FIXX: FInding eXploits from eXamples

34th USENIX Security Symposium
August 13 - 15, 2025. Seattle, WA, USA

Neil P. Thimmaiah
npemma2@uic.edu
University of Illinois Chicago
Chicago, USA

Yashashvi J. Dave
yashashvidave@gmail.com
University of Illinois Chicago
Chicago, USA

Rigel Gjomemo
rgjome1@uic.edu
Discovery Partners Institute
University of Illinois System
Chicago, USA

V.N. Venkatakrishnan
venkat@uic.edu
Discovery Partners Institute
University of Illinois System
Chicago, USA

Static Analysis Limitations

- **Taint-style vulnerabilities**
  - Malicious data flows from input locations (sources) → locations that carry out sensitive operations (sinks) in web applications
- **Detection**
  - Current Static Application Security Testing (SAST) tools use sophisticated techniques to find vulnerable paths
    - Symbolic execution
    - Constraint solving
  - Paths are then disclosed to developers through CVEs → they however focus on a single vulnerability instance
- **Limitations**
  - Insufficient information about vulnerabilities in the disclosures
  - Multiple false negatives in current analysis tools
  - Numerous vulnerabilities go undetected

..

Example

CVE-2024-25868

```
$membershipType = $_POST['membershipType'];
$membershipAmount = $_POST['membershipAmount'];

$insertQuery = "INSERT INTO membership_types
(type, amount) VALUES ('$membershipType',
$membershipAmount)";

echo "{$membershipType}";
```

Disclosed vulnerability

```
$fullname = $_POST['fullname'];
$address = $_POST['address'];

$updateQuery = "UPDATE members SET fullname
= '$fullname', address = '$address' WHERE id
= $memberId";

echo "{$row['fullname']}";
echo "{$row['address']}";
```

Similar undisclosed vulnerability

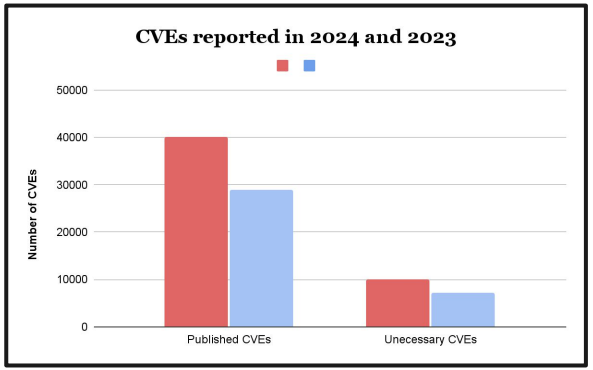
- Application CodeAstro Management contains a disclosed exploitable path
- Malicious input is inserted into the database and retrieved elsewhere
- Another similar exploitable path exists in the application too
- Vulnerability hasn't been reported or included in the original CVE

Problem: Have you detected all vulnerabilities?

→ “1 out of every 4 detected 0-day exploits could potentially have been avoided if a more thorough investigation and patching effort were explored”

→ “A correct patch is one that fixes a bug with complete accuracy, meaning the patch no longer allows any exploitation of the vulnerability”

Problem Statement: Given a disclosed vulnerability are there any **similar** undisclosed vulnerabilities?




GOOGLE PROJECT ZERO

Déjà vu-lnerability

A Year in Review of 0-days Exploited In-The-Wild in 2020 – Posted by Maddie Stone, Project Zero

25% of the 0-days detected in 2020 are closely related to previously publicly disclosed vulnerabilities.

Incomplete patches on on publicly disclosed vulnerabilities allow attackers to use 0-days against users with less effort because they can focus on the vulnerabilities already discussed and simply find a new bug in a code flow is both time a resource extensive.

When a 0-day is caught the wild it a gift for defenders to be put to how a much advantage of how attackers are exploiting publicly known vulnerabilities. This information aids in propering patching consistency and robustness, as well as detection methods.

When a 0-day is caught in the wild is a gift for Defenders need to take as much advantage of its ongoing around capabilities.

