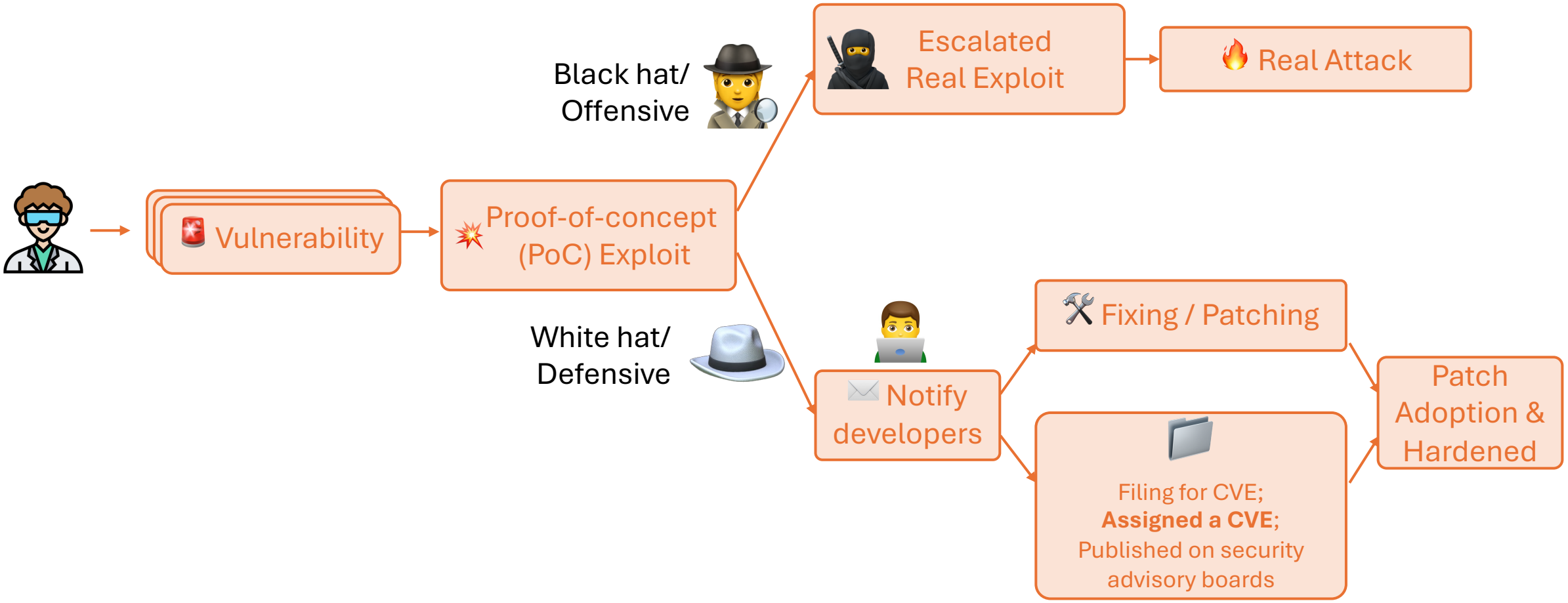
Building a sustainable professional security organization

Over the last months the way PHP security work is organized has changed considerably. In May 2026, the PHP Foundation created the Ecosystem Security Team, which works alongside the PHP security team on triage, coordination, and hardening. Volker Dusch's work as AI Security Engineer in Residence at the PHP Foundation has significantly improved communication and collaboration between the PHP security team and the Foundation. Alexandre Daubois joined the PHP Foundation in July to help the PHP security team triage, handle, and fix security reports and Daniel Scherzer joined in September to take on hardening work.

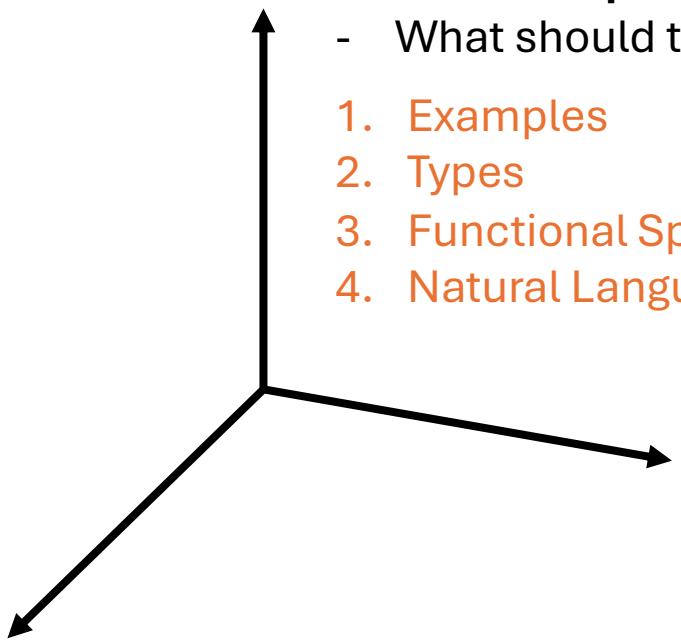
Earlier this year, we worked with Alpha-Omega to scan PHP with Anthropic's Claude Mythos. But since the Mythos scan, we have added other tools, including OpenAI's Daybreak Red for validating reports and exploits, alongside sanitizer builds and reproduction environments that match our CI on every platform PHP supports.

The PHP Foundation got the PHP project into the Patch the Planet program, as part of which security firm Calif began an audit of PHP on July 30th, 2026. On August 11th, researcher Ryan A at Calif reported an unconditional remote code execution with the same root cause as the OOM problem we had found in 2024. OpenAI independently reported the same root cause, followed by further attack vectors from OpenAI on September 18th and from Wiz on August 24th. More recently, Trail of Bits has also been auditing PHP. The reports from Calif, OpenAI, and Wiz made the full impact of the OOM problem clear, and from August 11th onward, fixing it became the top priority for all of the PHP Foundation's team working on security. Without the Foundation's funding it would have been impossible to have multiple people work on this issue full time, eventually through weekends and around the clock to coordinate and communicate with other involved parties.

Life-cycle of security vulnerabilities



The Course So Far



Behavioral Specification
- What should the program do?

1. Examples
2. Types
3. Functional Specifications
4. Natural Language

Structural Specification
- What is the space of the programs?

General Purpose Programming Language
Python / Java / C / Rust / …

Domain Specific Languages

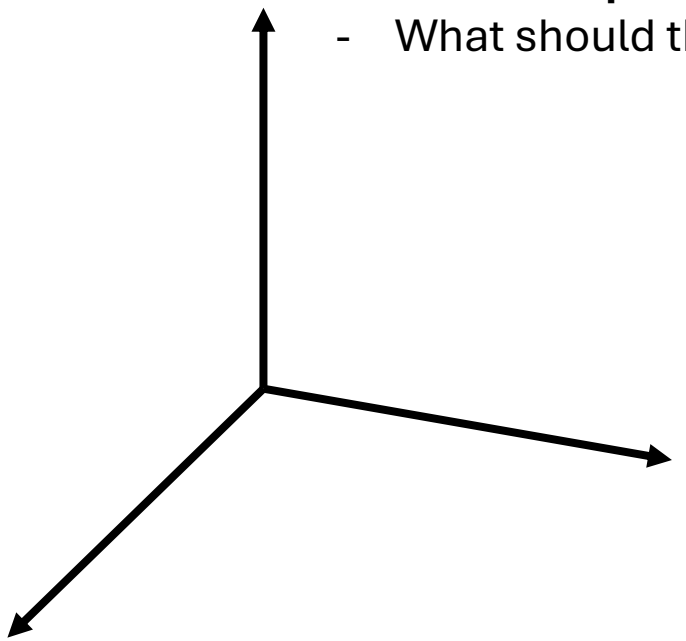
Synthesis Strategy
- How do we find such a program?

Enumeration: Bottom-up and Top-Down

Programming with Large Language Models
- Next token prediction, prompting, controlled decoding
- Iterative refinement, agentic frameworks and tool use
- Pre-training, fine-tuning, reinforcement learning

Dimensions in Program Synthesis

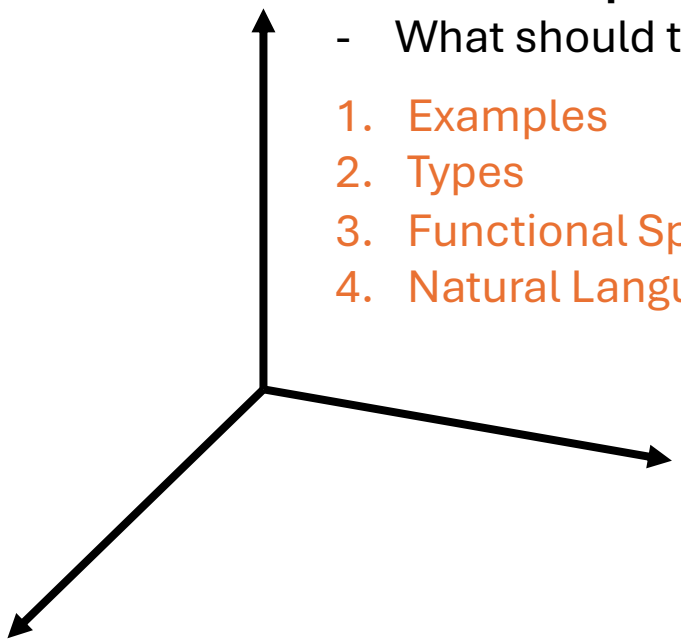
Behavioral Specification
- What should the program do?



Synthesis Strategy
- How do we find such a program?

Structural Specification
- What is the space of the programs?

The Course So Far



Behavioral Specification
- What should the program do?

1. Examples
2. Types
3. Functional Specifications
4. Natural Language

Structural Specification
- What is the space of the programs?

General Purpose Programming Language
Python / Java / C / Rust / …

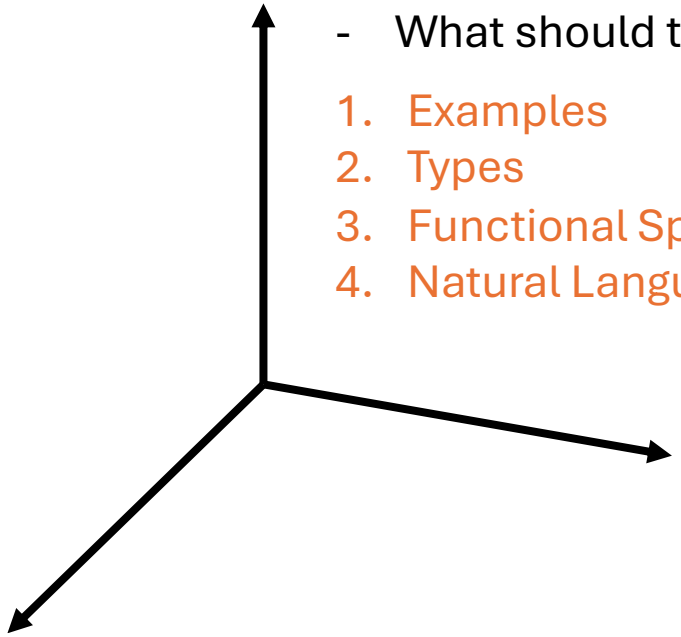
Custom Domain Specific Languages

Synthesis Strategy
- How do we find such a program?

Enumeration: Bottom-up and Top-Down

Programming with Large Language Models
- Next token prediction, prompting, controlled decoding
- Iterative refinement, agentic frameworks and tool use
- Pre-training, fine-tuning, reinforcement learning

Considering software quality...



Behavioral Specification

- What should the program do?

1. Examples
2. Types
3. Functional Specifications
4. Natural Language

Correctness Specifications

Structural Specification

- What is the space of the programs?

General Purpose Programming Language

Python / Java / C / Rust / ...

Fixed

Custom Domain Specific Languages

Synthesis Strategy

- How do we find such a program?

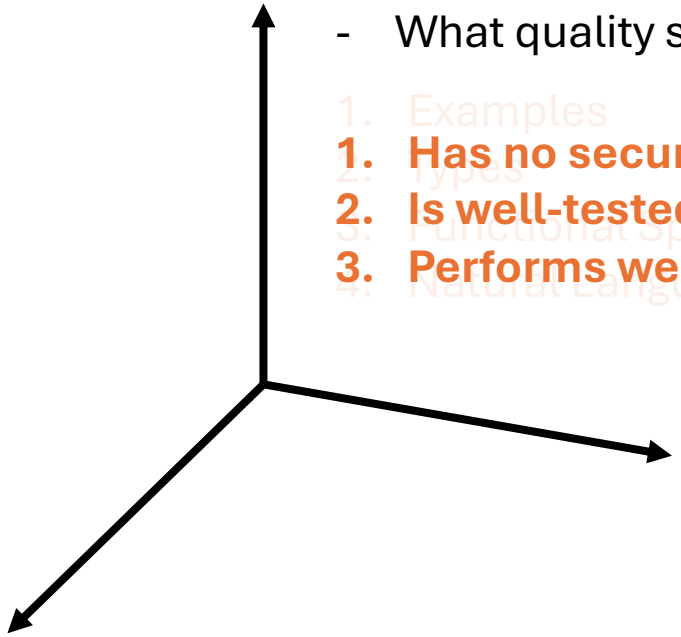
Enumeration: Bottom-up and Top-Down

Programming with Large Language Models

- Next token prediction, prompting, controlled decoding
- Iterative refinement, agentic frameworks and tool use
- Pre-training, fine-tuning, reinforcement learning

Fixed

Considering software quality...



Behavioral Specification

- What quality should programs have?

1. Examples
2. Types
3. Functional Specifications
4. Natural Language

1. **Has no security flaw** (safety, privacy, etc.)
2. **Is well-tested** (no regression, good coverage, etc.)
3. **Performs well** (fast, use less resource, etc.)

Quality Specifications

Synthesis Strategy

- How do we find such a program?

Enumeration: Bottom-up and Top-Down

Programming with Large Language Models

- Next token prediction, prompting, controlled decoding
- Iterative refinement, agentic frameworks and tool use
- Pre-training, fine-tuning, reinforcement learning

Fixed

Structural Specification

- What is the space of the programs?

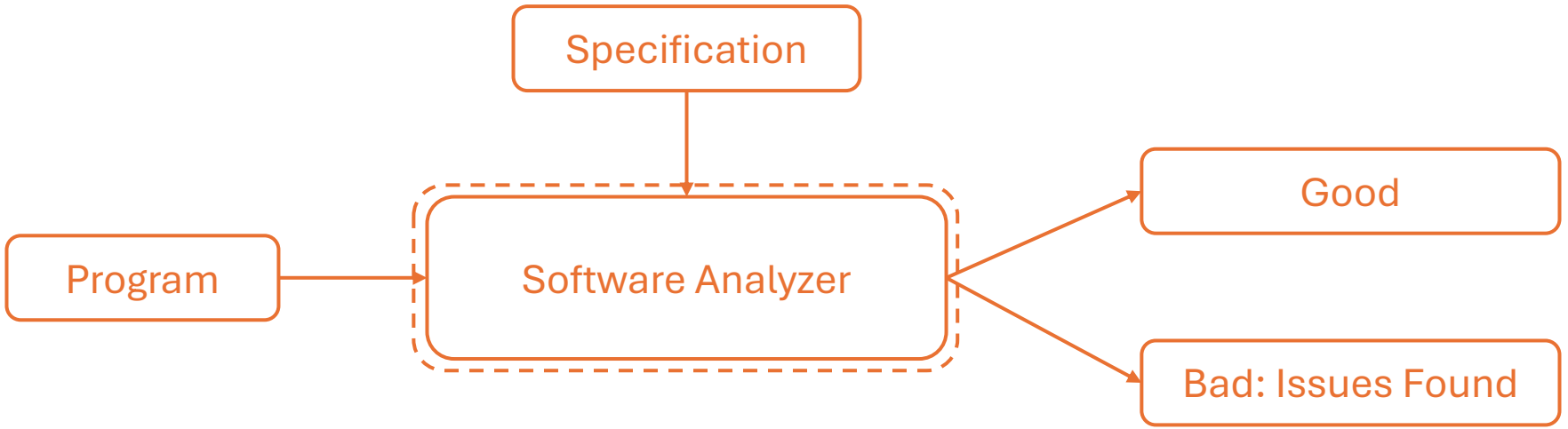
General Purpose Programming Language

Python / Java / C / Rust / ...

