Arbiter: Bridging the Static and Dynamic Divide in Vulnerability Discovery on Binary Programs

Jayakrishna Vadayath¹, Moritz Eckert², Kyle Zeng¹, Nicolaas Weideman³, Gokulkrishna Praveen Menon¹, Yanick Fratantonio⁴, Davide Balzarotti², Adam Doupé¹, Tiffany Bao¹, Ruoyu Wang¹, Christophe Hauser³, Yan Shoshitaishvili¹

¹Arizona State University  
²EURECOM  
³University of Southern California  
⁴Cisco Systems Inc
Fuzzing

Memory corruption

Sanitizer
Intuition

The properties of a vulnerability convey requirements on analysis techniques.

Vulnerability detection is a vulnerability-driven process.
Pros and Cons

Static
• High scalability
• High coverage
• More False Positives

Dynamic
• Limited scalability
• Limited coverage
• Fewer False Positives
Vulnerability Properties

We identified three properties that enable the composited use of analyses to achieve scalability and precision.

P1: Data-flow Sensitive Vulnerabilities. [DFSV]

P2: Easily Identifiable Sources or Sinks. [EISS]

P3: Control-flow-determined Aliasing. [CFDA]
The Analyses

Allows us to use the high scalability and high coverage of static analysis to identify candidate paths.

Provides high precision by enabling the use of Under-Constrained Symbolic Execution (UCSE) to filter out false-positives.

Supports an adaptive augmentation of context sensitivity in UCSE, providing a configurable trade-off between precision and soundness.
Arbiter

Static identification of slices

Under-Constrained Symbolic Execution

Static data flow recovery

Adaptive False Positive Reduction
# Property-Compliant Vulnerabilities

<table>
<thead>
<tr>
<th>CWE ID</th>
<th>Description</th>
<th>CVE Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
<td>CVE-2018-18311</td>
</tr>
<tr>
<td>CWE-134</td>
<td>Use of Externally-Controlled Format String</td>
<td>CVE-2012-0809</td>
</tr>
<tr>
<td>CWE-252</td>
<td>Unchecked Return Value</td>
<td>CVE-2013-4559</td>
</tr>
<tr>
<td>CWE-337</td>
<td>Predictable Seed in Pseudo-Random Number Generator</td>
<td>CVE-2020-13784</td>
</tr>
</tbody>
</table>
Evaluation

- Evaluated on 76k binaries, generated a total of 1130 alerts.
- 661 True Positives, 410 False Positives.

<table>
<thead>
<tr>
<th>Class</th>
<th>Alerts</th>
<th>True Positives</th>
<th>False Positives</th>
<th>Untriage-able</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-131</td>
<td>436</td>
<td>194</td>
<td>195</td>
<td>47</td>
</tr>
<tr>
<td>CWE-134</td>
<td>158</td>
<td>12</td>
<td>142</td>
<td>4</td>
</tr>
<tr>
<td>CWE-252</td>
<td>159</td>
<td>83</td>
<td>71</td>
<td>5</td>
</tr>
<tr>
<td>CWE-337</td>
<td>377</td>
<td>372</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Validation of Static Analysis

Evaluation vs AFL
We triggered 25 bugs discovered by Arbiter. Of those:

• Only 7 were in standalone binaries.
• AFL could fuzz 5 (2 binaries were too complex).
• AFL only found 3 bugs in 24 hours.
Case Study: OCaml

In triaging Arbiter's CWE-131 reports, we kept finding identical cases of buggy heap management code.

Looking into the corresponding source code, there were no bugs. All of the programs were OCaml...

The bug was introduced at compile time by the OCaml compiler, affected all 32-bit OCaml programs.
Evaluation on The Juliet Dataset

- True Positive
- False Positive
- New Positive

Percentage

0 10 20 30 40 50 60 70 80
Case Study: The Juliet Dataset

New Bugs!
Arbiter found 190 previously-unknown bugs in Juliet, missed by all existing work!

The Cause?
Source code analyzer modeling of the abs() C function is (apparently) wrong.
Summary

• Vulnerability detection is a vulnerability-driven process.

• Three vulnerability properties enable hybrid analysis, which improves scalability and precision.

• Evaluation on 76k binaries shows ~60% true positive rate.
Thank you!

Questions?

https://github.com/jkrshnmenon/arbiter

Jayakrishna (Jay) Vadayath

@jkrshnmenon

jkrshnmenon@asu.edu

jkrshnmenon
Backup slides
The pattern: Arbiter casts an initial wide net, followed by alert reductions up to 10x per step.

* Final alerts can be higher than prior steps due to paths that are traced to program entry point during earlier false positive reduction steps.

<table>
<thead>
<tr>
<th></th>
<th>Context-Insensitive UCSE</th>
<th>Context-Sensitive UCSE #1</th>
<th>Context-Sensitive UCSE #2</th>
<th>Context-Sensitive UCSE #3</th>
<th>Final Alerts*</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-131</td>
<td>692,501</td>
<td>31,436</td>
<td>3,310</td>
<td>1,037</td>
<td>351</td>
</tr>
<tr>
<td>CWE-134</td>
<td>429,711</td>
<td>39,305</td>
<td>4,894</td>
<td>489</td>
<td>222</td>
</tr>
<tr>
<td>CWE-252</td>
<td>55,388</td>
<td>4,413</td>
<td>942</td>
<td>126</td>
<td>107</td>
</tr>
<tr>
<td>CWE-337</td>
<td>4,641</td>
<td>2,741</td>
<td>633</td>
<td>92</td>
<td>45</td>
</tr>
</tbody>
</table>