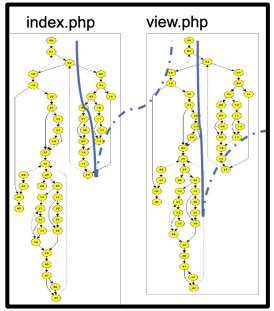
UIC DPI DISCOVERY PARTNERS INSTITUTE

PART OF THE UNIVERSITY OF ILLINOIS SYSTEM

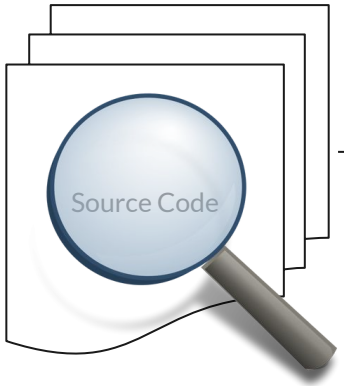
Solution: FInding eXploits from eXamples (FIXX)

CVE-2023-41593
CVE-2024-53604
CVE-2021-3298
CVE-2022-30478

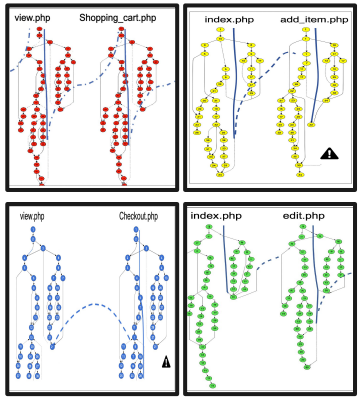
Disclosed Information



Exploitable Path



Similar Paths Discovery



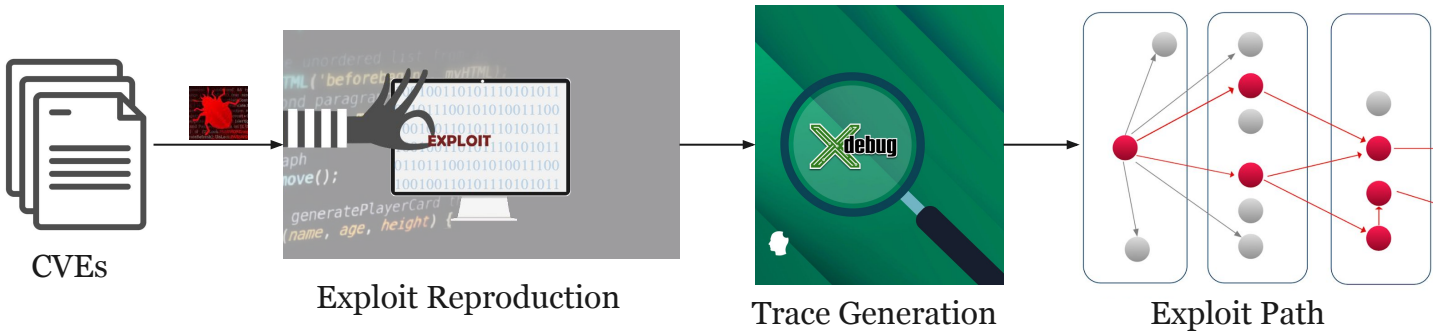
New Similar Exploits

Goal: Reduce false negatives in the detection of exploitable paths by static analysis and dynamic analysis

UIC DPI DISCOVERY PARTNERS INSTITUTE
PART OF THE UNIVERSITY OF ILLINOIS SYSTEM

Step I: Exploit Executioner

- CVE details are obtained and exploit is reproduced using a malicious payload
- A trace of the executed PHP code is obtained which contains the exploitable path from the source to sink
- An exploit dataflow path is extracted from the trace
- Control dependencies are identified to create a complete subgraph containing the dataflow



CVEs | Exploit Reproduction | Trace Generation | Exploit Path

Similarity Notion

- **Goal:** Find paths similar to the exploit path
- Similarity
  - Instruction similarity
  - Path similarity