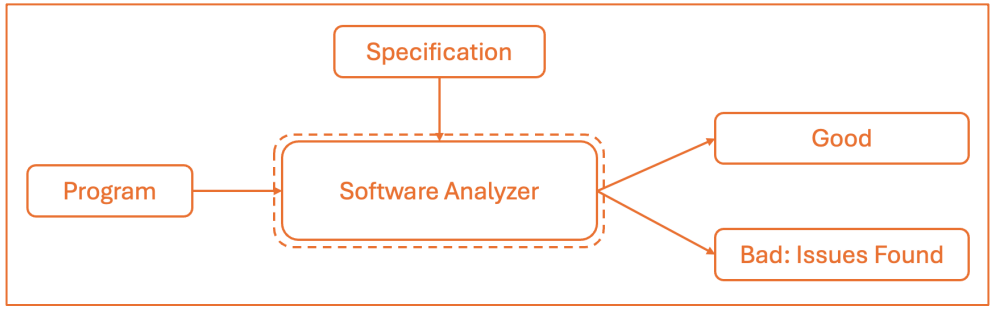
Custom Domain Specific Languages

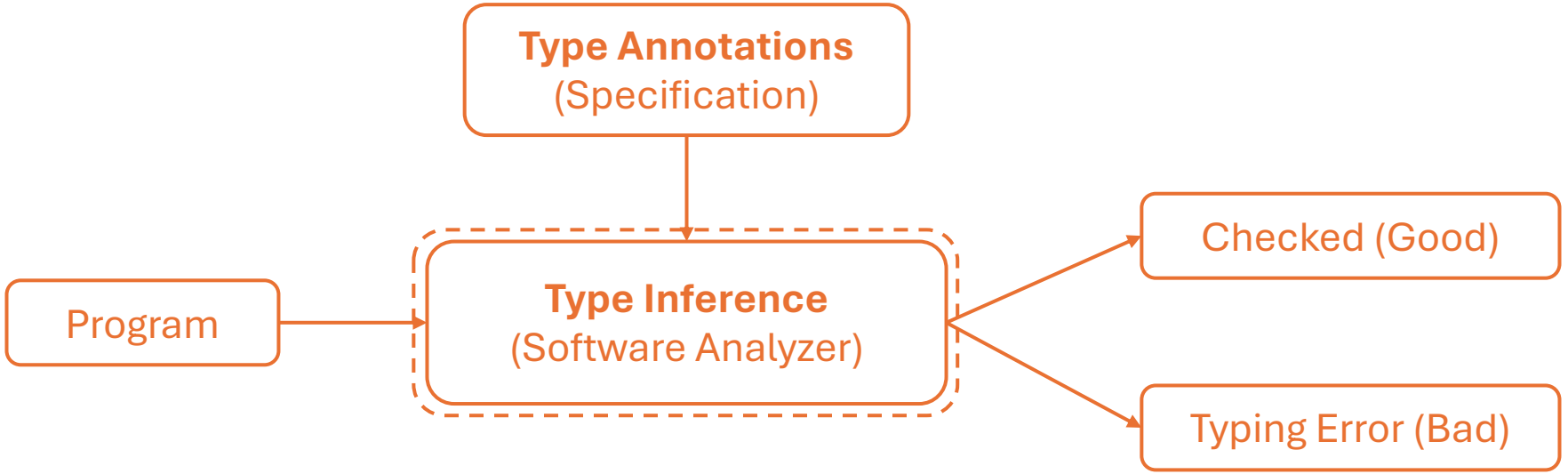
Fixed

Software Analysis

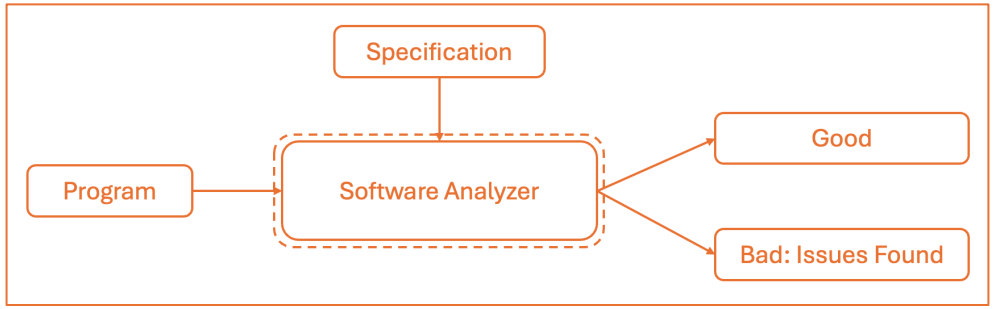


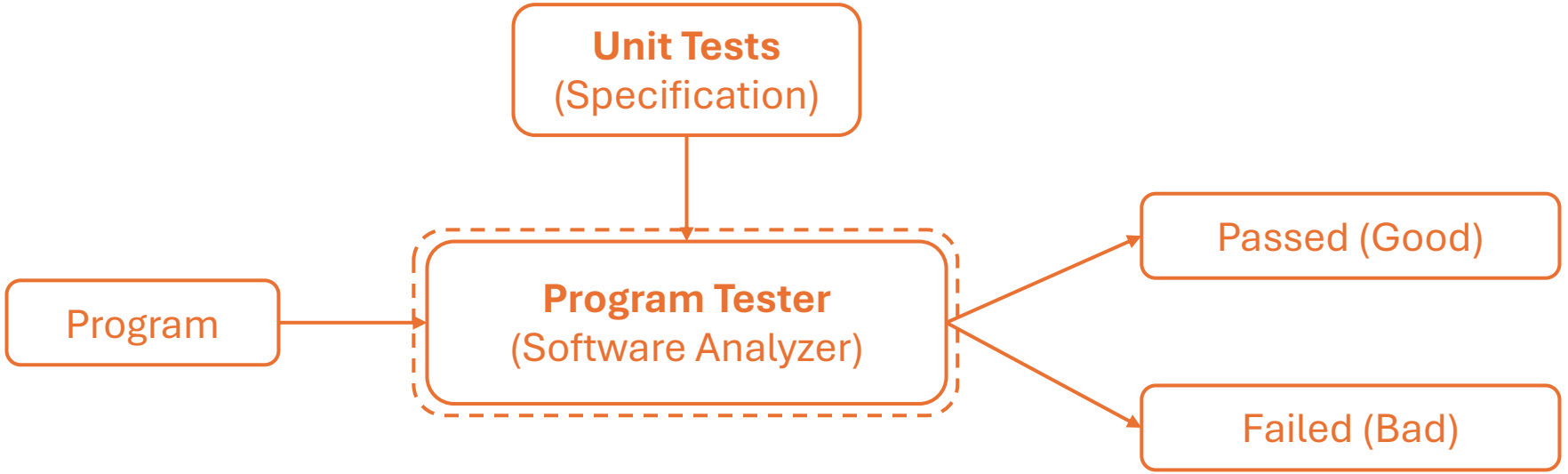
Software Analysis: Testing



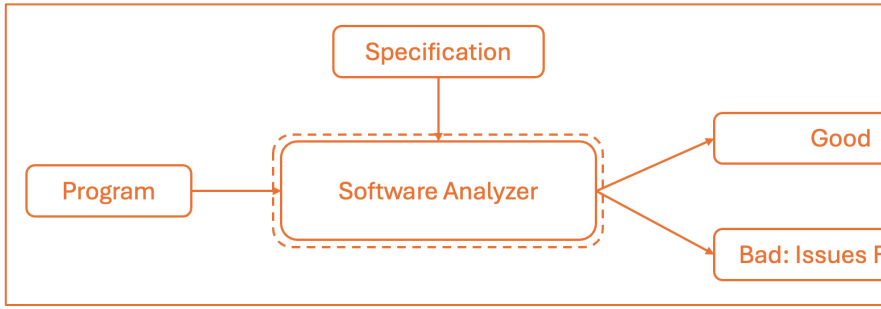


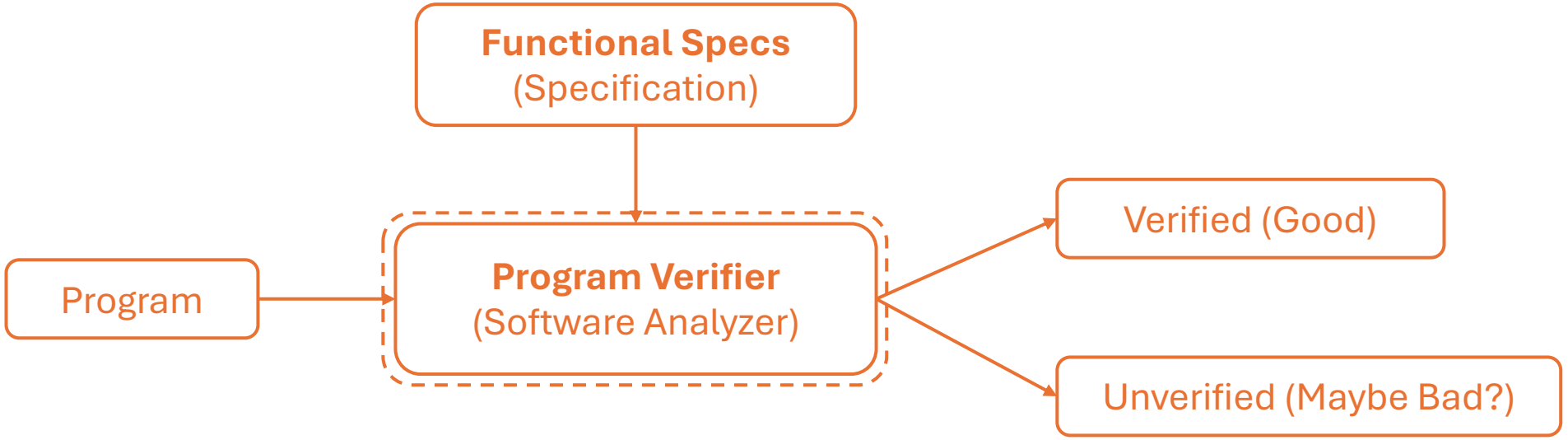
Software Analysis: Testing



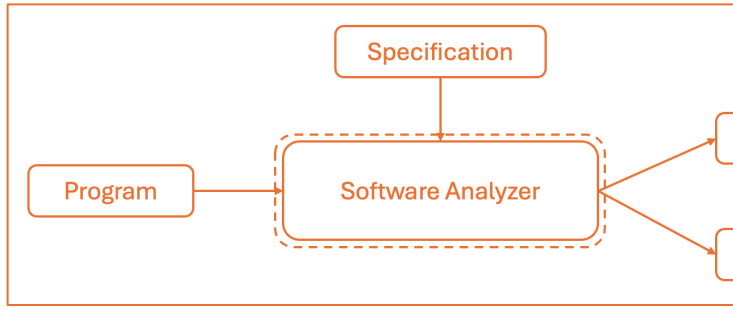


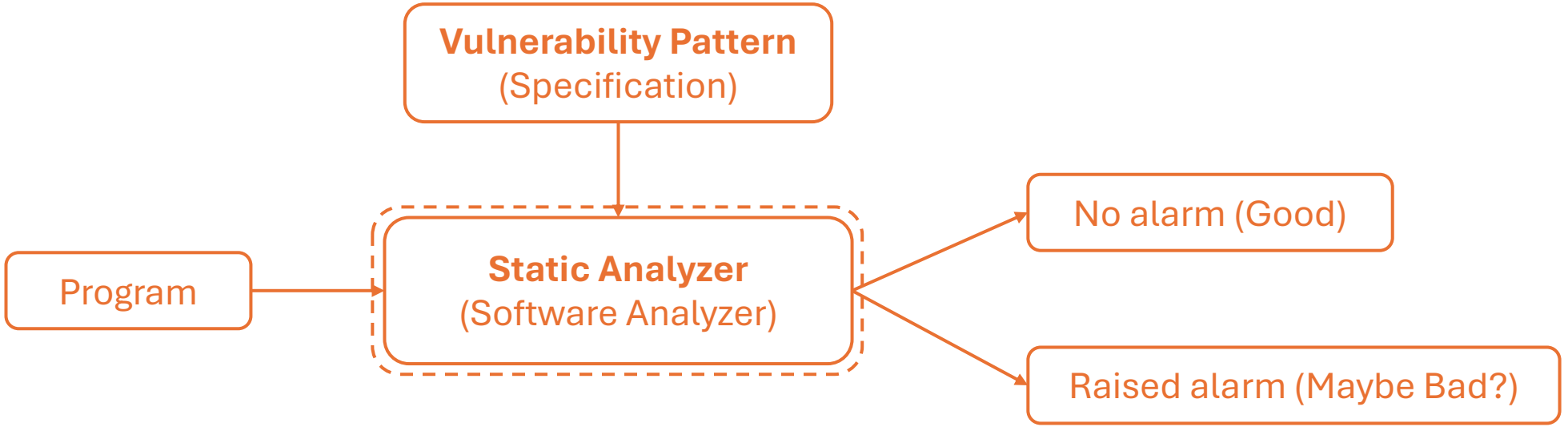
Software Analysis: Verification



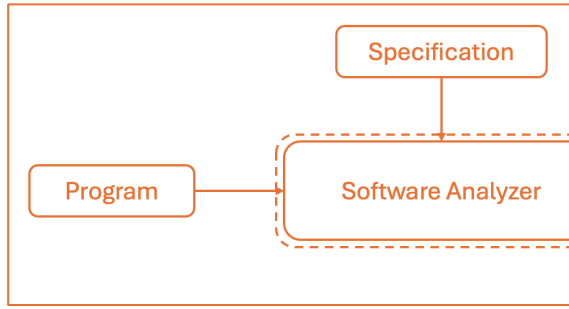


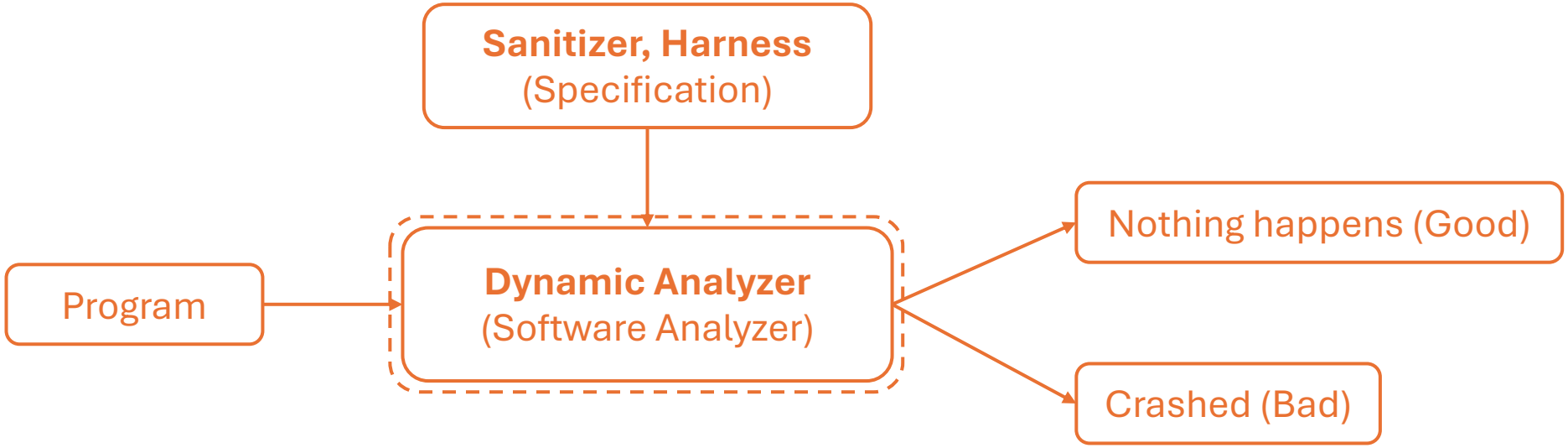
Software Analysis: Static Analysis





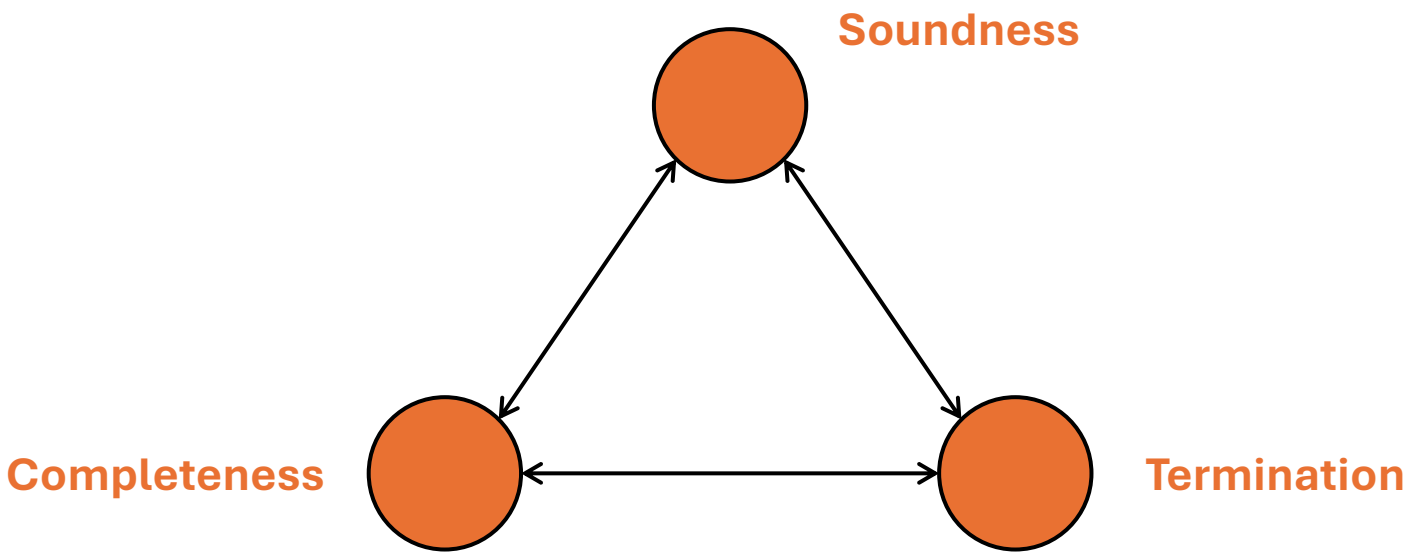
Software Analysis: Dynamic Analysis



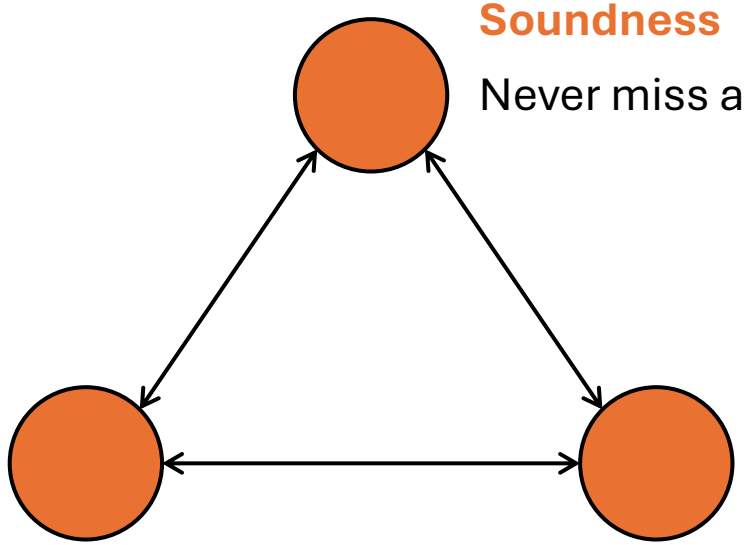


	Program is actually Bad	**Program is actually Good**
Reported Bad (positive)	True Pos **(TP)**	False Pos **(FP)**
Not Reported Bad (negative)	False Neg **(FN)**	True Neg **(TN)**

Software Analysis: Impossible Triangle



Software Analysis: Impossible Triangle



Soundness

Never miss a real bug

Completeness

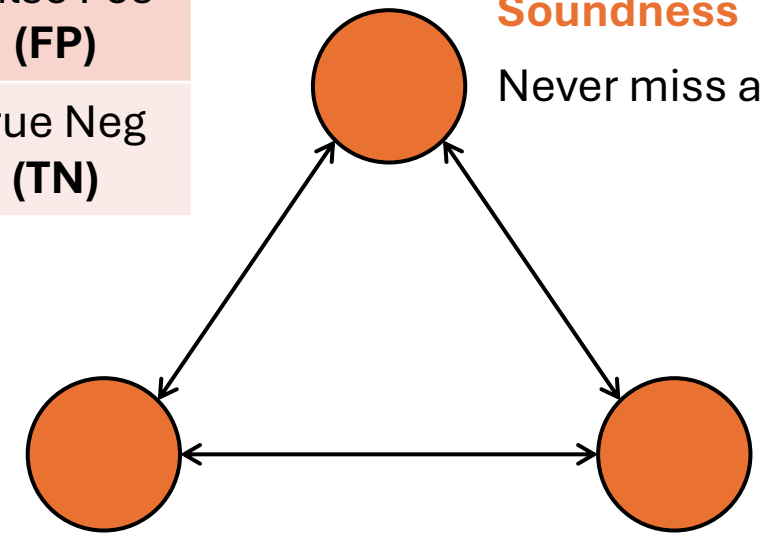
Never raise a false alarm

Termination

Always finish the analysis within finite time

Software Analysis: Impossible Triangle

	Real Bug	Not a Bug
Reported (positive)	True Pos **(TP)**	False Pos **(FP)**
Not Reported (negative)	False Neg **(FN)**	True Neg **(TN)**



Soundness

Never miss a real bug

Completeness

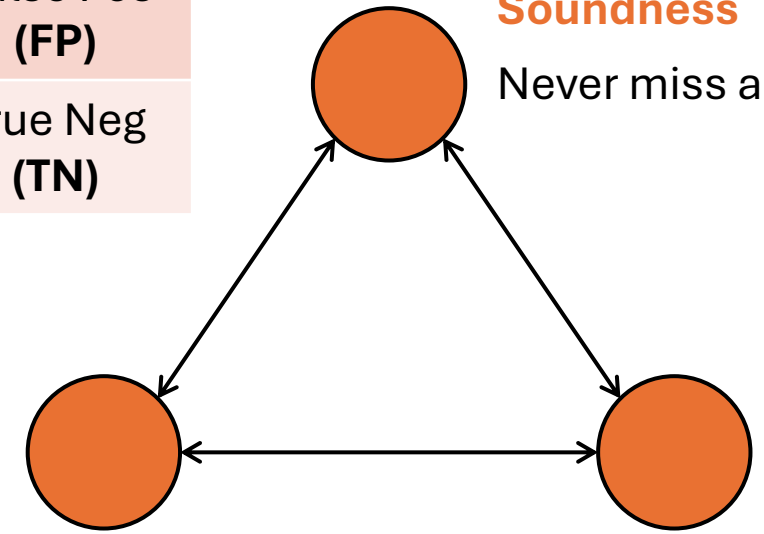
Never raise a false alarm

Termination

Always finish the analysis within finite time

Software Analysis: Impossible Triangle

	Real Bug	Not a Bug
Reported (positive)	True Pos **(TP)**	False Pos **(FP)**
Not Reported (negative)	False Neg **(FN)**	True Neg **(TN)**



Soundness #FN = 0

Never miss a real bug

#FP = 0 **Completeness**

Never raise a false alarm

Termination

Always finish the analysis within finite time

Software Analysis: Impossible Triangle

Dynamic Analysis:

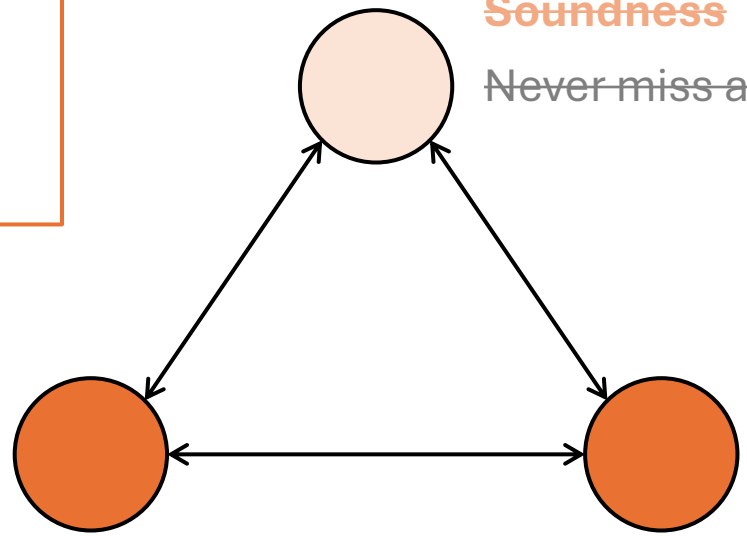
Unit Testing

- (assuming tests are correct)

Fuzzing

- (assuming harness is real)

The unit test cases might not be comprehensive enough.



