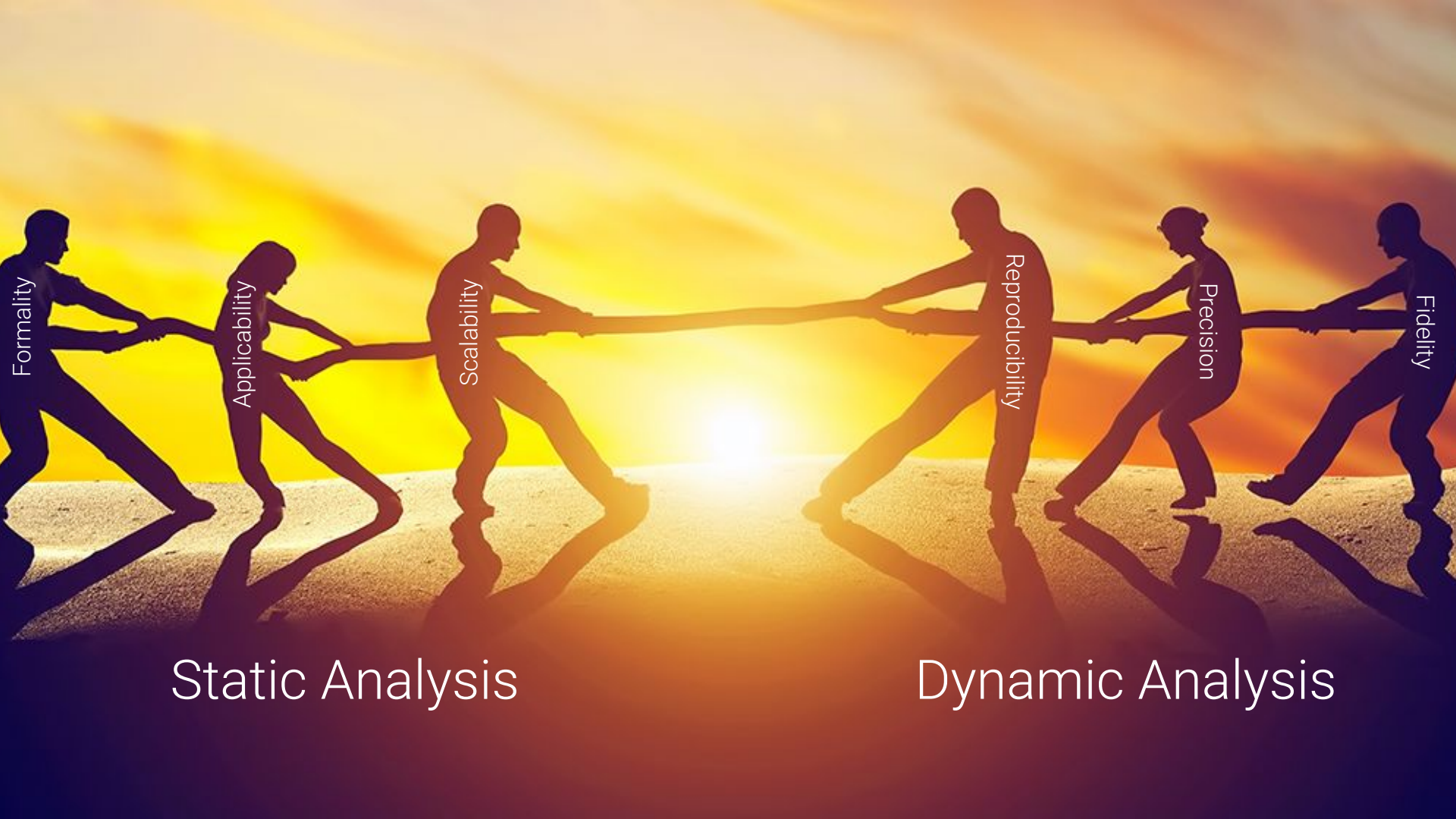
# Arbiter: Bridging the Static and Dynamic Divide in Vulnerability Discovery on Binary Programs

Jayakrishna Vadayath[1], Moritz Eckert[2], Kyle Zeng[1], Nicolaas Weideman[3], Gokulkrishna Praveen Menon[1], Yanick Fratantonio[4], Davide Balzarotti[2], Adam Doupé[1], Tiffany Bao[1], Ruoyu Wang[1], Christophe Hauser[3], Yan Shoshitaishvili[1]

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





Formality
Applicability
Scalability
Reproducibility
Precision
Fidelity
Static Analysis
Dynamic Analysis