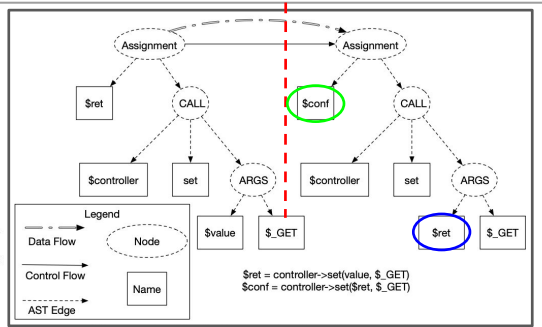
Instruction Similarity

Two instruction nodes of a code property graph are said to be *similar modulo X*, where X is a non-negative integer, if their AST subtrees have the same structure and node types, and there are X corresponding Name nodes that are different

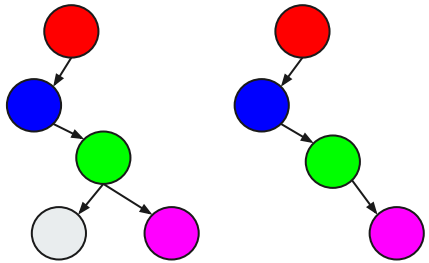
Path Similarity

A detected path is similar to the original exploit path if it has n% of instruction nodes that are *similar modulo X* and n is greater than or equal to the provided similarity threshold

Similarity Modulo 2 →

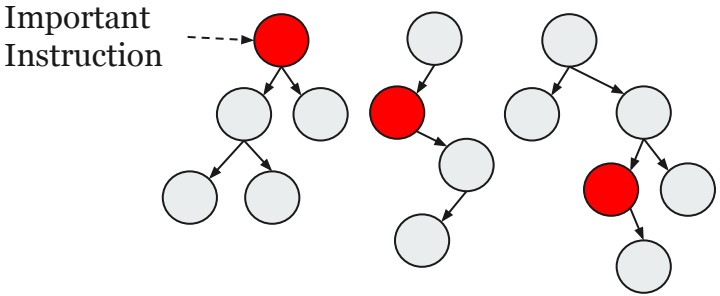


Instruction similarity 80% →

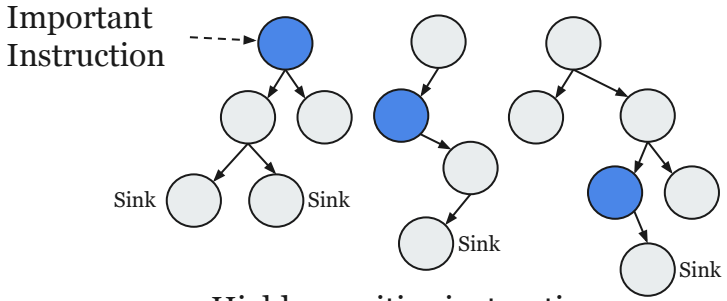


Search Focus: Metrics

- Goal: Focus search to similar **exploitable** paths
- Two novel focus instruction-level metrics
  - *Reusability Score* → The **repetition** of an instruction in the code
  - *Sensitivity Score* → The **cruciality** of an instruction towards an exploit
- *Selectivity Score* = *Reusability Score* x *Sensitivity Score* → final score for selecting important instructions



Highly reused instruction in multiple locations



Highly sensitive instruction towards a sink

Search Focus: Seed Identification

- FIXX extracts *seed* instructions from the exploitable path
  - Seeds are instructions with high reusability and sensitivity scores
- FIXX finds instructions that are similar to the seed instructions elsewhere in the application

$ret = controller->set(value, $_GET)