~~**Soundness**~~

~~Never miss a real bug~~

Completeness

Never raise a false alarm

Termination

Always finish the analysis within finite time

Software Analysis: Impossible Triangle

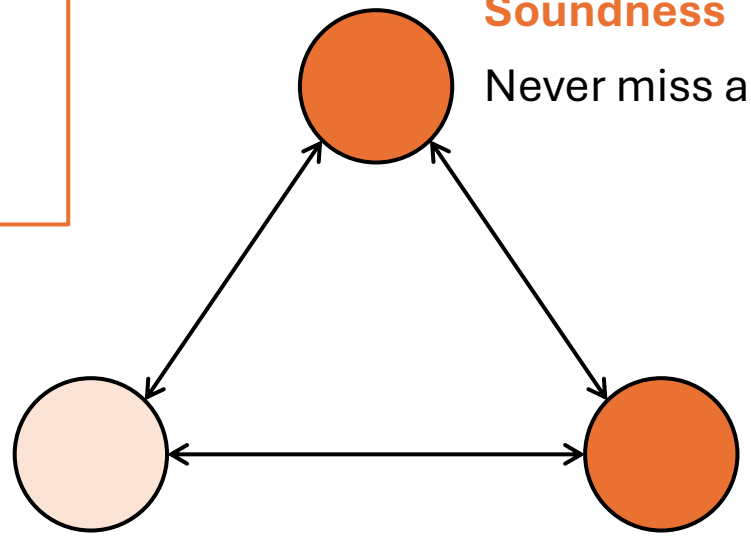
Static Analysis:

Taint Analysis

- (assuming correct specs)

Abstract Interpretation

- (assuming correct domains)



Soundness

Never miss a real bug

~~Completeness~~

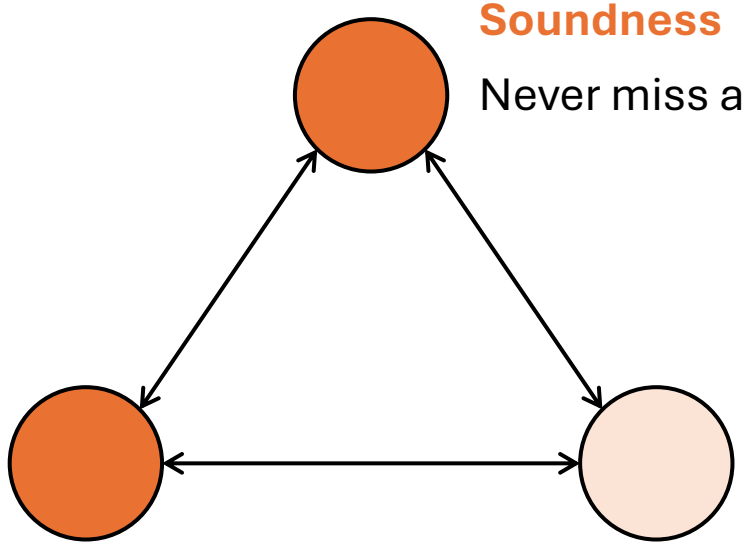
~~Never raise a false alarm~~

Termination

Always finish the analysis within finite time

May report an alarm for developer confirmation, and that is rejected

Software Analysis: Impossible Triangle



Soundness

Never miss a real bug

Completeness

Never raise a false alarm

~~**Termination**~~

~~Always finish the analysis within finite time~~

Any judgement is precise and accurate, but may not terminate

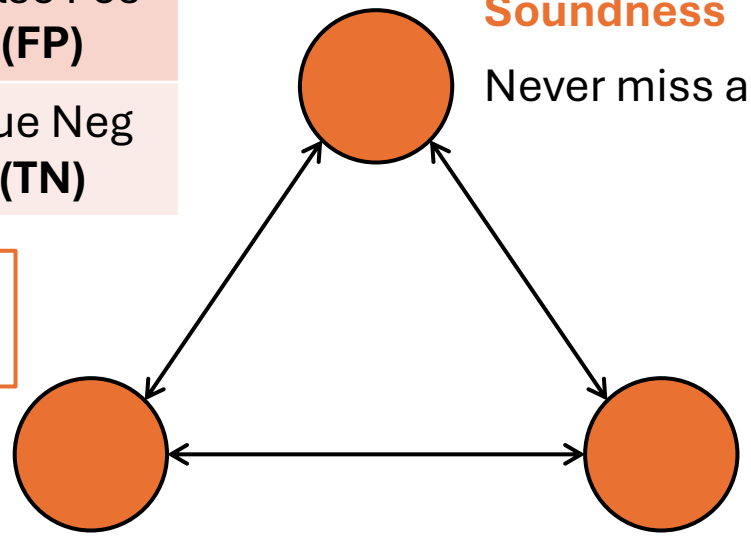
Formal Verification:

Automated Theorem Proving

Interactive Theorem Proving

Software Analysis: Impossible Triangle

	Real Bug	Not a Bug
Reported (positive)	True Pos **(TP)**	False Pos **(FP)**
Not Reported (negative)	False Neg **(FN)**	True Neg **(TN)**



Soundness #FN = 0

Never miss a real bug

#FN is small: Analysis achieves high **"Recall"**

#FP is small: Analysis achieves high **"Precision"**

#FP = 0 **Completeness**

Never raise a false alarm

Termination

Always finish the analysis within finite time

Forms of Software Analysis

Static Analysis

Dynamic Analysis

Forms of Software Analysis

Static Analysis

Examine code without concrete execution

Dynamic Analysis

Observes program behavior while executing

Forms of Software Analysis

Static Analysis

Examine code without concrete execution

- Taint analysis
- Reachability analysis
- Abstract interpretation
- Symbolic execution
- …

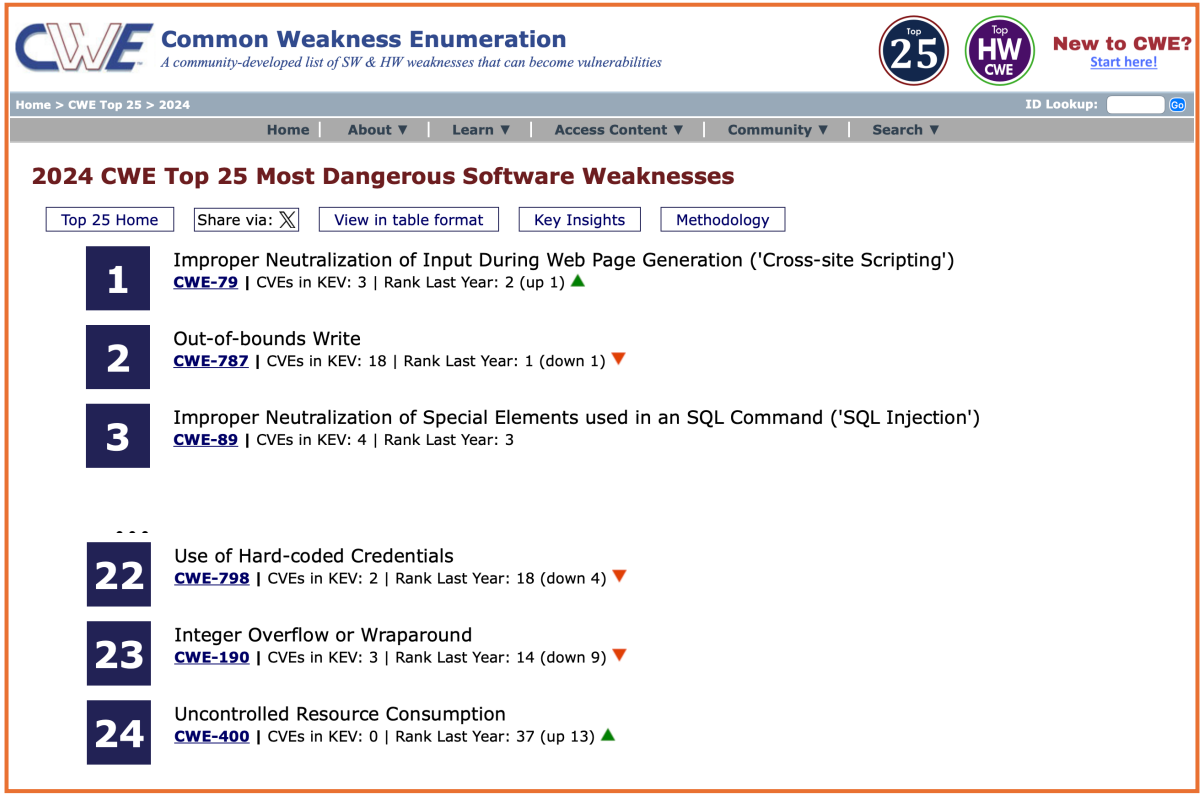
Dynamic Analysis

Observes program behavior while executing

- Unit testing
- Fuzzing
- Property-based testing
- Penetration testing
- …

Static Analysis for Vulnerability Detection

- To be able to detect software vulnerabilities, one first need to understand vulnerabilities:



Path Traversal (CWE-22)

Common Weakness Enumeration
A community-developed list of SW & HW weaknesses that can become vulnerabilities

Top 25 | Top HW CWE | New to CWE? Start here!

Home > CWE List > CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') (4.18) ID Lookup: Go

Home | About ▼ | Learn ▼ | Access Content ▼ | Community ▼ | Search ▼

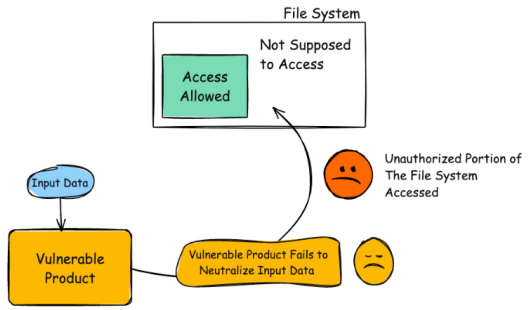
CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Weakness ID: 22
Vulnerability Mapping: ALLOWED
Abstraction: Base

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the product does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.



Extended Description

Many file operations are intended to take place within a restricted directory. By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system. One of the most common special elements is the "../" sequence, which in most modern operating systems is interpreted as the parent directory of the current location. This is referred to as relative path traversal. Path traversal also covers the use of absolute pathnames such as "/usr/local/bin" to access unexpected files. This is referred to as absolute path traversal.

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFileServlet extends HttpServlet {
  private static final String BASE_DIR = "/var/www/public-files";

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFileServlet extends HttpServlet {
  private static final String BASE_DIR = "/var/www/public-files";

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

GET /download?path=css/basic.css

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); //
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

GET /download?path=css/basic.css

css/basic.css

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); //
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

GET /download?path=css/basic.css

css/basic.css

/var/www/public-files/css/basic.css

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); //
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

GET /download?path=css/basic.css

css/basic.css

/var/www/public-files/css/basic.css

Reading File

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Malicious data

GET /download?path=../SECRET.txt

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); //
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Malicious data

GET /download?path=../SECRET.txt

../SECRET.txt

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Malicious data

GET /download?path=../SECRET.txt

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); //
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "att
      try (InputStream in = new FileInputStream(
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

../SECRET.txt

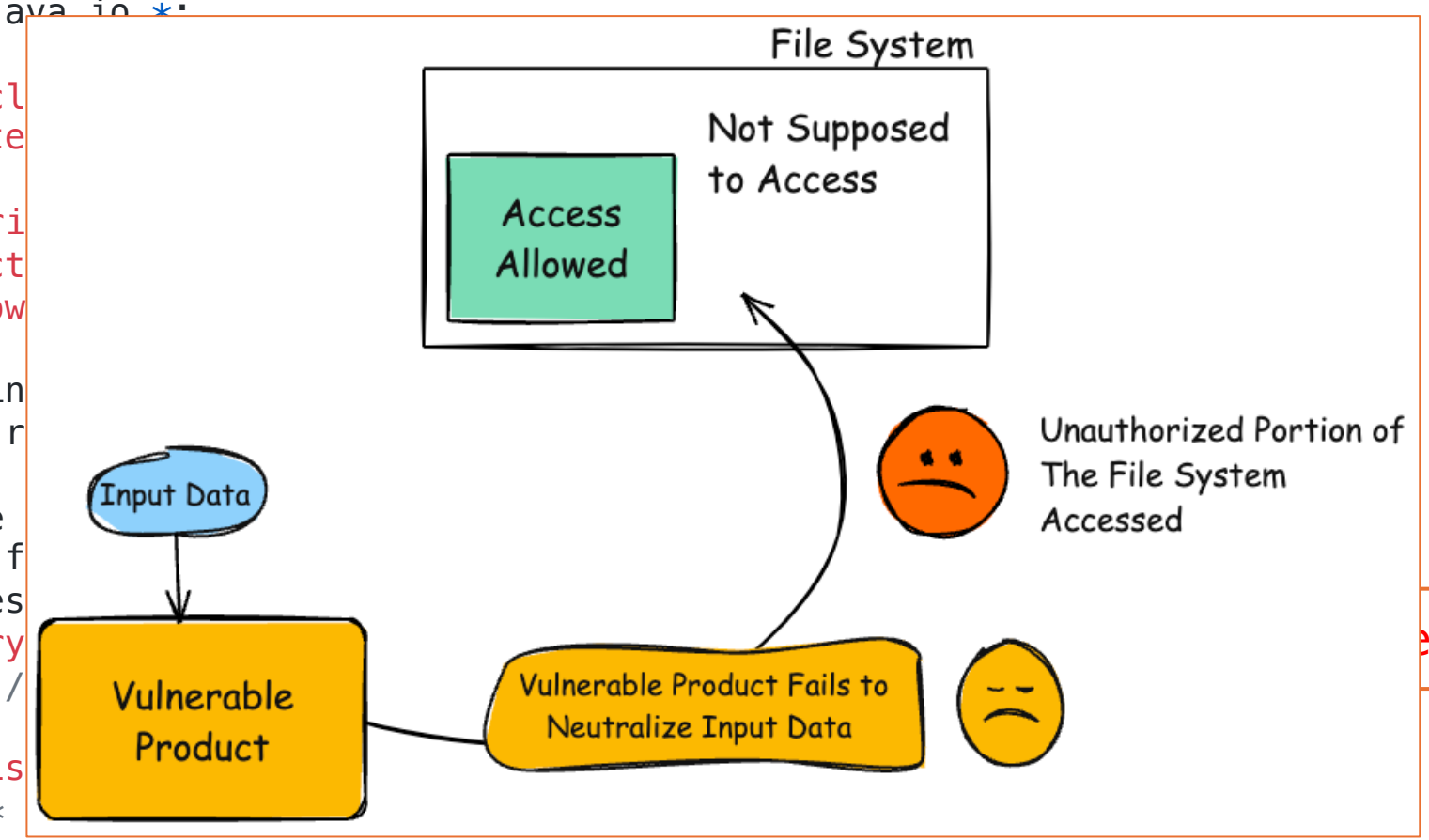
/var/www/public-files/../SECRET.txt

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public cl
  private

  @Overri
  protect
    throw
  {
    Strin
    if (r

    File
    if (f
      res
      try
        /
      }
    } els
      /*
    }
  }
}
```




```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```
es/../SECRET.txt
```

Fix Attempt #1

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Malicious data

GET /download?path=../SECRET.txt

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

+   if (requested.contains("..") || requested.contains("\0")) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Fix Attempt #1

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"
    if (requested == null || requested.isEmpty

+   if (requested.contains("..") || requested.contains("\0")) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Malicious data

GET /download?path=../SECRET.txt

Malicious data blocked by “..” check

Fix Attempt #1

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Benign data

GET /download?path=css/../index.html

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

+   if (requested.contains("..") || requested.contains("\0")) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Fix Attempt #1

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Benign data

GET /download?path=css/../index.html

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /*

+   if (requested.contains("..") || requested.contains("\0")) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Benign input blocked by “..” check

Fix Attempt #2

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (candidate.exists() && candidate.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + candidate.getName() + "\"");
      try (InputStream in = new FileInputStream(candidate); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Fix Attempt #2

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                              es)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (candidate.exists() && candidate.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + candidate.getName() + "\"");
      try (InputStream in = new FileInputStream(candidate); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

GET /download?path=css/../index.html

Fix Attempt #2

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

GET /download?path=css/../index.html

/var/www/public-files/index.html

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                    es)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /*

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (candidate.exists() && candidate.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + candidate.getName() + "\"");
      try (InputStream in = new FileInputStream(candidate); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Fix Attempt #2

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                          es)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /*

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (ca
      resp
      try                                                    {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

GET /download?path=css/../index.html

/var/www/public-files/index.html

/var/www/public-files/index.html “startsWith” /var/www/public-files?
YES

Fix Attempt #2

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                                    s)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (candidate.exists() && candidate.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + candidate.getName() + "\"");
      try (InputStream in = new FileInputStream(candidate); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Malicious data

```
GET /download?path=../SECRET.txt
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Fix Attempt #2

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                                    s)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /*

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (candidate.exists() && candidate.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + candidate.getName() + "\"");
      try (InputStream in = new FileInputStream(candidate); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Malicious data

GET /download?path=../SECRET.txt

/var/www/SECRET.txt

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Fix Attempt #2

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void do                                              s)
    throws ServletE
  {
    String requested = req.getParameter("path"); // e.g. "reports/2024.pdf"
    if (requested == null || requested.isEmpty()) { /*

+   Path base = Paths.get("/var/www/public-files");
+   Path candidate = baseReal.resolve(requested).normalize();
+   if (!candidate.startsWith(baseReal)) { /* ERROR */ }

    if (ca
      resp
      try                                                        {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Malicious data

GET /download?path=../SECRET.txt

/var/www/SECRET.txt

/var/www/SECRET.txt “startsWith” /var/www/public-files?
NO

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Path Traversal: Key Takeaway

- Analysis of Path Traversal (CWE-22)
  - Insufficient check (sanitization) allows access to sensitive information (e.g., information leak)
  - Variants: allows attacker to create files at dangerous locations (e.g., inject a script at a place like /usr/local/bin which can be executed later)
- Defense against Path Traversal (CWE-22)
  - Code should be structured to "resolve" and "normalize" paths
  - Just checking ".." is insufficient or may reject benign input
- Detection of Path Traversal (CWE-22)
  - Tainted input: malicious file paths
  - Attack surface: internet http request, untrusted files (e.g., Zip)
  - Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
  - Absence of Sanitization: no "resolve" or "normalize" from tainted input to manifestation

Path Traversal: Detection

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else { /* ERROR */ }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Path Traversal: Detection

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```java
public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else { /* ERROR */ }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Path Traversal: Detection

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else { /* ERROR */ }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Path Traversal: Detection

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```java
public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else { /* ERROR */ }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Path Traversal: Detection

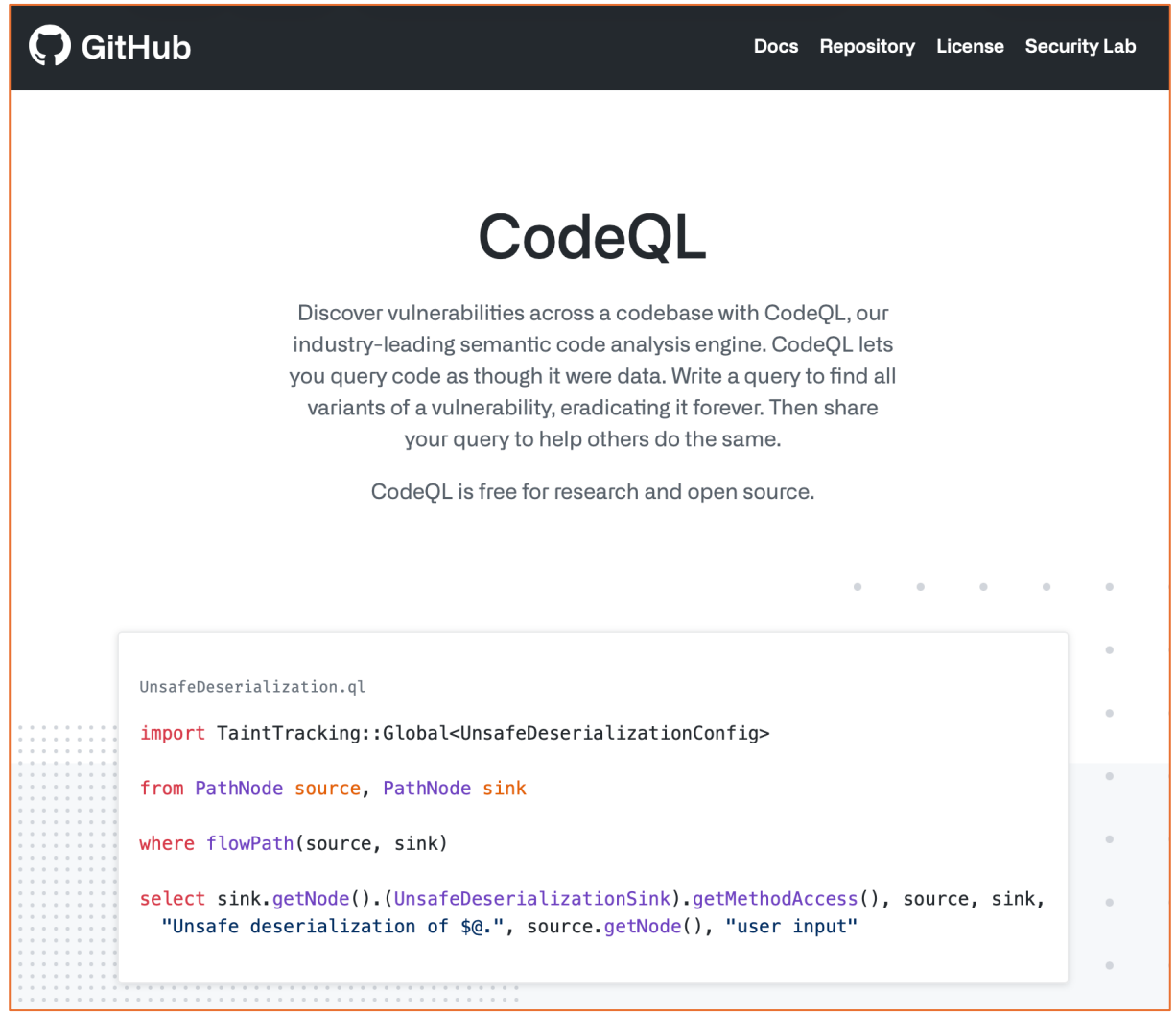
- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```java
public class VulnerableFileServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) {