# Fuzzing

# 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.** 


**P2: Easily Identifiable Sources or Sinks.** 


**P3: Control-flow-determined Aliasing.** 

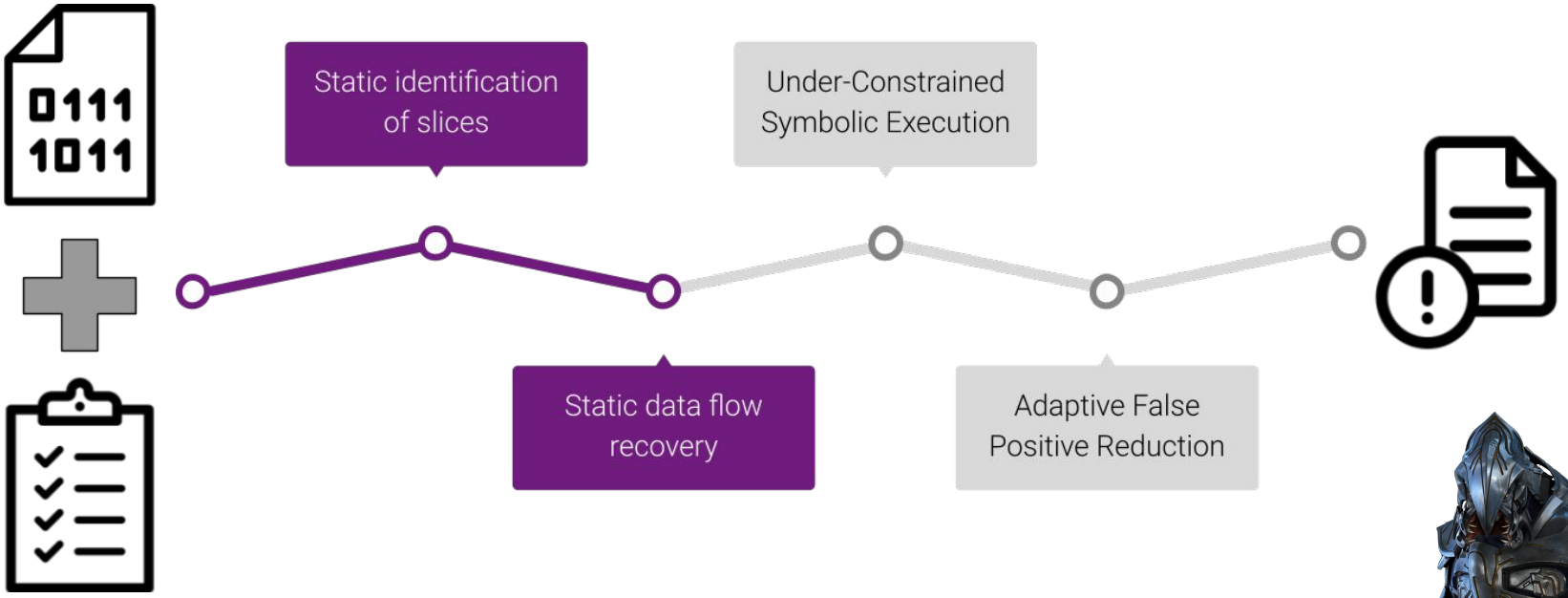
# The Analyses

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

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

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

# Arbiter


0111
1011
Static identification of slices
Under-Constrained Symbolic Execution
Static data flow recovery
Adaptive False Positive Reduction

# Property-Compliant Vulnerabilities

| CWE ID | Description | CVE Example |
|---|---|---|
| CWE-131 | Incorrect Calculation of Buffer Size | CVE-2018-18311 |
| CWE-134 | Use of Externally-Controlled Format String | CVE-2012-0809 |
| CWE-252 | Unchecked Return Value | CVE-2013-4559 |
| CWE-337 | Predictable Seed in Pseudo-Random Number Generator | CVE-2020-13784 |

# Evaluation

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

| Class | Alerts | True Positives | False Positives | Untriage-able |
|---|---|---|---|---|
| CWE-131 | 436 | 194 | 195 | 47 |
| CWE-134 | 158 | 12 | 142 | 4 |
| CWE-252 | 159 | 83 | 71 | 5 |
| CWE-337 | 377 | 372 | 2 | 3 |