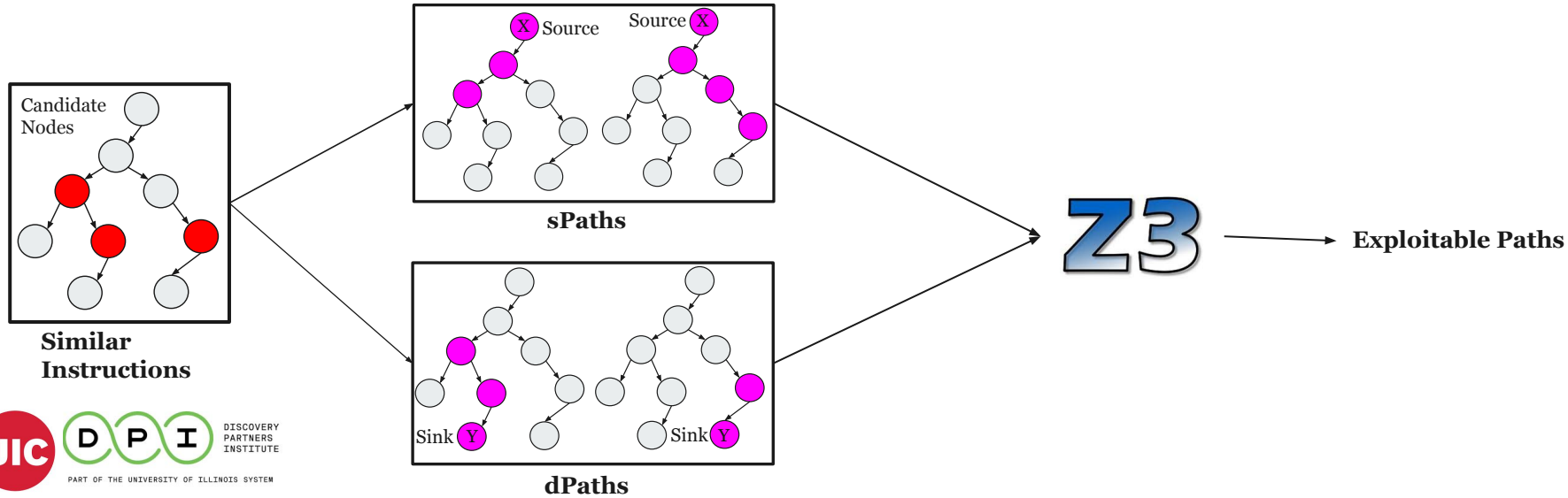
Seed Instruction

$conf = controller->set($ret, $_GET) (similar modulo 2)
$test = controller->set(value, $_GET) (similar modulo 2)
$id = controller->set(id, $_POST) (similar modulo 3)

Candidate Instructions similar to the seed

Step III: Detecting & Verifying Similar Paths

- FIXX uncovers paths similar to the exploitable paths using the path similarity measure
  - Detects sPaths (paths from sources to candidate instructions)
  - Detects dPaths (paths from candidate instructions to sinks)
- FIXX performs symbolic execution using inputs from an SMT solver
  - Verifies if the similar path is exploitable

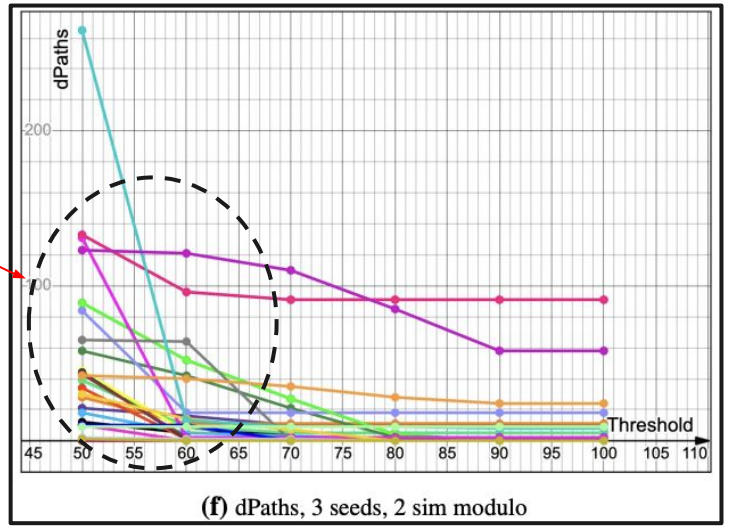


Evaluation

- Application selection
  - <1k to 255k LOCs and 10k PHP files
  - Popular applications
- 300 CVE entries in total
- 132 reproducible CVE entries
- 32 CVEs produce additional similar paths
- 10 new CVEs
- Comparison with vulnerability detection tools