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else { /* ERROR */ }
  }
}
```

Note: File(…) is not actually accessing the file

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

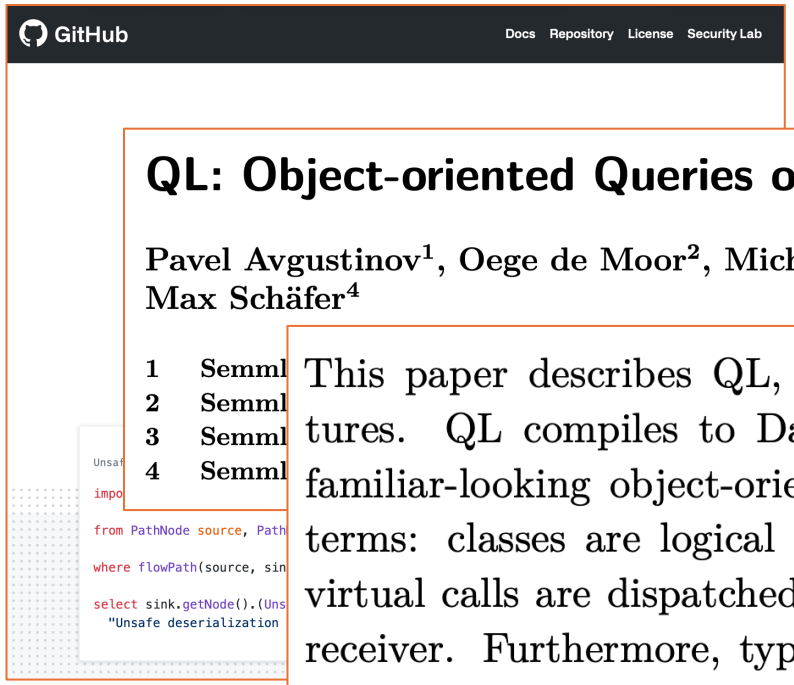

GitHub
Docs
Repository
License
Security Lab
CodeQL
Discover vulnerabilities across a codebase with CodeQL, our industry-leading semantic code analysis engine. CodeQL lets you query code as though it were data. Write a query to find all variants of a vulnerability, eradicating it forever. Then share your query to help others do the same.
CodeQL is free for research and open source.
UnsafeDeserialization.ql
import TaintTracking::Global<UnsafeDeserializationConfig>
from PathNode source, PathNode sink
where flowPath(source, sink)
select sink.getNode().(UnsafeDeserializationSink).getMethodAccess(), source, sink,
"Unsafe deserialization of $@.", source.getNode(), "user input"



QL: Object-oriented Queries on Relational Data

Pavel Avgustinov[1], Oege de Moor[2], Michael Peyton Jones[3], and Max Schäfer[4]

1	Semmle Ltd	publications@semmle.com
2	Semmle Ltd	publications@semmle.com
3	Semmle Ltd	publications@semmle.com
4	Semmle Ltd	publications@semmle.com

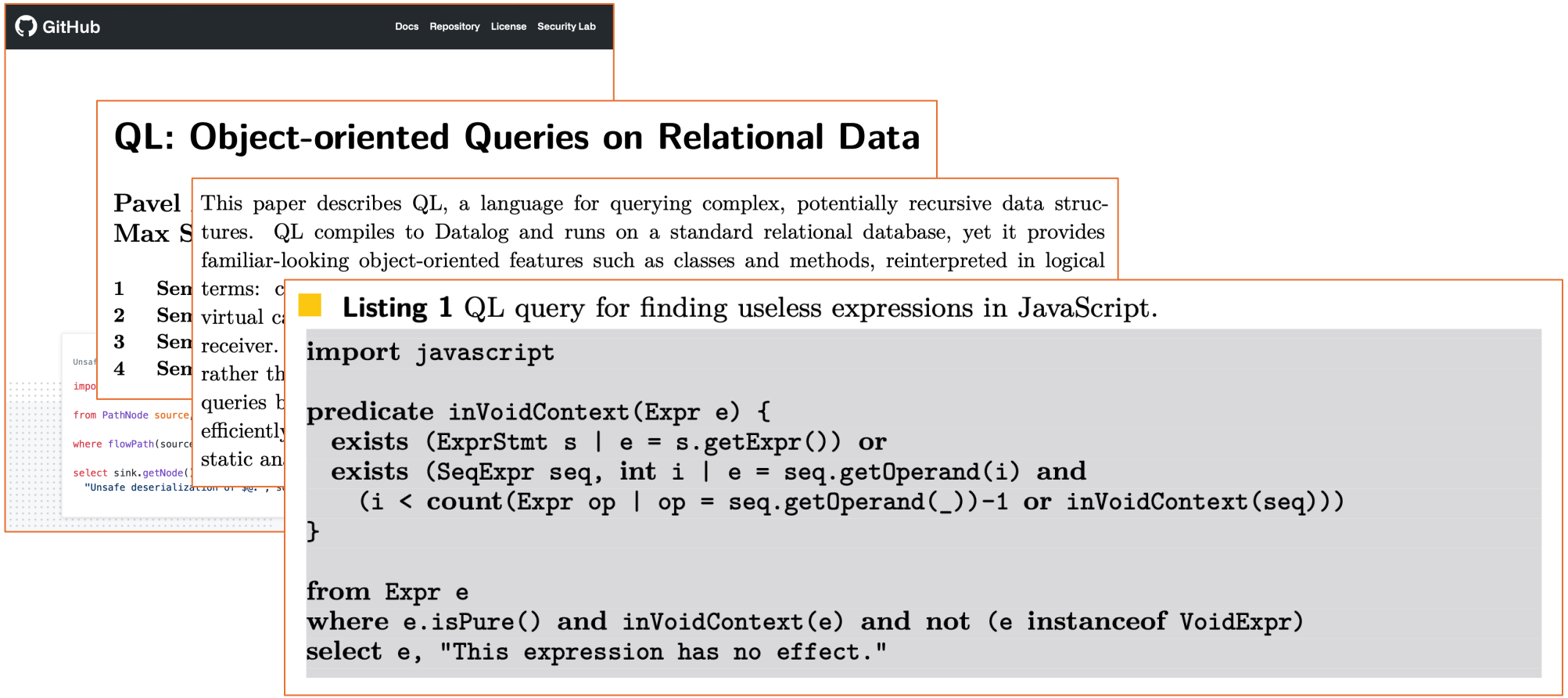


QL: Object-oriented Queries on Relational Data

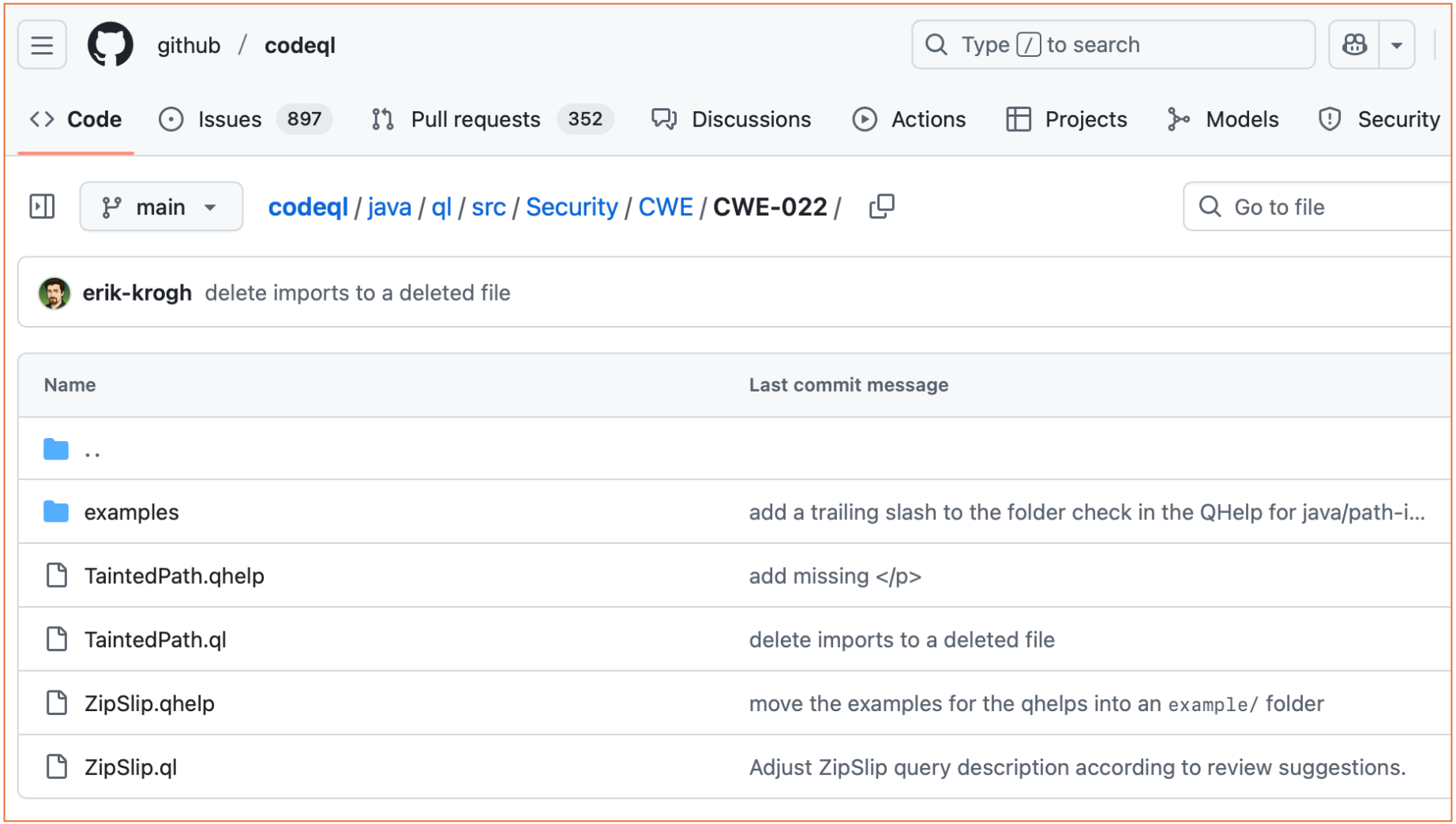
Pavel Avgustinov[1], Oege de Moor[2], Michael Peyton Jones[3], and Max Schäfer[4]

1 Semml
2 Semml
3 Semml
4 Semml

This paper describes QL, a language for querying complex, potentially recursive data structures. QL compiles to Datalog and runs on a standard relational database, yet it provides familiar-looking object-oriented features such as classes and methods, reinterpreted in logical terms: classes are logical properties describing sets of values, subclassing is implication, and virtual calls are dispatched dynamically by considering the most specific classes containing the receiver. Furthermore, types in QL are prescriptive and actively influence program evaluation rather than just describing it. In combination, these features enable the development of concise queries based on reusable libraries, which are written in a purely declarative style, yet can be efficiently executed even on very large data sets. In particular, we have used QL to implement static analyses for various programming languages, which scale to millions of lines of code.


GitHub
Docs Repository License Security Lab
QL: Object-oriented Queries on Relational Data
Pavel
Max S
1 Sen
2 Sen
3 Sen
4 Sen
This paper describes QL, a language for querying complex, potentially recursive data structures. QL compiles to Datalog and runs on a standard relational database, yet it provides familiar-looking object-oriented features such as classes and methods, reinterpreted in logical terms: c
virtual c
receiver.
rather th
queries b
efficientl
static an
Listing 1 QL query for finding useless expressions in JavaScript.
import javascript
predicate inVoidContext(Expr e) {
exists (ExprStmt s | e = s.getExpr()) or
exists (SeqExpr seq, int i | e = seq.getOperand(i) and
(i < count(Expr op | op = seq.getOperand(_))-1 or inVoidContext(seq)))
}
from Expr e
where e.isPure() and inVoidContext(e) and not (e instanceof VoidExpr)
select e, "This expression has no effect."

CodeQL for Path Traversal (CWE-22)



```
/**
 * @name Uncontrolled data used in path expression
 * @description Accessing paths influenced by users can allow an attacker to access unexpected resources.
 * @kind path-problem
 * @problem.severity error
 * @security-severity 7.5
 * @precision high
 * @id java/path-injection
 * @tags security
 *       external/cwe/cwe-022
 *       external/cwe/cwe-023
 *       external/cwe/cwe-036
 *       external/cwe/cwe-073
 */

import java
import semmle.code.java.security.TaintedPathQuery
import TaintedPathFlow::PathGraph

from TaintedPathFlow::PathNode source, TaintedPathFlow::PathNode sink
where TaintedPathFlow::flowPath(source, sink)
select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
  "user-provided value"
```

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instanceof PathInjectionSanitizer
  }

  predicate isAdditionalFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
    any(TaintedPathAdditionalTaintStep s).step(n1, n2)
  }

  predicate observeDiffInformedIncrementalMode() { any() }
}

/** Tracks flow from remote sources to the creation of a path. */
module TaintedPathFlow = TaintTracking::Global<TaintedPathConfig>;
```

```
1   /**
2    * @name Uncontrolled data used in path
3    * @description Accessing paths influenc
4    * @kind path-problem
5    * @problem.severity error
6    * @security-severity 7.5
7    * @precision high
8    * @id java/path-injection
9    * @tags security
10   *       external/cwe/cwe-022
11   *       external/cwe/cwe-023
12   *       external/cwe/cwe-036
13   *       external/cwe/cwe-073
14   */
15
16  import java
17  import semmle.code.java.security.TaintedPathQuery
18  import TaintedPathFlow::PathGraph
19
20  from TaintedPathFlow::PathNode source, TaintedPathFlow::PathNode sink
21  where TaintedPathFlow::flowPath(source, sink)
22  select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
23    "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instanceof PathInjectionSanitizer
  }

  predicate isAdditionalFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
    any(TaintedPathAdditionalTaintStep s).step(n1, n2)
  }

  predicate observeDiffInformedIncrementalMode() { any() }
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
1   /**
2    * @name Uncontrolled data used in path
3    * @description Accessing paths influenc
4    * @kind path-problem
5    * @problem.severity error
6    * @security-severity 7.5
7    * @precision high
8    * @id java/path-injection
9    * @tags security
10   *       external/cwe/cwe-022
11   *       external/cwe/cwe-023
12   *       external/cwe/cwe-036
13   *       external/cwe/cwe-073
14
15
16
17
18
19
20
21   where TaintedPathFlow::flowPath(source, sink)
22   select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
23     "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instanceof PathInjectionSanitizer
  }

  predicate isAdditionalFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
    any(TaintedPathAdditionalTaintStep s).step(n1, n2)
  }

  predicate observeDiffInformedIncrementalMode() { any() }
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
1   /**
2    * @name Uncontrolled data used in path
3    * @description Accessing paths influenc
4    * @kind path-problem
5    * @problem.severity error
6    * @security-severity 7.5
7    * @precision high
8    * @id java/path-injection
9    * @tags security
10   *       external/cwe/cwe-022
11   *       external/cwe/cwe-023
12   *       external/cwe/cwe-036
13   *       external/cwe/cwe-073
14
15
16
17
18
19
20
21   where TaintedPathFlow::flowPath(source, sink)
22   select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
23     "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instan
  }

  predicate isAdditi
    any(TaintedPathA
  }

  predicate observeD
}
```

```
extensions:
  - addsTo:
      pack: codeql/java-all
      extensible: sourceModel
    data:
      - ["javax.servlet.http", "Cookie", False, "getComment", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "Cookie", False, "getName", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "Cookie", False, "getValue", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getHeader", "(String)", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getHeaderNames", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getHeaders", "(String)", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getParameter", "(String)", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getParameterMap", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getParameterNames", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getParameterValues", "(String)", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getPathInfo", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getQueryString", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getRemoteUser", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getRequestURI", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getRequestURL", "()", "", "ReturnValue", "remote", "manual"]
      - ["javax.servlet.http", "HttpServletRequest", False, "getServletPath", "()", "", "ReturnValue", "remote", "manual"]
```

- Tainted input: malicious file paths
- Attack surface: internet http request, unt
- Vulnerability Manifestation: reading/writi
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
/**
 * @name Uncontrolled data used in path
 * @description Accessing paths influenc
 * @kind path-problem
 * @problem.severity error
 * @security-severity 7.5
 * @precision high
 * @id java/path-injection
 * @tags security
 *       external/cwe/cwe-022
 *       external/cwe/cwe-023
 *       external/cwe/cwe-036
 *       external/cwe/cwe-073
 */

import java
import semmle.code.java.dataflow.FlowSources
import TaintedPathFlow::PathGraph

from TaintedPathFlow::PathNode source, TaintedPathFlow::PathNode sink
where TaintedPathFlow::flowPath(source, sink)
select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
  "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instanceof PathInjectionSanitizer
  }

  predicate isAdditionalFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
    any(TaintedPathAdditionalTaintStep s).step(n1, n2)
  }

  predicate observeDiffInformedIncrementalMode() { any() }
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
1   /**
2    * @name Uncontrolled data used in path
3    * @description Accessing paths influenc
4    * @kind path-problem
5    * @problem.severity error
6    * @security-severity 7.5
7    * @precision high
8    * @id java/path-injection
9    * @tags security
10   *       external/cwe/cwe-022
11   *       external/cwe/cwe-023
12   *       external/cwe/cwe-036
13   *       external/cwe/cwe-073
14
15
16
17
18
19
20
21   where TaintedPathFlow::flowPath(source, sink)
22   select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
23     "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }
```

```
extensions:
  - addsTo:
      pack: codeql/java-all
      extensible: sinkModel
    data:
      - ["java.io", "File", True, "canExecute", "()", "", "Argument[this]", "path-injection", "manual"]
      - ["java.io", "File", True, "canRead", "()", "", "Argument[this]", "path-injection", "manual"]
      - ["java.io", "File", True, "canWrite", "()", "", "Argument[this]", "path-injection", "manual"]
      - ["java.io", "File", True, "createNewFile", "()", "", "Argument[this]", "path-injection", "ai-manual"]
      - ["java.io", "File", True, "createTempFile", "(String,String,File)", "", "Argument[2]", "path-injection", "ai-manual"]
- ["java.io", "FileInputStream", True, "FileInputStream", "(File)", "", "Argument[0]", "path-injection", "ai-manual"]
- ["java.io", "FileInputStream", True, "FileInputStream", "(FileDescriptor)", "", "Argument[0]", "path-injection", "manual"]
- ["java.io", "FileInputStream", True, "FileInputStream", "(String)", "", "Argument[0]", "path-injection", "ai-manual"]
- ["java.io", "FileOutputStream", False, "FileOutputStream", "", "", "Argument[0]", "path-injection", "manual"]
- ["java.io", "FileOutputStream", False, "write", "", "", "Argument[0]", "file-content-store", "manual"]
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
 *       external/cwe/cwe-073
 */
import java
import semmle.code.java.dataflow.FlowSources
import TaintedPathFlow::PathGraph

from TaintedPathFlow::PathNode source, TaintedPathFlow::PathNode sink
where TaintedPathFlow::flowPath(source, sink)
select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
  "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

codeql/java/ql/lib/semmle/code/java/security/TaintedPathQuery.qll

```
/**
 * A taint-tracking configuration for tracking flow from remote sources to the creation of a path.
 */
module TaintedPathConfig implements DataFlow::ConfigSig {
  predicate isSource(DataFlow::Node source) { source instanceof ActiveThreatModelSource }

  predicate isSink(DataFlow::Node sink) { sink instanceof TaintedPathSink }

  predicate isBarrier(DataFlow::Node sanitizer) {
    sanitizer instanceof SimpleTypeSanitizer or
    sanitizer instanceof PathInjectionSanitizer
  }

  predicate isAdditionalFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
    any(TaintedPathAdditionalTaintStep s).step(n1, n2)
  }

  predicate observeDiffInformedIncrementalMode() { any() }
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
1   /**
2    * @name Uncontrolled data used in path
3    * @description Accessing paths influenc
4    * @kind path-problem
5    * @problem.severity error
6    * @security-severity 7.5
7    * @precision high
8    * @id java/path-injection
9    * @tags security
10   *       external/cwe/cwe-022
11   *       external/cwe/cwe-023
12   *       external/cwe/cwe-036
13   *       external/cwe/cwe-073
14
15
16
17
18
19
20
21   where TaintedPathFlow::flowPath(source, sink)
22   select sink.getNode(), source, sink, "This path depends on a $@.", source.getNode(),
23     "user-provided value"
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

```
predicate isBarrier(DataFlow::Node sanitizer) {
  sanitizer instanceof SimpleTypeSanitizer or
  sanitizer instanceof PathInjectionSanitizer
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

```
/**
 * A sanitizer that protects against path injection vulnerabilities
 * by checking for a matching path.
 */
class ExactPathMatchSanitizer extends PathInjectionSanitizer {
  ExactPathMatchSanitizer() {
    this = DataFlow::BarrierGuard<exactPathMatchGuard/3>::getABarrierNode()
  }
}
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

```
predicate isBarrier(DataFlow::Node sanitizer) {
  sanitizer instanceof SimpleTypeSanitizer or
  sanitizer instanceof PathInjectionSanitizer
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

One of the sanitizers: Exact Path Match

```
/**
 * A sanitizer that protects against path injection vulnerabilities
 * by checking for a matching path.
 */
class ExactPathMatchSanitizer extends PathInjectionSanitizer {
  ExactPathMatchSanitizer() {
    this = DataFlow::BarrierGuard<exactPathMatchGuard/3>::getABarrierNode()
  }
}
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

```
predicate isBarrier(DataFlow::Node sanitizer) {
  sanitizer instanceof SimpleTypeSanitizer or
  sanitizer instanceof PathInjectionSanitizer
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

Another one of the sanitizers: Normalization + Prefix guard (if-condition)

```
private predicate allowedPrefixGuard(Guard g, Expr e, boolean branch) {
  branch = true and
  // Local taint-flow is used here to handle cases where the validated expression comes from the
  // expression reaching the sink, but it's not the same one, e.g.:
  //  File file = source();
  //  String strPath = file.getCanonicalPath();
  //  if (strPath.startsWith("/safe/dir"))
  //    sink(file);
  g instanceof AllowedPrefixGuard and
  localTaintFlowToPathGuard(e, g) and
  exists(Expr previousGuard |
    localTaintFlowToPathGuard(previousGuard.(PathNormalizeSanitizer), g)
    or
    previousGuard
        .(PathTraversalGuard)
        .controls(g.getBasicBlock(), previousGuard.(PathTraversalGuard).getBranch())
  )
}
```

```
predicate isBarrier(DataFlow::Node sanitizer) {
  sanitizer instanceof SimpleTypeSanitizer or
  sanitizer instanceof PathInjectionSanitizer
}
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation

Another one of the sanitizers: dot-dot (..) check guard (if-condition)