# 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

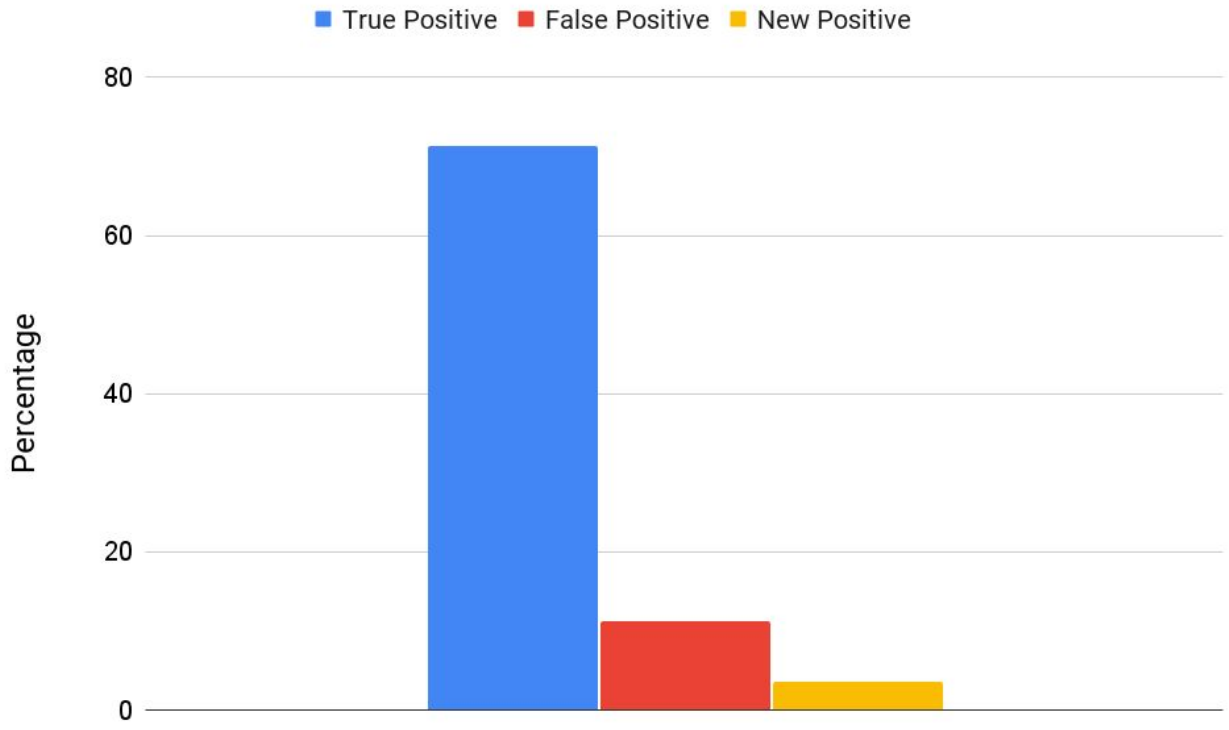
In triaging Arbiter's CWE-131 reports, we kept finding identical cases of buggy heap management code.

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.

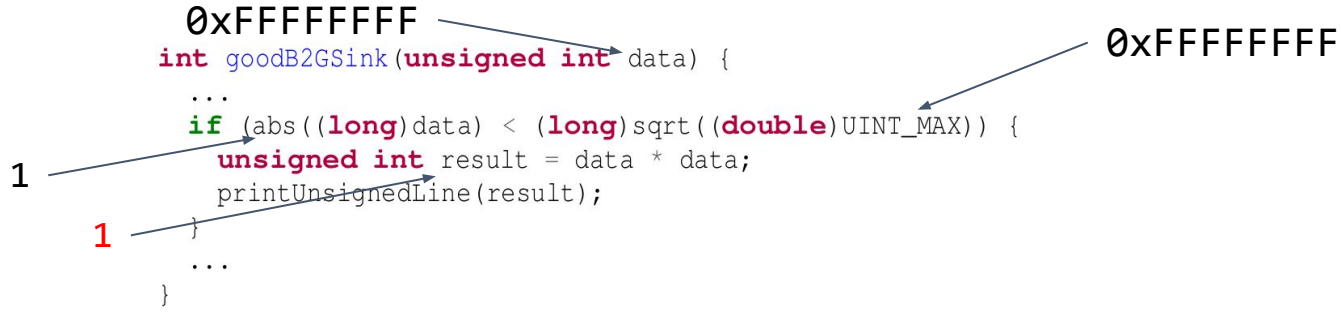
# Evaluation on The Juliet Dataset



# 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

# Digging In

| | Static Detections | Context-Insensitive UCSE | Context-Sensitive UCSE #1 | Context-Sensitive UCSE #2 | Context-Sensitive UCSE #3 | Final Alerts* |
|---|---|---|---|---|---|---|
| CWE-131 | 692,501 | 31,436 | 3,310 | 1,037 | 351 | 436 |
| CWE-134 | 429,711 | 39,305 | 4,894 | 489 | 222 | 158 |
| CWE-252 | 55,388 | 4,413 | 942 | 126 | 107 | 159 |
| CWE-337 | 4,641 | 2,741 | 633 | 92 | 45 | 377 |

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.