Performance

- All applications with high average selectivity scores
  - Seeds are highly reused and sensitive towards exploit
  - Top seeds amongst ranked seeds contribute significantly to uncovering paths
- Example graph with the following parameters:
  - Seed Value → 3
  - Similarity Modulo → 2
  - Threshold → 50 - 100%
- Most paths are concentrated here…
  - ~ 60% path similarity
  - 3-4 seed values
  - 2 similarity modulo
- As the similarity threshold increases, the number of paths uncovered decreases
- On the other hand, as the similarity modulo increases the number of paths discovered increases



(f) dPaths, 3 seeds, 2 sim modulo

The number of similar paths uncovered by FIXX is **directly** proportional to the similarity modulo and **inversely** proportional to the similarity threshold

Results

- FIXX discovered 1097 exploitable paths in 19 PHP applications across 32 CVE reports
- 10 new CVE reports have been submitted to MITRE and published
- FIXX is quite robust
    - Original CVE can be re-discovered form the new CVE
- FIXX uncovers exploitable paths for both XSS and SQL Injection vulnerabilities in PHP web applications

Effectiveness

- We compare FIXX against Navex, RIPS and CloneDr
- FIXX is able to detect additional exploitable paths
- Approaches like RIPS and Navex detect a subset of these paths
- CloneDr is unable to detect similar blocks of code containing the exploitable path
- Outperforms RIPS, Navex and CloneDr

Application	Exp. Paths	Detected by RIPS	Detected by Navex
Collabtive98	9	0/9	2/9
Hospital Management System11	6	5/6	3/6
Dairy Farm Management93	33	32/33	6/33
CodeAstro68	38	6/38	0/38

Results: New CVEs

- Similar exploitable paths submitted to MITRE as CVEs
- 10 new CVEs have been published

Application	Original CVE	Sim. Nodes	sPaths	dPaths	Exp. Paths	Re-Discover Original CVE	New CVEs
Collabtive98	CVE-2021-3298 (XSS)	19	0	16	9	No	CVE-2024-46240 CVE-2024-48706 CVE-2024-48707 CVE-2024-48708
Hospital Management System11	CVE-2021-39411 (XSS)	144	6	171	6	Yes	CVE-2024-46237 CVE-2024-46238 CVE-2024-46239
Dairy Farm Management93	CVE-2023-41593 (XSS)	39	0	34	33	Yes	CVE-2024-46241
CodeAstro68	CVE-2024-25868 (XSS)	62	4	39	38	Yes	CVE-2024-46236 CVE-2024-48709

Takeaways

- Current SAST approaches use sophisticated techniques to find vulnerable paths
  - Undisclosed information in CVEs lead to numerous missing paths in applications
- FIXX:
  - Automatic approach that uses novel metrics to uncover all similar instances of exploitable paths
  - It can uncover similar exploitable paths
- FIXX has been evaluated on 32 CVE reports across 19 PHP applications
  - **1097 similar exploitable paths** have been uncovered
  - 10 new CVEs have been published
  - Similar path details are also available on our GitHub (https://github.com/neilthimmaiah01)
- Outperforms Navex, RIPS and CloneDr and serves as an efficient way of detecting paths
- FIXX demonstrates that it is possible to automatically uncover similar exploitable paths

UIC DPI DISCOVERY PARTNERS INSTITUTE
PART OF THE UNIVERSITY OF ILLINOIS SYSTEM

Questions…?

https://www.usenix.org/conference/usenixsecurity25/presentation/thimmaiah

Check out FIXX!

THANK YOU!

https://zenodo.org/records/15144145

Check out our Artifact!

Contact: Neil Thimmaiah
npemma2@uic.edu

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

UIC DPI DISCOVERY PARTNERS INSTITUTE
PART OF THE UNIVERSITY OF ILLINOIS SYSTEM

References

1. 2021. The MITRE Corporation: Common Vulnerabilities and Exposures.https://cve.mitre.org/.
2. Déjà vulnerability. https://googleprojectzero.blogspot.com /2021/02/deja-vu-lnerability.html, 2021. Accessed: 2021-11- 04
3. The neo4j graph platform – the #1 platform for connected data. https://neo4j.com/, 2018. Accessed: 2020-10-15.
4. FIXX - https://www.usenix.org/conference/usenixsecurity25/presentation/thimmaiah