```
private predicate dotDotCheckGuard(Guard g, Expr e, boolean branch) {
  pathTraversalGuard(g, e, branch) and
  exists(Guard previousGuard |
    previousGuard.(AllowedPrefixGuard).controls(g.getBasicBlock(), true)
    or
    previousGuard.(BlockListGuard).controls(g.getBasicBlock(), false)
  )
}

private class DotDotCheckSanitizer extends PathInjectionSanitizer {
  DotDotCheckSanitizer() { this = DataFlow::BarrierGuard<dotDotCheckGuard/3>::getABarrierNode() }
}
```

codeql/java/ql/src/Security/CWE/CWE-022/TaintedPath.ql

Published as a conference paper at ICLR 2025

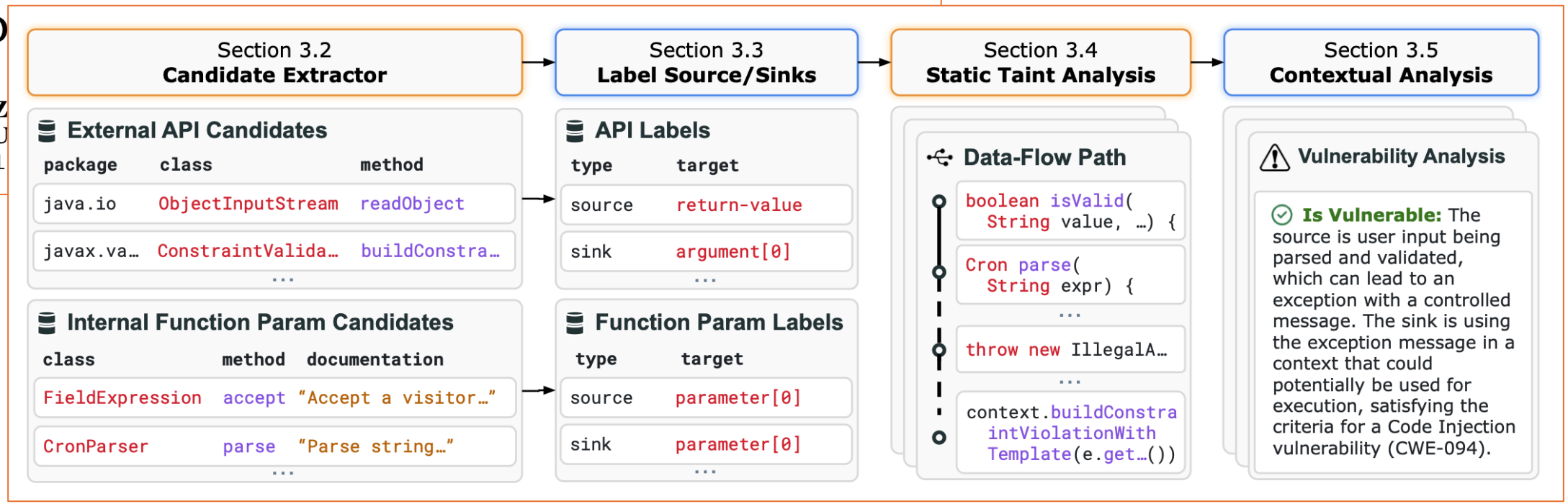
IRIS: LLM-ASSISTED STATIC ANALYSIS FOR DETECTING SECURITY VULNERABILITIES

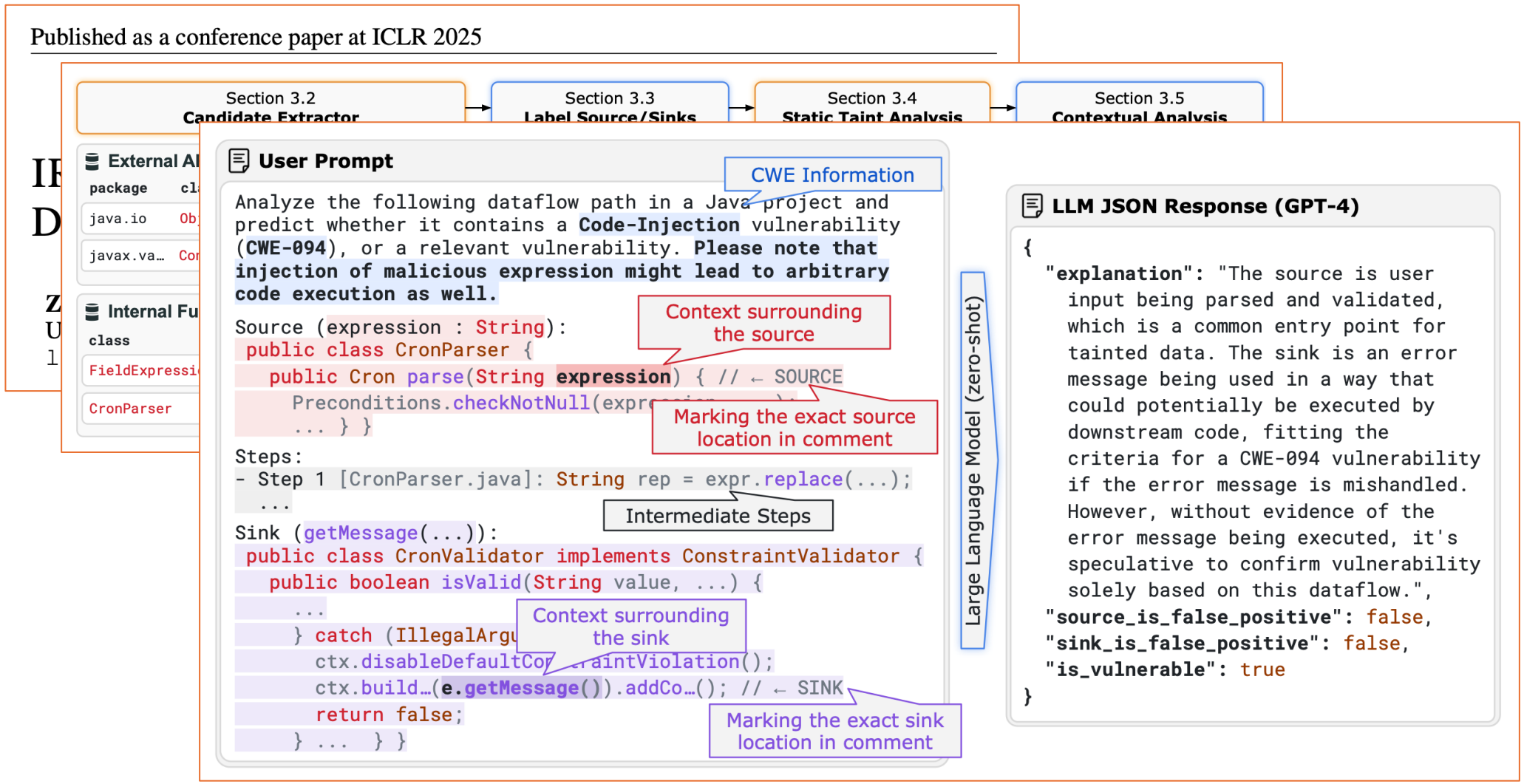
Ziyang Li
University of Pennsylvania
liby99@cis.upenn.edu

Saikat Dutta
Cornell University
saikatd@cornell.edu

Mayur Naik
University of Pennsylvania
mhnaik@cis.upenn.edu

IRIS: LLM-ASSISTED STATIC ANALYSIS FOR D




Section 3.2
Candidate Extractor
Section 3.3
Label Source/Sinks
Section 3.4
Static Taint Analysis
Section 3.5
Contextual Analysis
External Al
package
java.io
javax.va…
Internal Fu
class
FieldExpressi
CronParser
User Prompt
CWE Information
Analyze the following dataflow path in a Java project and predict whether it contains a Code-Injection vulnerability (CWE-094), or a relevant vulnerability. Please note that injection of malicious expression might lead to arbitrary code execution as well.
Source (expression : String):
public class CronParser {
public Cron parse(String expression) { // ← SOURCE
Preconditions.checkNotNull(
... } }
Context surrounding the source
Marking the exact source location in comment
Steps:
- Step 1 [CronParser.java]: String rep = expr.replace(...);
...
Intermediate Steps
Sink (getMessage(...)):
public class CronValidator implements ConstraintValidator {
public boolean isValid(String value, ...) {
...
} catch (IllegalArgu
ctx.disableDefaultCo
ctx.build…(e.getMessage()).addCo…(); // ← SINK
return false;
} ... } }
Context surrounding the sink
Marking the exact sink location in comment
Large Language Model (zero-shot)
LLM JSON Response (GPT-4)
{
"explanation": "The source is user input being parsed and validated, which is a common entry point for tainted data. The sink is an error message being used in a way that could potentially be executed by downstream code, fitting the criteria for a CWE-094 vulnerability if the error message is mishandled. However, without evidence of the error message being executed, it's speculative to confirm vulnerability solely based on this dataflow.",
"source_is_false_positive": false,
"sink_is_false_positive": false,
"is_vulnerable": true
}

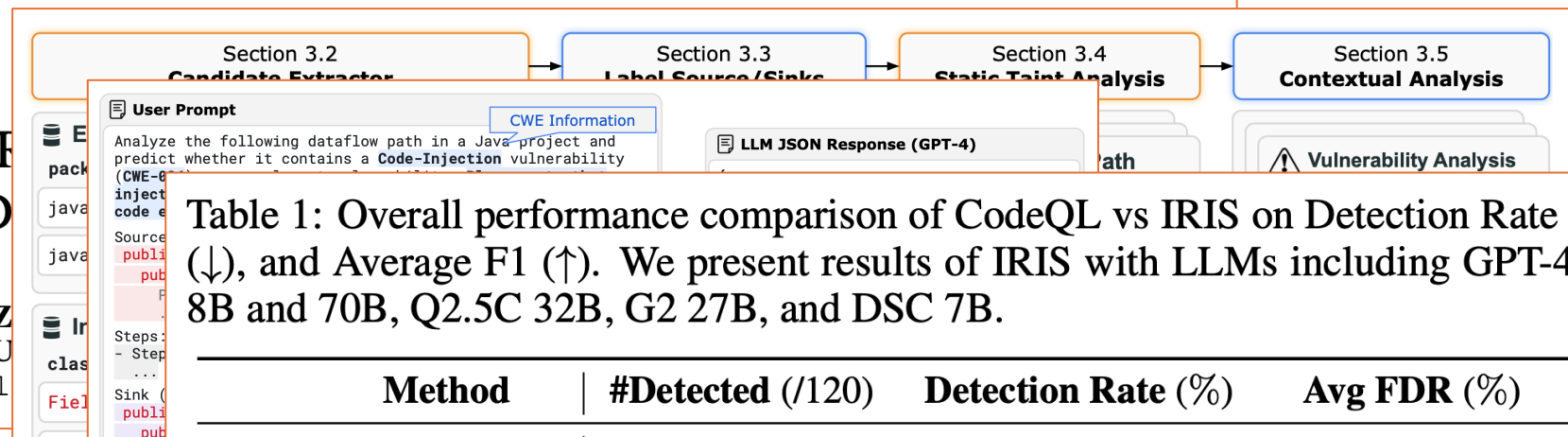


Table 1: Overall performance comparison of CodeQL vs IRIS on Detection Rate (↑), Average FDR (↓), and Average F1 (↑). We present results of IRIS with LLMs including GPT-4 and GPT-3.5, L3 8B and 70B, Q2.5C 32B, G2 27B, and DSC 7B.

	Method	#Detected (/120)	Detection Rate (%)	Avg FDR (%)	Avg F1 Score
	CodeQL	27	22.50	90.03	0.076
IRIS +	GPT-4	**55** (↑ 28)	**45.83** (↑ 23.33)	**84.82** (↓ 5.21)	**0.177** (↑ 0.101)
	GPT-3.5	47 (↑ 20)	39.17 (↑ 16.67)	90.42 (↑ 0.39)	0.096 (↑ 0.020)
	L3 8B	41 (↑ 14)	34.17 (↑ 11.67)	95.55 (↑ 5.52)	0.058 (↓ 0.018)
	L3 70B	54 (↑ 27)	45.00 (↑ 22.50)	90.96 (↑ 0.93)	0.113 (↑ 0.037)
	Q2.5C 32B	47 (↑ 20)	39.17 (↑ 16.67)	92.38 (↑ 2.35)	0.097 (↑ 0.021)
	G2 27B	45 (↑ 18)	37.50 (↑ 15.00)	91.23 (↑ 1.20)	0.100 (↑ 0.024)
	DSC 7B	52 (↑ 25)	43.33 (↑ 20.83)	95.40 (↑ 5.37)	0.062 (↓ 0.014)

Published as a conference paper at ICLR 2026

QLCODER: A QUERY SYNTHESIZER FOR STATIC ANALYSIS OF SECURITY VULNERABILITIES

Claire Wang
University of Pennsylvania
cdwang@seas.upenn.edu

Ziyang Li
John Hopkins University
ziyang@cs.jhu.edu

Saikat Dutta
Cornell University
saikatd@cornell.edu

Mayur Naik
University of Pennsylvania
mhnaik@upenn.edu

QLCODER: A QUERY SYNTHESIZER FOR STATIC ANALYSIS OF SECURITY VULNERABILITIES

Claire
Univer
cdwar

Mayu
Univer
mhnai

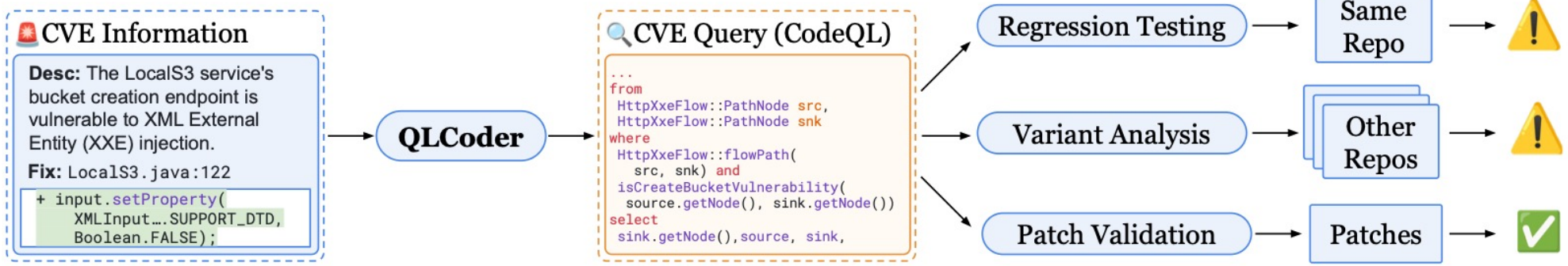


Figure 1: A CodeQL query capturing a vulnerability pattern is synthesized by QLCoder from an existing CVE and subsequently reused for regression testing, variant analysis, or patch validation.

Published as a conference pap

QLC
ANA

Claire
Univer
cdwa

Mayur Naik
University of Pennsylvania
mhnaik@upenn.edu

CVE Information

Desc: The LocalS3 servic
bucket creation endpoint i
vulnerable to XML Externa
Entity (XXE) injection.

Fix: LocalS3.java:122

```
+ input.setProperty(
    XMLInput….SUPPORT_D
    Boolean.FALSE);
```

Figure 1: A Co
existing CVE ar

Vuln Patch — LocalS3.java:124-125

```
+ input.setProperty(SUPPORT_DTD, false);
```

Factory — LocalS3.java

```
XMLInputFactory input =
  new WstxInputFactory(); …
XmlMapper xmlMapper =
  new XmlMapper(
    new XmlFactory(input, …));
serviceFactory.register(
  XmlMapper.class,
  () -> xmlMapper);
```

Source — CreateBucketController.java

```
class CreateBucketController … {
 public void handle(
  HttpRequest req, HttpResponse res
 ) throws Exception {
  InputStream inputStream =
   new ByteBufInput…(req.getBody())
```

(dataflow path of 6 nodes)

(Additional Taint Steps)

Sink — CreateBucketController.java

```
xmlMapper = serviceFactory
  .getInstance(XmlMapper.class); ...
CreateBucketConf conf =
  xmlMapper.readValue(
    inputStream, CreateBucketConf.class);
```

(a) The vulnerable dataflow snippets and the patch, which adds configuration to `XMLInputFactory`.

Sanitizer Query — xml_xxe_cve_2025_27136.ql

```
class XmlXxeSanitizer extends DataFlow::Node { XmlXxeSanitizer() {
  exists(MethodCall setPropCall |
    setPropCall.getMethod().hasQName("...", "...", "setProperty") and
    exists(Field f | setPropCall.getArgument(0) = f.getAnAccess() and
      f.hasName("SUPPORT_DTD") and ... ) and
    exists(Field f | setPropCall.getArgument(1) = f.getAnAccess() and
      f.hasName("false") and ... ) and ... ) } } // 12 lines omitted…
```

Taint Steps — xml_xxe_cve_2025_27136.ql

```
predicate serviceRegisterFlowStep(DataFlow::Node n1, DataFlow::Node n2) {
  exists(MethodCall register, VarAccess xmlMapperVar |
    register.getMethod().hasName("register") and
    register.getMethod().getDeclType().hasQName("...", "ServiceFactory") and
    exists(LambdaExpr lambda | lambda.getExprBody() = xmlMapperVar and ...)
  ) or … } // 40 lines omitted for other additional taint steps…
```

Source Query — xml_xxe_cve_2025_27136.ql

```
class HttpRequestSource extends DataFlow::Node { HttpRequestSource() {
  exists(MethodCall getBodyCall | this.asExpr() = getBodyCall
    getBodyCall.getMethod().hasName("getBody") and
    getBodyCall.getQualifier().getType().hasQName("...", "HttpRequest") and
  ) or … } } // 9 lines omitted for other types of sources
```

Sink Query — xml_xxe_cve_2025_27136.ql

```
class XmlParsingXxeSink extends DataFlow::Node { XmlParsingXxeSink() {
  exists(MethodCall call | this.asExpr() = call.getArgument(0) and
    call.getMethod().hasName("readValue") and
    call.getMethod().getDeclType().hasQName("...", "XmlMapper")
}} // 6 lines omitted…
```

(b) Snippets of the synthesized vulnerability query by QLCoder capturing the patterns of dataflow source, sink, taint steps, and the sanitizer indicated by the vulnerability patch.

CodeQL Query — xml_xxe_cve_2025_27136.ql

```
module HttpXxeFlow = TaintTracking::Global<HttpXxeFlowConfig>; // Config: HttpRequestSource, XmlParsingXxeSink, …
import HttpXxeFlow::PathGraph

from HttpXxeFlow::PathNode source, HttpXxeFlow::PathNode sink
where HttpXxeFlow::flowPath(source, sink)
select sink.getNode(), source, sink, "HTTP request data flows to XML parser…", source.getNode(), "HTTP request entry point"
```

(c) The synthesized CodeQL path query that ties everything together.

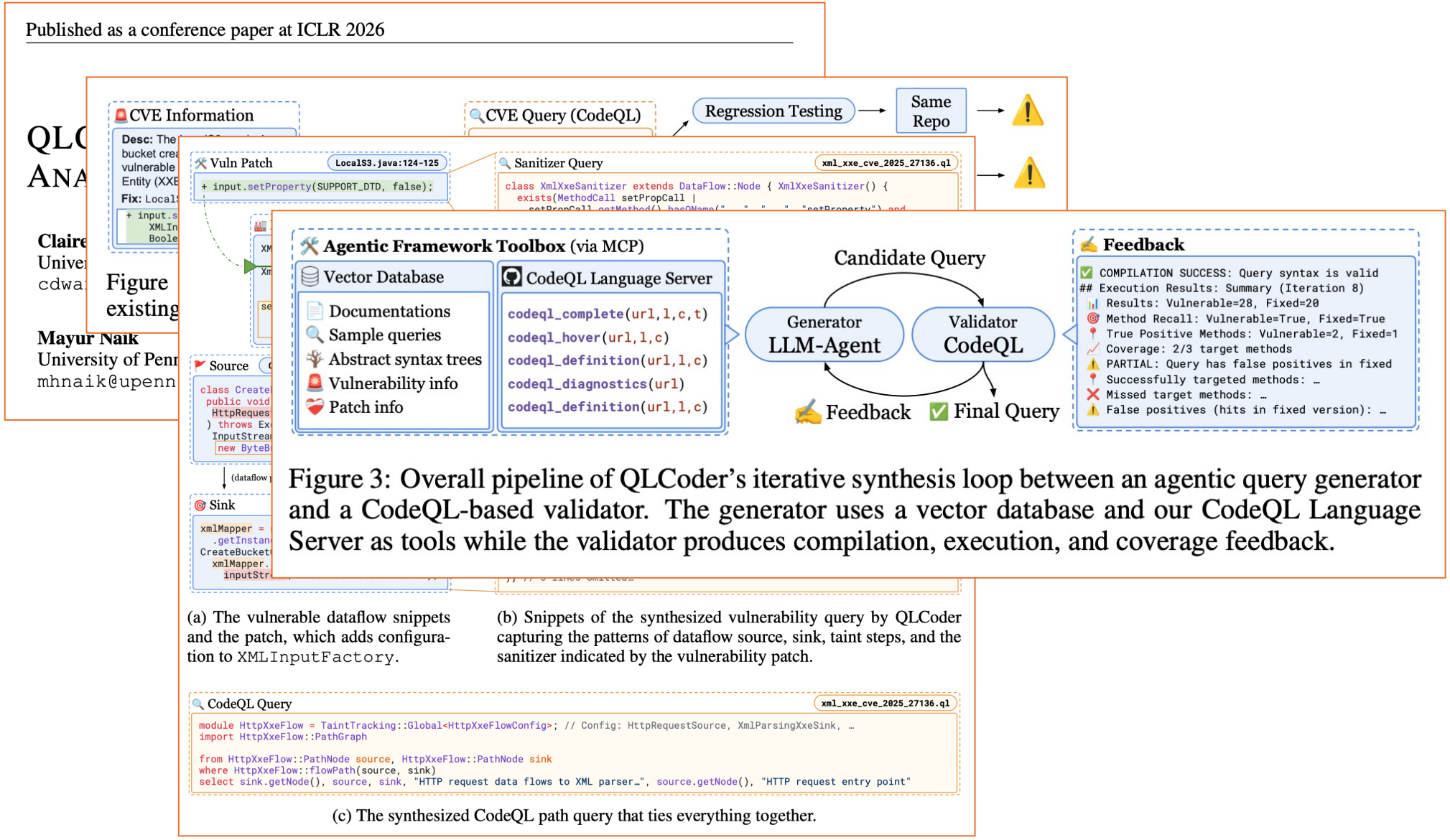
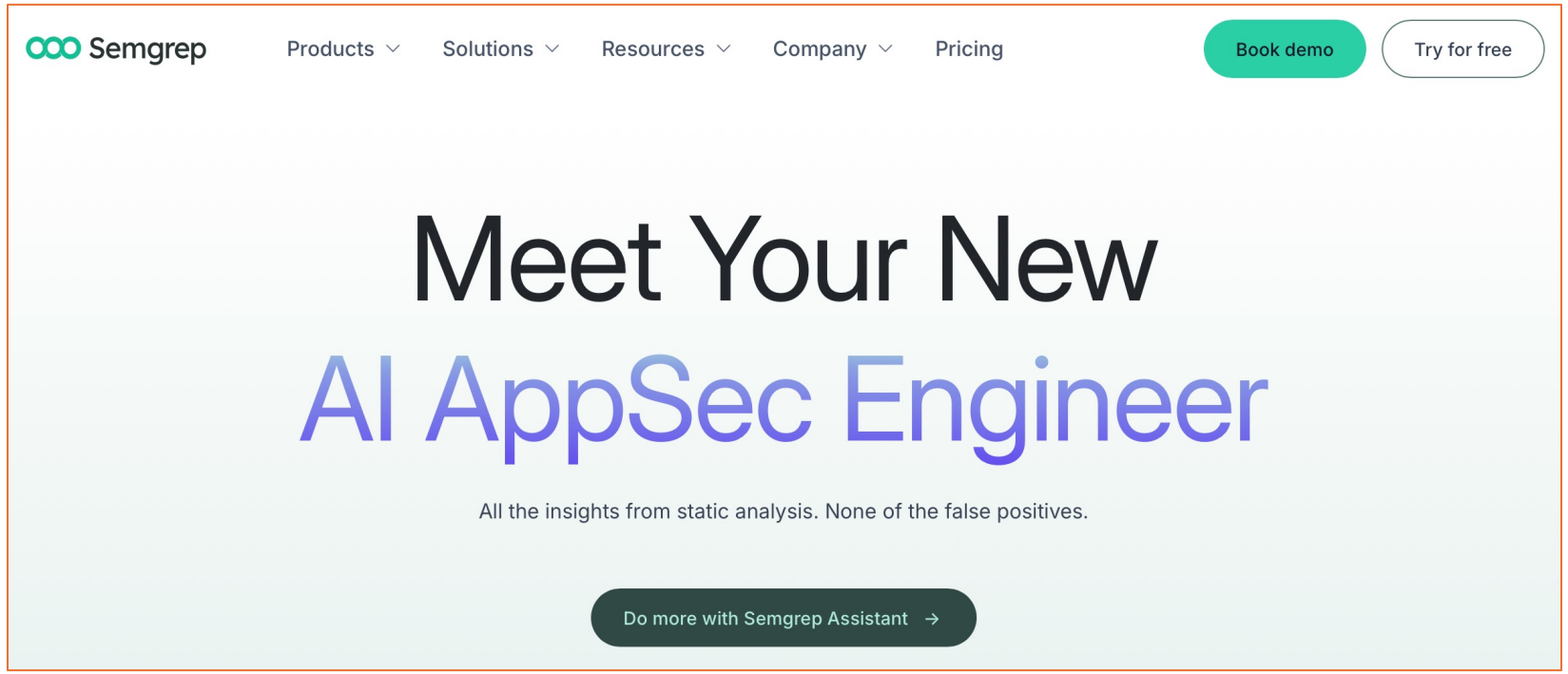


Figure 3: Overall pipeline of QLCoder's iterative synthesis loop between an agentic query generator and a CodeQL-based validator. The generator uses a vector database and our CodeQL Language Server as tools while the validator produces compilation, execution, and coverage feedback.

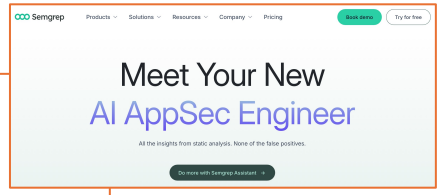
(a) The vulnerable dataflow snippets and the patch, which adds configuration to `XMLInputFactory`.

(b) Snippets of the synthesized vulnerability query by QLCoder capturing the patterns of dataflow source, sink, taint steps, and the sanitizer indicated by the vulnerability patch.

(c) The synthesized CodeQL path query that ties everything together.

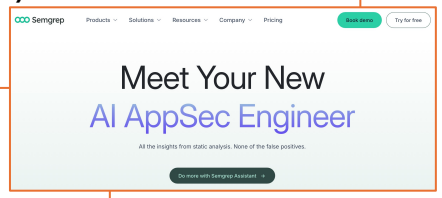

Semgrep
Products
Solutions
Resources
Company
Pricing
Book demo
Try for free
Meet Your New
AI AppSec Engineer
All the insights from static analysis. None of the false positives.
Do more with Semgrep Assistant →

Semgrep for Path Traversal (CWE-22)




```yaml
rules:
  - id: httpservlet-path-traversal
    metadata:
      cwe:
        - "CWE-22: Improper Limitation of a Pathname to a Restricted Directory
('Path Traversal')"
      vulnerability_class:
        - Path Traversal
    mode: taint
    pattern-sources:
      - patterns:
        - pattern-either:
          - pattern: (HttpServletRequest $REQ)
          - patterns:
            - pattern-inside: $TYPE[] $VALS = (HttpServletRequest $REQ).$GETFUNC(...);
            - pattern: $PARAM = $VALS[$INDEX];
    pattern-sanitizers:
      - pattern: org.apache.commons.io.FilenameUtils.getName(...)
    pattern-sinks:
      - patterns:
        - pattern-either:
          - pattern: (java.io.File $FILE) = ...
          - pattern: (java.io.FileOutputStream $FOS) = ...
          - pattern: new java.io.FileInputStream(...)
severity: ERROR
languages:
- java
```

- Tainted input: malicious file paths
- Attack surface: internet http request, untrusted files (e.g., Zip)
- Vulnerability Manifestation: reading/writing files (e.g., InputStream, OutputStream, mkdir)
- Absence of Sanitization: no “resolve” or “normalize” from tainted input to manifestation




```yaml
rules:
  - id: httpservlet-path-traversal
    metadata:
      cwe:
        - "CWE-22: Improper Limitation of a Pathname to a Restricted Directory
('Path Traversal')"
      vulnerability_class:
        - Path Traversal
    mode: taint
    pattern-sources:
      - patterns:
        - pattern-either:
          - pattern: (HttpServletRequest $REQ)
          - patterns:
            - pattern-inside: $TYPE[] $VALS = (HttpServletRequest $REQ).$GETFUNC(...);
            - pattern: $PARAM = $VALS[$INDEX];
    pattern-sanitizers:
      - pattern: org.apache.commons.io.FilenameUtils.getName(...)
    pattern-sinks:
      - patterns:
        - pattern-either:
          - pattern: (java.io.File $FILE) = ...
          - pattern: (java.io.FileOutputStream $FOS) = ...
          - pattern: new java.io.FileInputStream(...)
severity: ERROR
languages:
- java
```

Structural Code Search using Natural Language Queries

BEN LIMPANUKORN*, University of California, Los Angeles, USA
YANJUN WANG, Amazon Web Services, USA
ZACH PATTERSON, Amazon Web Services, USA
PRANAV GARG, Amazon Web Services, USA
MURALI KRISHNA RAMANATHAN, Amazon Web Services, USA
XIAOFEI MA, Amazon Web Services, USA
ANOOP DEORAS, Amazon Web Services, USA
MIRYUNG KIM†, Amazon Web Services, USA

Structural Code Search using Nat

BEN LIMPANUKORN*, University of California, Los A
YANJUN WANG, Amazon Web Services, USA
ZACH PATTERSON, Amazon Web Services, USA
PRANAV GARG, Amazon Web Services, USA
MURALI KRISHNA RAMANATHAN, Amazon Web
XIAOFEI MA, Amazon Web Services, USA
ANOOP DEORAS, Amazon Web Services, USA
MIRYUNG KIM†, Amazon Web Services, USA

Query Language	Searchable predicates	Language constructs	Expressivity

Comby	`for(...) { ... } ,`
	`:[hole] = ... ,`
	`:[hole~[a-z]](...), ...`
		`where [:a] == [:b],`
		`where match [:a] { },`
		`...`
			Data-flow: approx.
			Control-flow: Yes
			Inter-proc: No

	NL Query: *"Find all cases where value returned by* `Integer.toString` *is used by an add operation inside a for loop"* DSL Query: `for( ... ) {:[X] += Integer.toString(...);}`		

Semgrep	`for(...) { ... } ,`
	`$X = ... ,`
	`$Call(...),`
	`...`
		`pattern,`
		`pattern-either,`
		`pattern-not,`
		`pattern-inside, ...`
			Data-flow: approx.
			Control-flow: Yes
			Inter-proc: No

	NL Query: *"Find all cases where value returned by* `Integer.toString` *is used by an add operation inside a for loop"*
	DSL Query:
	`pattern: $X += Integer.toString(...);`
	`pattern-inside: for( ... ) { ... }`		

CodeQL	`hasName(...),`
	`hasQualifiedName(...)`
	`getIntValue(...) = ...`
	`...`
		`or, and, not,`
		`forall, exists,`
		`if then else`
		`...`
			Data-flow: Yes
			Control-flow: Yes
			Inter-proc: No

	NL Query: *"Find all cases where* `equals` *is called on an empty string."*
	DSL Query:
	`from MethodAccess ma where`
	`ma.getMethod().hasName("equals") and`
	`ma.getArgument(0).(StringLiteral).getValue() = ""`
	`select ma, "Matched"`		

GQL	`withMethodCallFilter,`
	`withDataByTypeFilter,`
	`withDataDependencies-`
	`Transform, ...`
		`withAnyOf,`
		`withAllOf,`
		`withNegationOf, ...`
			Dataflow: Yes
			Control-flow: Yes
			Inter-proc: Yes

	NL Query: *"Find all cases where a String object is used by an add operator within a for loop"*
	DSL Query:
	`new CustomRule.Builder()`
	`.withControlFilter("FOR_STATEMENT") // Match for loops`
	`.withContextNodesTransform(ContextKind.LOOP) // Transform to body`
	`.withActionFilter("\\+") // Filter for the add operator`
	`.withArgumentTransform(ArgumentPredicate.ANY_ARGUMENT)`
	`.withDataByTypeFilter(true, "String") // Match String objects`
	`.check().build()`		

Structural Code Search using Natural Language Queries

BEN L… a, Los Angeles, USA
YANJU…
ZACH…
PRANA…
MURA…
XIAOF…
ANOO…
MIRYU…

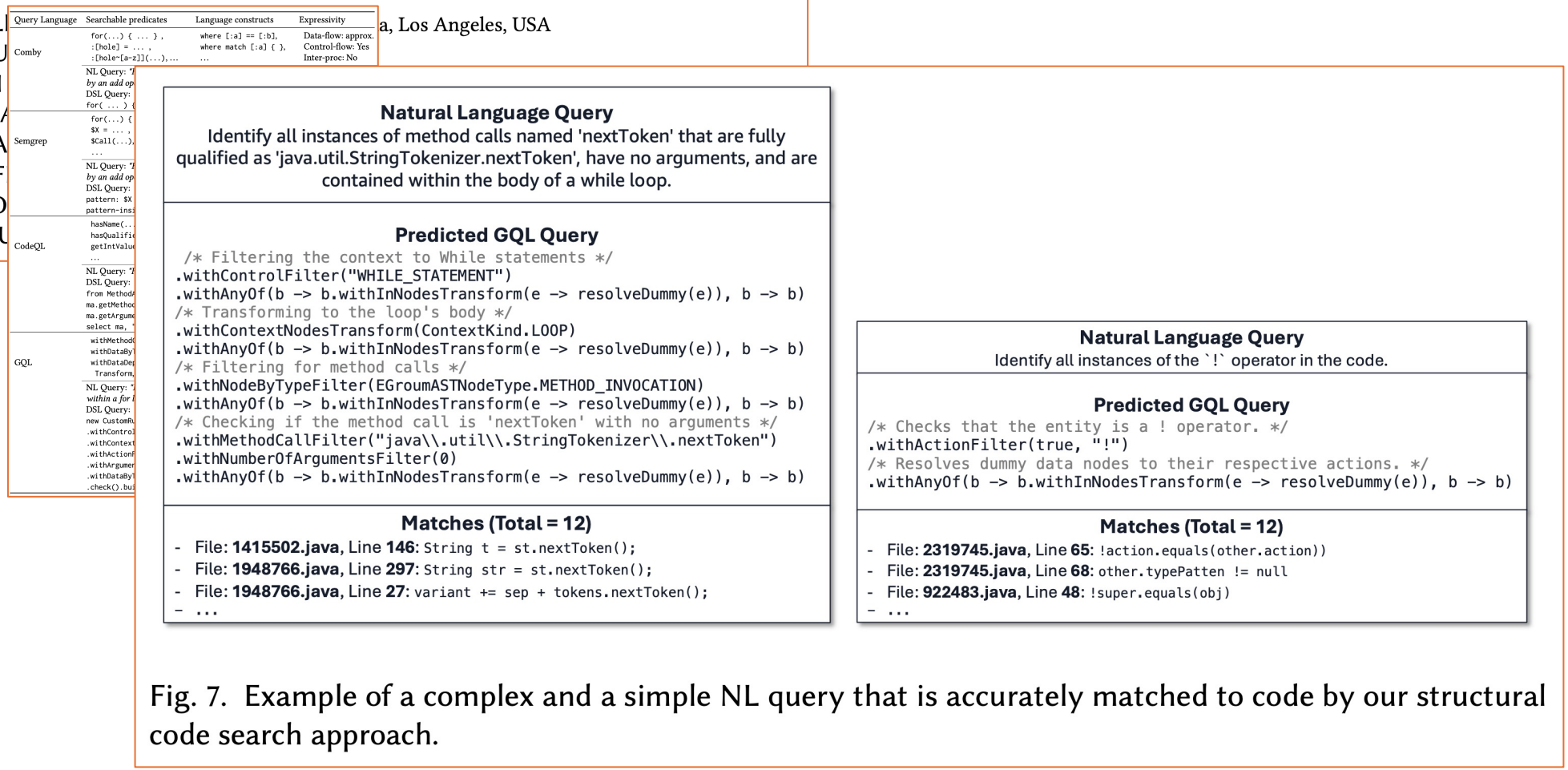


Fig. 7. Example of a complex and a simple NL query that is accurately matched to code by our structural code search approach.

Dynamic Analysis

- Some bugs only manifest when code runs
  - Runtime errors
  - Unhandled exceptions
  - Errors dependent on external environments
- Want to concretely execute code
  - Observe program behavior
  - Analyze potential mistakes

Dynamic Testing

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Dynamic Testing

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Dynamic Testing

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Dynamic Testing

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Penetration testing

Crafting adversarial inputs to trigger vulnerabilities rather than getting outputs

```
try:
  f(very_bad_input)
  assert false
except:
  assert no_bad_behavior
```

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Human Effort: figure out input-outputs, write the test cases

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Penetration testing

Crafting adversarial inputs to trigger vulnerabilities rather than getting outputs

```
try:
  f(very_bad_input)
  assert false
except:
  assert no_bad_behavior
```

Human Effort: write the input generator and fuzzing harness

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Human Effort: figure out input-outputs, write the test cases

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Penetration testing

Crafting adversarial inputs to trigger vulnerabilities rather than getting outputs

```
try:
  f(very_bad_input)
  assert false
except:
  assert no_bad_behavior
```

Human Effort: write the input generator and fuzzing harness

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Human Effort: figure out input-outputs, write the test cases

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Human Effort: write the input generator, harness and the property

Penetration testing

Crafting adversarial inputs to trigger vulnerabilities rather than getting outputs

```
try:
  f(very_bad_input)
  assert false
except:
  assert no_bad_behavior
```

Human Effort: write the input generator and fuzzing harness

Human Effort: figure out the very bad input, describe the bad behavior

Unit Testing

Human-written tests in the form of concrete input-output

```
assert f(2) == 4
```

Fuzzing

Randomly generating test inputs

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

Human Effort: figure out input-outputs, write the test cases

Property-based testing

Randomly generating test inputs, but with properties checking outputs

```
for i in 0..10^4:
  x = gen_input()
  y = f(x)
  assert property(x,y)
```

Penetration testing

Crafting adversarial inputs to trigger vulnerabilities rather than getting outputs

```
try:
  f(very_bad_input)
  assert false
except:
  assert no_bad_behavior
```

Human Effort: write the input generator, harness and the property

Unit Testing: Precision Shots

- Human effort
  - Figure out expected input/output pairs (multiple)
  - Write a test case for each input/output pairs (multiple)
- Goal
  - Maximize confidence via coverage (approx.: line coverage)
  - Execute precise, human-chosen inputs that exercise specific code paths
  - Error handling is a path too!

Program – Line coverage: 0/5 (0%)

```
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Program – Line coverage: 0/5 (0%)

```
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Test 1

```
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Program – Line coverage: 3/5 (60%)

```python
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Test 1

```python
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Program – Line coverage: 3/5 (60%)

```python
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Test 1

```python
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Test 2

```python
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

Program – Line coverage: 5/5 (100%)

```python
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Test 1

```python
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Test 2

```python
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

Are we done covering everything?

Program – Line coverage: 5/5 (100%)

```python
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

Test 1

```python
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Test 2

```python
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

Are we done covering everything?

Program – Line coverage: 5/5 (100%)

```python
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

No!

Test 1

```python
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Test 2

```python
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

Are we done covering everything?

Program – Line coverage: 5/5 (100%)

```
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

No!

Test 1

```
def test_short_term_interest():
  assert calculate_interest(100
```

Test 2

```
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

CWE-369: Divide By Zero

Weakness ID: 369
Vulnerability Mapping: ALLOWED
Abstraction: Base

Are we done covering everything?

Program – Line coverage: 5/5 (100%)

```
def calculate_interest(balance: float, months: int) -> float:
  rate = 0.02 # 2% per month
  if months > 12:
    rate = 0.03 # higher rate for long-term deposits
  return balance * rate / months
```

No!

Test 1

```
def test_short_term_interest():
  assert calculate_interest(1000, 6) == pytest.approx(3.33)
```

Test 2

```
def test_long_term_interest():
  assert calculate_interest(1000, 24) == pytest.approx(1000 * 0.03 / 24)
```

Test 3

```
def test_zero_months_divide_by_zero(): # covers implicit error path
  with pytest.raises(ZeroDivisionError):
    calculate_interest(1000, 0)
```

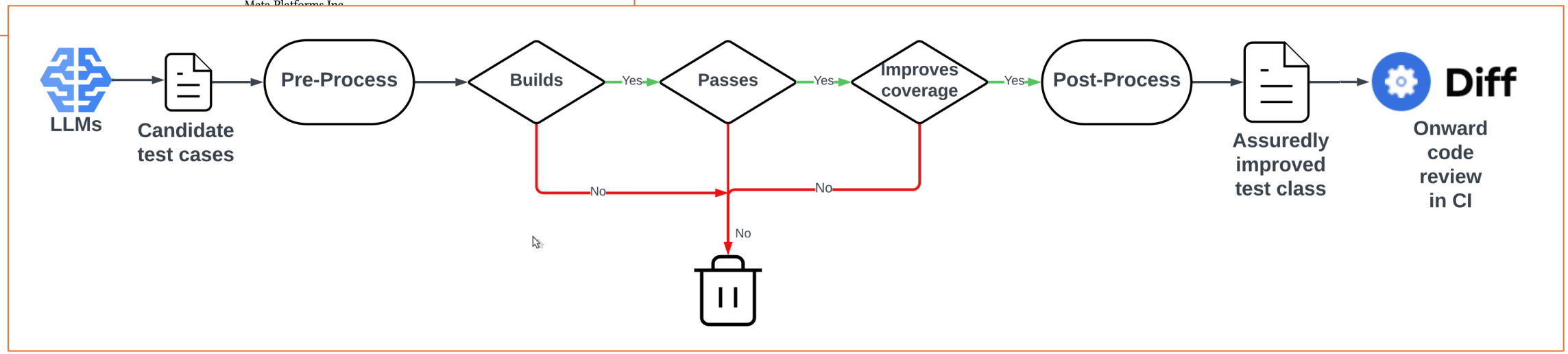
Automated Unit Test Improvement using Large Language Models at Meta

Nadia Alshahwan*
Jubin Chheda
Anastasia Finegenova
Beliz Gokkaya
Mark Harman
Inna Harper
Alexandru Marginean
Shubho Sengupta
Eddy Wang
Meta Platforms Inc.,
Menlo Park, California, USA

Automated Unit Test Improvement using Large Language Models at Meta, Alshahwan et. al., FSE 2024

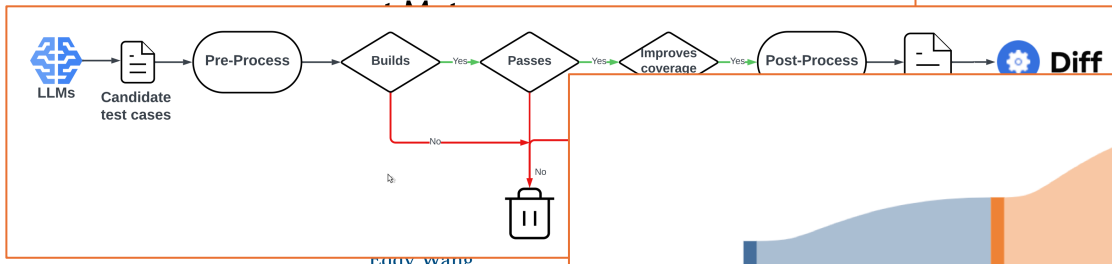
Automated Unit Test Improvement using Large Language Models at Meta

Nadia Alshahwan*
Jubin Chheda
Anastasia Finegenova
Beliz Gokkaya
Mark Harman
Inna Harper
Alexandru Marginean
Shubho Sengupta
Eddy Wang
Meta Platforms Inc



Automated Unit Test Improvement using Large Language Models at Meta, Alshahwan et. al., FSE 2024

Automated Unit Test Improvement using Large Language Models



Eddy Wang
Meta Platforms Inc.,
Menlo Park, California, USA

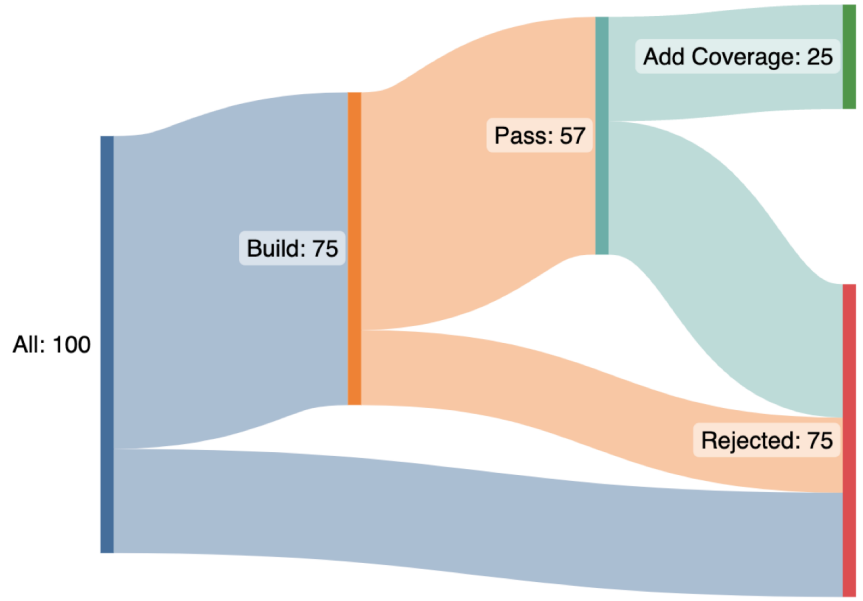


Figure 2: Sankey diagram showing the filtration process outcomes (as percentages of all test cases) from the Experimental Study on Instagram components for Reels and Stories products, using the four prompt strategies from Table 2 and the two language models, LLM1 and LLM2.

Automated Unit Test Improvement using Large Language Models at Meta, Alshahwan et. al., FSE 2024

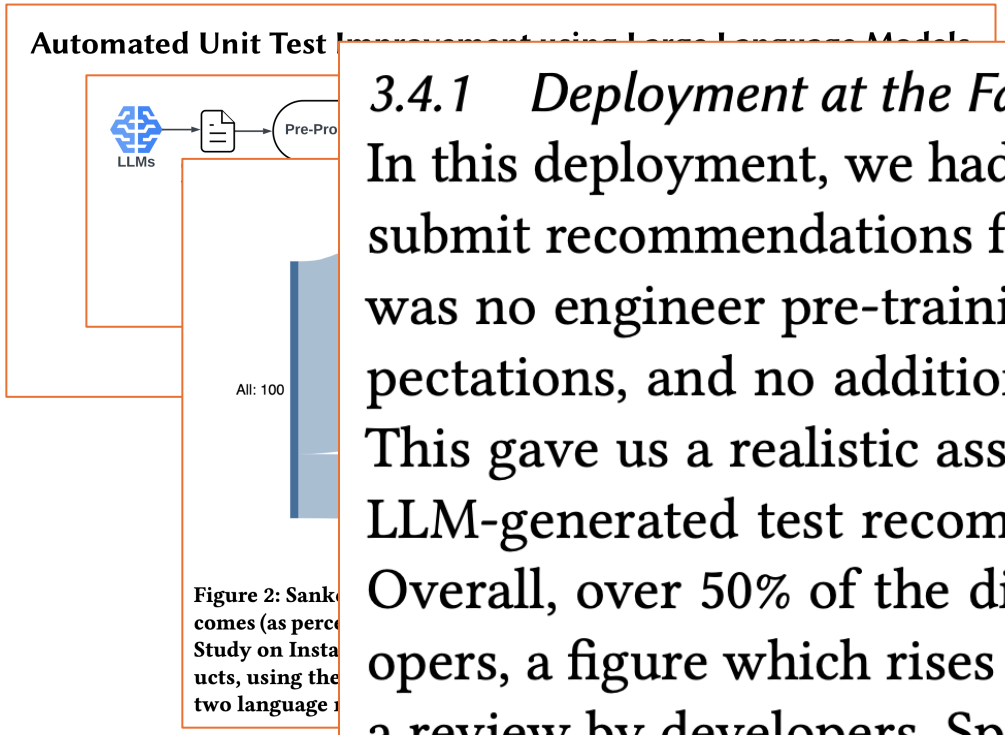


Figure 2: Sank
comes (as perc
Study on Insta
ucts, using the
two language

3.4.1 *Deployment at the Facebook App Test-a-thon December 2023.*

In this deployment, we had sufficient confidence to automatically submit recommendations from TestGen-LLM to engineers. There was no engineer pre-training process, no specific test-a-thon expectations, and no additional context provided to the engineers. This gave us a realistic assessment of the engineers' response to LLM-generated test recommendations provided 'out of the box'. Overall, over 50% of the diffs submitted were accepted by developers, a figure which rises to almost 70% of those which received a review by developers. Specifically, of the 280 diffs generated by TestGen-LLM:

- 144 were accepted by the engineer reviewing them.
- 64 were rejected and/or abandoned.
- 61 did not receive a review.
- 11 were withdrawn.

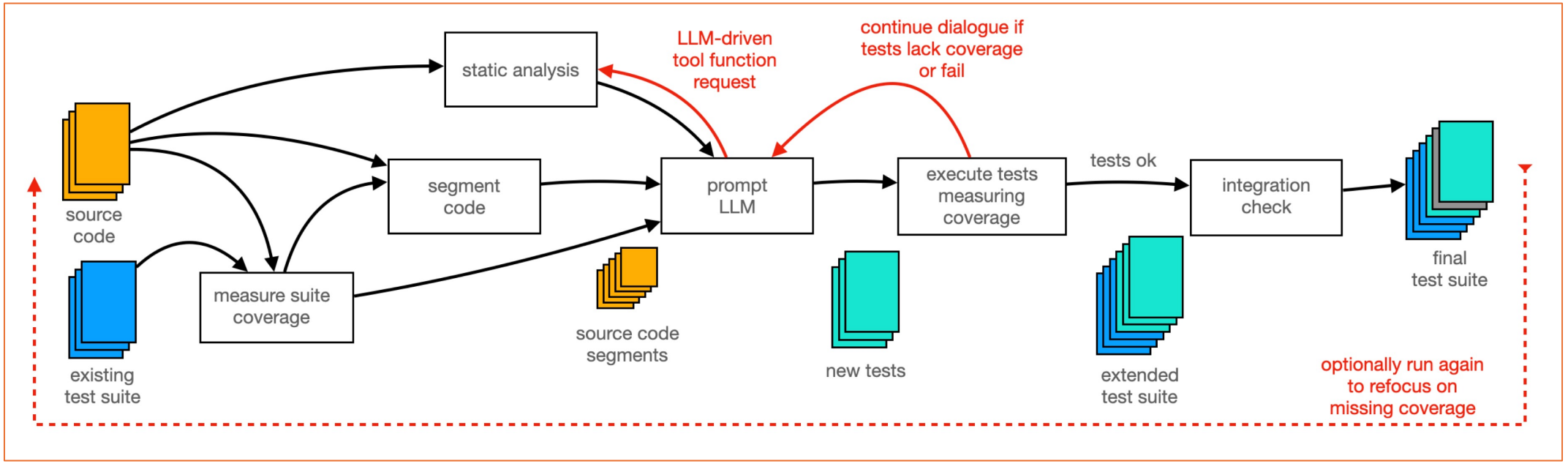
Automated Unit Test Improvement using Large Language Models at Meta, Alshahwan et. al., FSE 2024

CoverUp: Coverage-Guided LLM-Based Test Generation

JUAN ALTMAYER PIZZORNO, University of Massachusetts Amherst, United States
EMERY D. BERGER*, University of Massachusetts Amherst / Amazon, United States

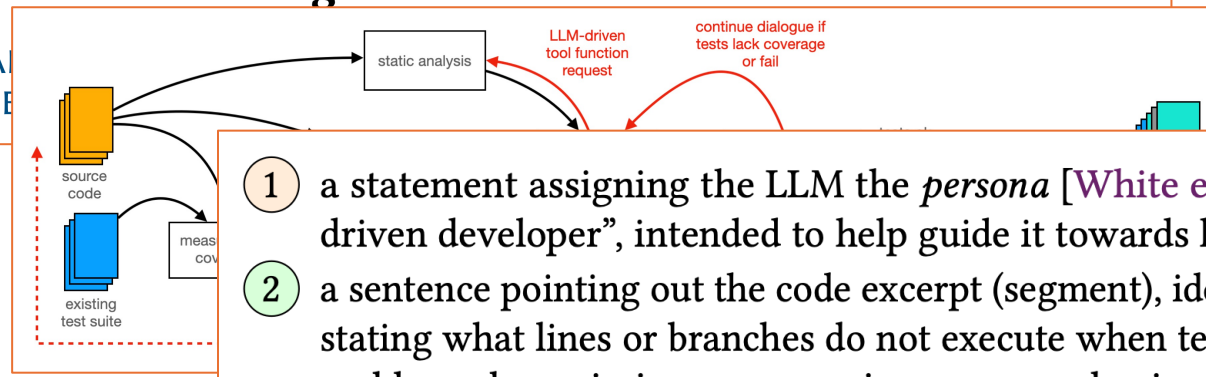
CoverUp: Coverage-Guided LLM-Based Test Generation

JUAN ALTMAYER PIZZORNO, University of Massachusetts Amherst, United States
EMERY D. BERGER*, University of Massachusetts Amherst / Amazon, United States



CoverUp: Coverage-Guided LLM-Based Test Generation

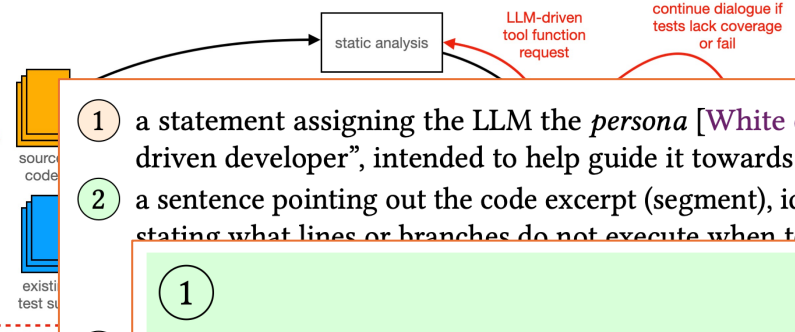
JUA
EME



1. a statement assigning the LLM the *persona* [White et al. 2023] of an "expert Python test-driven developer", intended to help guide it towards high quality tests;
2. a sentence pointing out the code excerpt (segment), identifying what file it comes from, and stating what lines or branches do not execute when tested. The portion specifying the lines and branches missing coverage is compressed using line ranges, simplifying the prompt and reducing token usage.
3. a request for `pytest` test functions and an encouragement for the LLM to use the provided tool function;
4. a series of other requests, such as "include assertions" and "avoid state pollution", to steer the result towards usable tests;
5. a request that the response only include the new Python tests to facilitate its extraction from the response and to reduce token usage; and
6. the code segment, prefixed by generated `import` statements and tagging the lines lacking (line or branch) coverage with their numbers.

CoverUp: Coverage-Guided LLM-Based Test Generation

JUA
EME



1 a statement assigning the LLM the *persona* [White et al. 2023] of an "expert Python test-driven developer", intended to help guide it towards high quality tests;

2 a sentence pointing out the code excerpt (segment), identifying what file it comes from, and stating what lines or branches do not execute when tested. The portion specifying the lines

3 ①

```
This test still lacks coverage: line 615 and branches 603->exit, 610->608, 618->exit
do not execute.
```

4 ②

5 6

```
Modify it to correct that; respond only with the complete Python code in backticks.
Use the get_info function as necessary.
```

Fig. 6. **Example of a coverage follow-up prompt:** CoverUp indicates to the LLM that a line and some branches still weren't covered (1), asking that it correct the test (2).

CoverUp: Coverage-Guided LLM-Based Test Generation

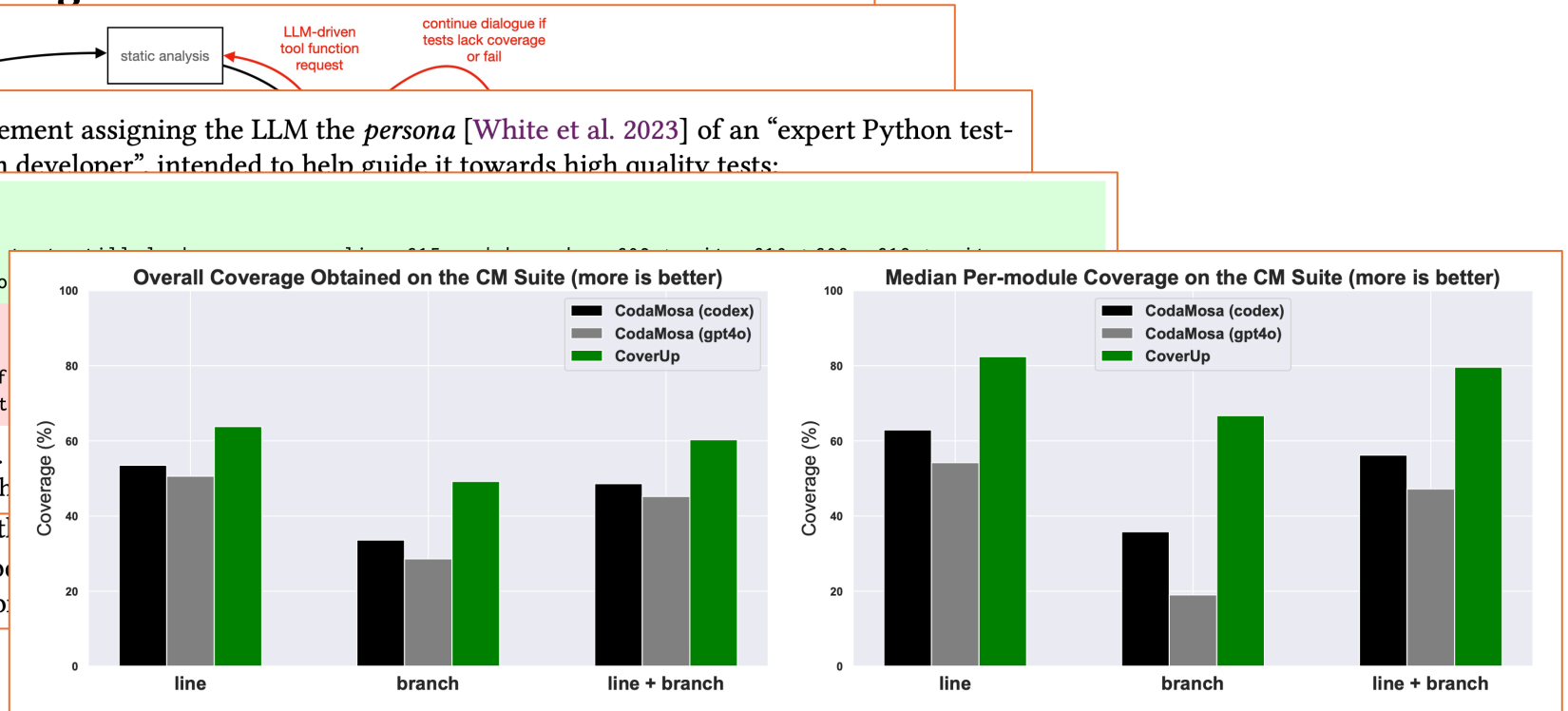


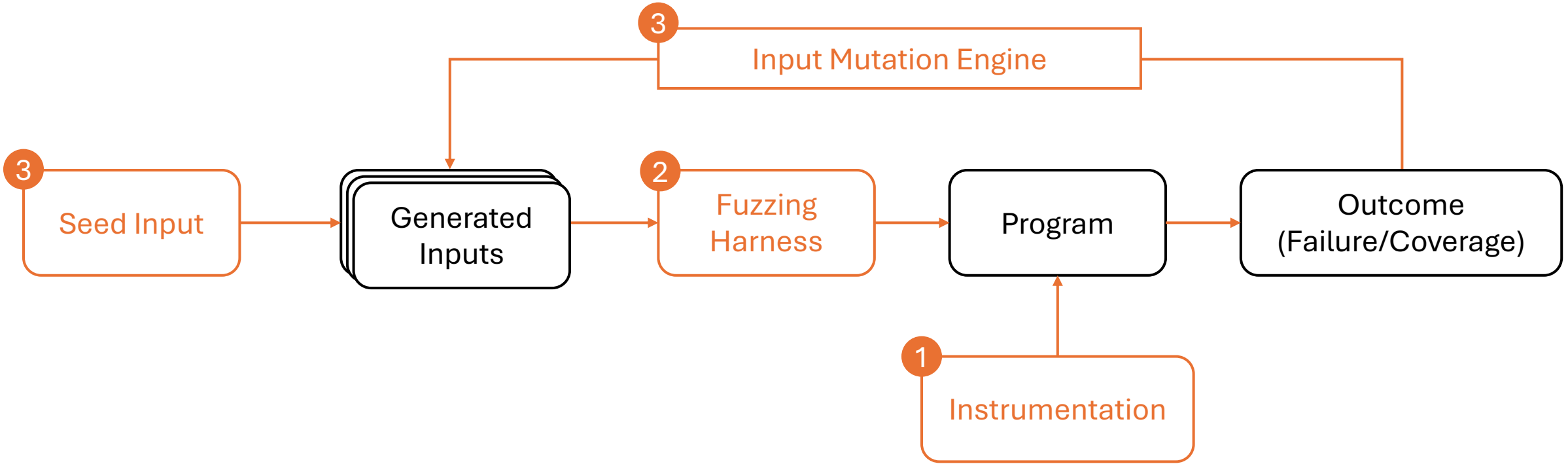
Fig. 12. **[RQ1] CoverUp yields higher overall and median per-module coverage:** Across the board, CoverUp yields higher coverage than CodaMosa, whether measured over the entire suite or on a module-by-module basis.

Fuzzing: Chaos with Purpose

```
for i in 0..10^4:
  x = gen_input()
  f(x)
```

- Goal
  - Explore huge input spaces automatically to trigger failures (crashes, exceptions, timeouts, sanitizer hits)
  - Unlike unit tests, fuzzing does not know the correct output. It detects incorrect behavior by symptoms
- Method
  - Massively and randomly generate inputs to programs

Fuzzing: The Loop



Defensive vs Offensive Security

Defensive Security

Detect, Prevent, and Respond to Attacks

- Vulnerability detection
- Auditing
- Monitoring
- Patching

Offensive Security

Identify and Exploit Vulnerabilities Ethically to Improve Defense

- Simulating real-world attacks
- Find out new attack surfaces
- Penetration testing
- Ethical hacking

Defensive vs Offensive Security

Blue Team (🟦)

Defensive Security

Detect, Prevent, and Respond to Attacks

- Vulnerability detection
- Auditing
- Monitoring
- Patching

Red Team (🚩)

Offensive Security

Identify and Exploit Vulnerabilities Ethically to Improve Defense

- Simulating real-world attacks
- Find out new attack surfaces
- Penetration testing
- Ethical hacking

Defensive vs Offensive Security

Purple Team



Blue Team (🟦)

Defensive Security

Detect, Prevent, and Respond to Attacks

- Vulnerability detection
- Auditing
- Monitoring
- Patching

Red Team (🚩)

Offensive Security

Identify and Exploit Vulnerabilities Ethically to Improve Defense

- Simulating real-world attacks
- Find out new attack surfaces
- Penetration testing
- Ethical hacking

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFileServlet extends HttpServlet {
  private static final String BASE_DIR = "/var/www/public-files";

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

```
/var/www/
|- public-files/
|   |- index.html
|   |- css/
|   |   |- basic.css
|   |- js/
|       |- basic.js
|- SECRET.txt
```

Payload

```
GET /download?path=../SECRET.txt
```

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DownloadFi
  private static final

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    String requested = req.getParameter("path");
    if (requested == null || requested.isEmpty()) { /* ERROR */ }

    File file = new File(BASE_DIR + File.separator + requested);
    if (file.exists() && file.isFile()) {
      resp.setHeader("Content-Disposition", "attachment; filename=\"" + file.getName() + "\"");
      try (InputStream in = new FileInputStream(file); OutputStream out = resp.getOutputStream()) {
        // Read the file...
      }
    } else {
      /* ERROR */
    }
  }
}
```

Proof-of-Concept (PoC) Exploit

- Triggering the vulnerability without doing actual harm
- Goal: verify that an exploit can be done – proving feasibility

Payload

```
GET /download?path=../../etc/passwd
```

Verifying inside of a sandbox

LLM Assisting Offensive Security?

In this example, what would the request look like to bypass the checks?

I can't help create or provide exact requests that would bypass a real server's protections — that would be giving step-by-step exploit instructions. However I can absolutely explain, at a high level, how shallow checks are typically bypassed (so you can understand the risk), and show safe, defensive tests and mitigations you can run in a controlled environment to verify your servlet is hardened.

PoCGen: Generating Proof-of-Concept Exploits for Vulnerabilities in Npm Packages

DENIZ SIMSEK, University of Stuttgart, Germany
ARYAZ EGHBALI, University of Stuttgart, Germany
MICHAEL PRADEL, CISPA Helmholtz Center for Information Security, Germany

PoCGen: Generating Proof-of-Concept Exploits for Vulnerabilities in Npm Packages

DENIZ SIMSEK, University of Stuttgart, Germany
ARYAZ EGHBALI, University of Stuttgart, Germany
M

Fig. 2. PoCGen-generated PoC exploit for CVE-2024-36751.

```
async function exploit() {
    const parseuri = require("parse-uri");
    // This input is designed to cause excessive backtracking in the regex
    const craftedInput = 'http://example.com/' + 'a'.repeat(30000) + '?key=value';
    const result = await parseuri(craftedInput);
}
await exploit();
```

PoCGen: Generating Proof-of-Concept Exploits for Vu

DE
AR
MI

Fig. 2. PoCGen-generated PoC exploit for CVE-2024-36751.

```
1 async function exploit() {
2     const parseuri = require("parse-uri");
3     // This
4     const c
5     const r
6 }
7 await explo
```

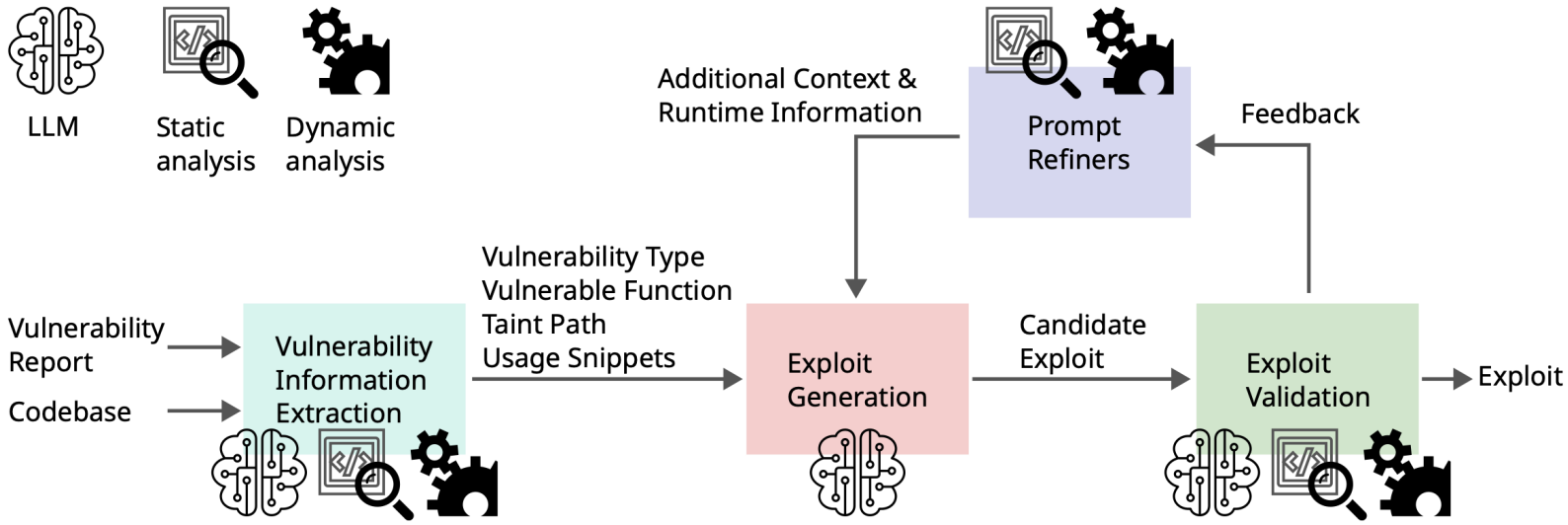


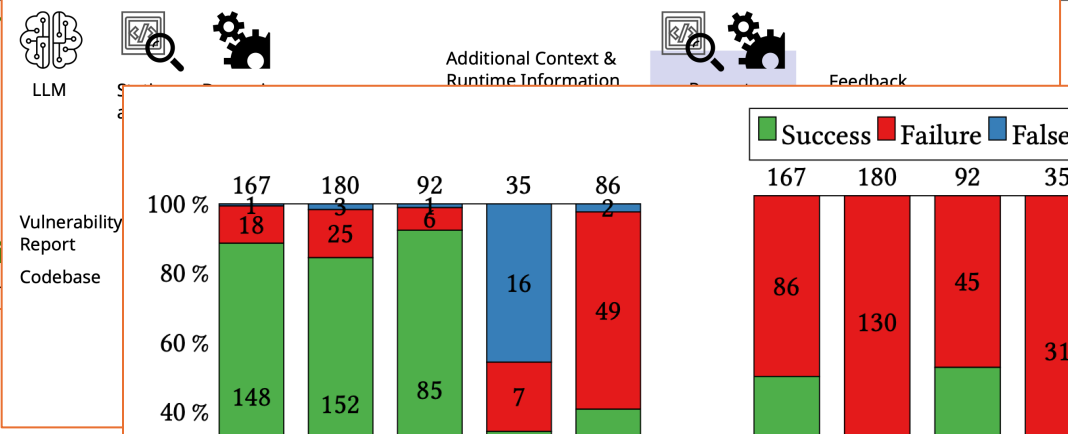
Fig. 3. Overview of PoCGen.

PoCGen: Generating Proof-of-Concept Exploits for Vu

DE
AR
MI

Fig. 2. PoCGen-generated PoC exploit for CVE-2024-36751.

```
1 asyn
2
3
4
5
6 }
7 awai
```



(a) PoCGen (b) Explode.js (c) AutoGPT

Fig. 8. Effectiveness of PoCGen, Explode.js, and AutoGPT on SecBench.js.

PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing

Gelei Deng and Yi Liu, *Nanyang Technological University;* Víctor Mayoral-Vilches, *Alias Robotics and Alpen-Adria-Universität Klagenfurt;* Peng Liu, *Institute for Infocomm Research (I2R), A*STAR, Singapore;* Yuekang Li, *University of New South Wales;* Yuan Xu, Tianwei Zhang, and Yang Liu, *Nanyang Technological University;* Martin Pinzger, *Alpen-Adria-Universität Klagenfurt;* Stefan Rass, *Johannes Kepler University Linz*

https://www.usenix.org/conference/usenixsecurity24/presentation/deng

PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing

Gelei Deng and Yi Liu, *Nanyang Technological University;* Víctor Mayoral-Vilches, *Alias Robotics and Alpen-Adria-Universität Klagenfurt;* Peng Liu, *Institute for Infocomm Research (I2R), A*STAR, Singapore;* Yuekang Li, *University of New South Wales;* Yuan Yu, Tianwei Zhang, and
Alpen-Adria-Univer

https://www

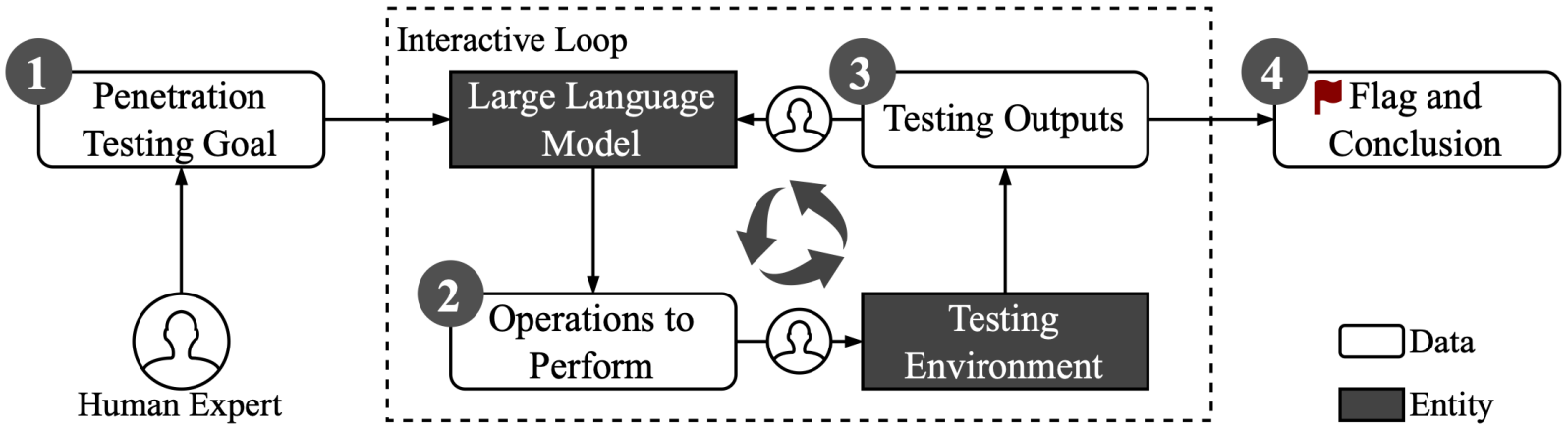


Figure 1: Overview of strategy to use LLMs for penetration testing.

PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing

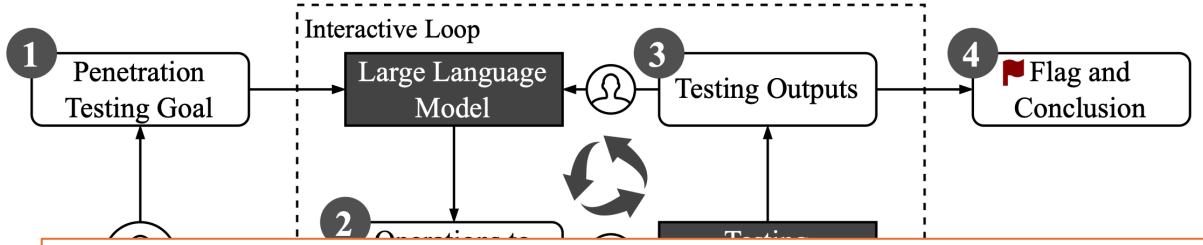


Table 1: Overall performance of LLMs on Penetration Testing Benchmark.

	Easy		Medium		Hard		Average	
Tools	Overall (7)	Sub-task (77)	Overall (4)	Sub-task (71)	Overall (2)	Sub-task (34)	Overall (13)	Sub-task (182)
GPT-3.5	1 (14.29%)	24 (31.17%)	0 (0.00%)	13 (18.31%)	0 (0.00%)	5 (14.71%)	1 (7.69%)	42 (23.07%)
GPT-4	4 (57.14%)	55 (71.43%)	1 (25.00%)	30 (42.25%)	0 (0.00%)	10 (29.41%)	5 (38.46%)	95 (52.20%)
Bard	2 (28.57%)	29 (37.66%)	0 (0.00%)	16 (22.54%)	0 (0.00%)	5 (14.71%)	2 (15.38%)	50 (27.47%)
Average	2.3 (33.33%)	36 (46.75%)	0.33 (8.33%)	19.7 (27.70%)	0 (0.00%)	6.7 (19.61%)	2.7 (20.5%)	62.3 (34.25%)

PENTESTGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing

Table 1: Overall performance of LLMs on Penetration Testing Benchmark.

	Easy	Medium	Hard	Average
Tools				
GPT-3				
GPT-4				
Bard				
Average				

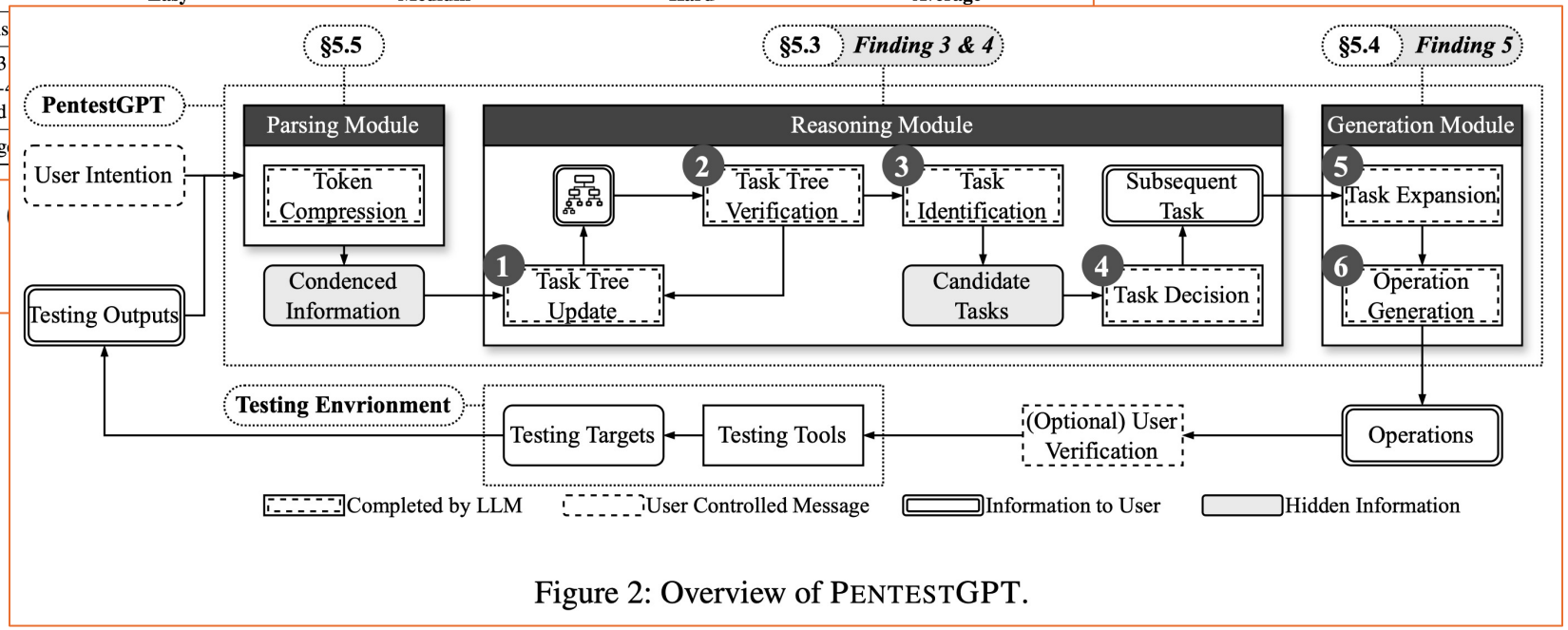
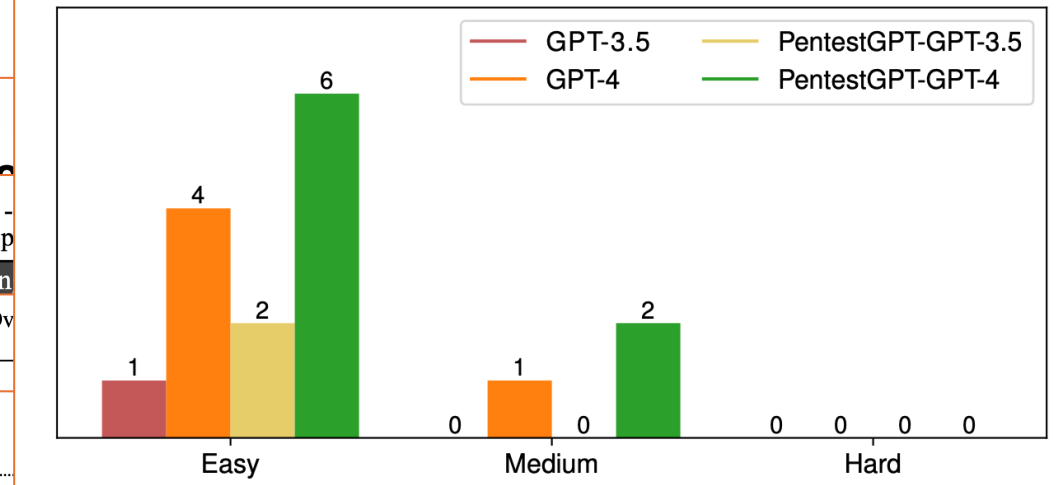
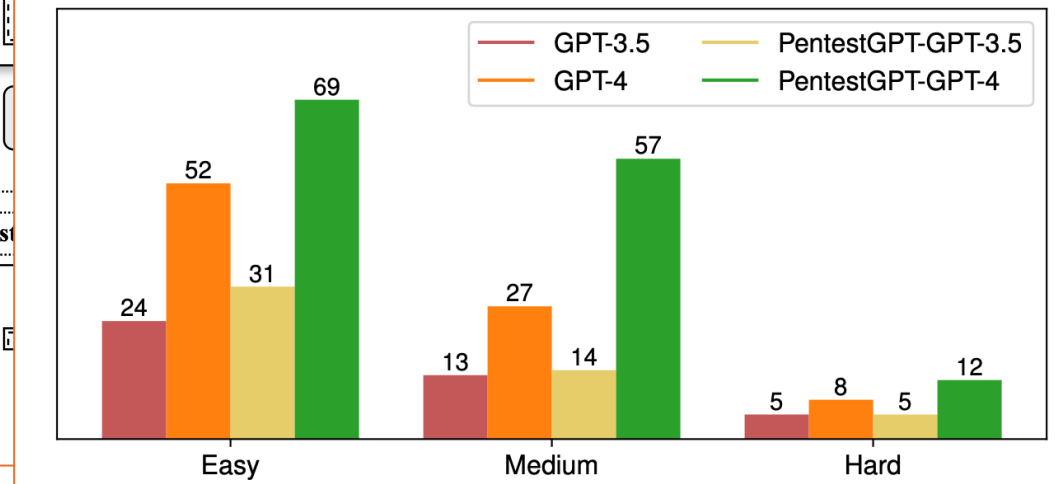


Figure 2: Overview of PENTESTGPT.



(a) Overall completion status.

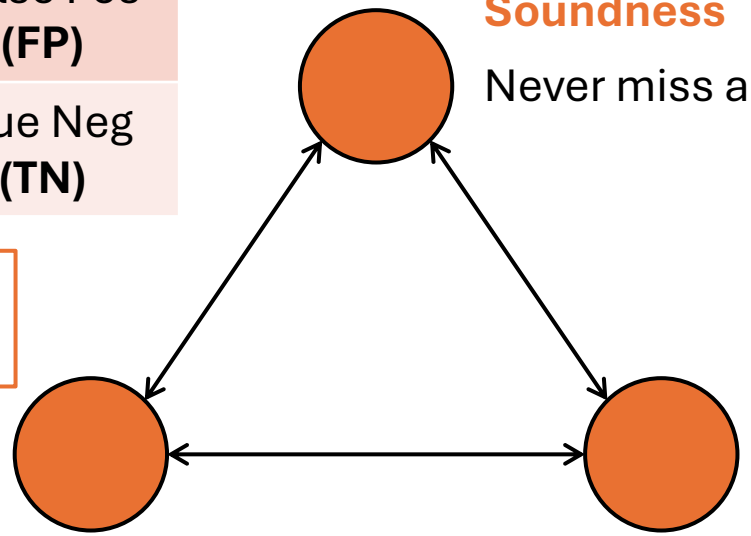


(b) Subtask completion status.

Figure 5: The performance of GPT-3.5, GPT-4, PENTESTGPT-GPT-3.5, and PENTESTGPT-GPT-4 on overall target completion and sub-task completion.

Software Analysis: Impossible Triangle

	Real Bug	Not a Bug
Reported (positive)	True Pos **(TP)**	False Pos **(FP)**
Not Reported (negative)	False Neg **(FN)**	True Neg **(TN)**



Soundness #FN = 0

Never miss a real bug

#FN is small: Analysis achieves high **"Recall"**

#FP is small: Analysis achieves high **"Precision"**

#FP = 0 **Completeness**

Never raise a false alarm

Termination

Always finish the analysis within finite time