RUHR UNIVERSITÄT BOCHUM RUB

UNIVERSITY OF BIRMINGHAM

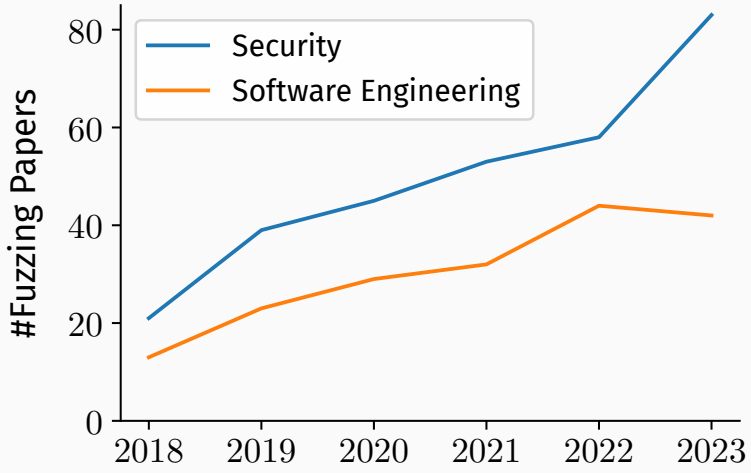
CISPA

SoK: Prudent Evaluation Practices for Fuzzing

Moritz Schloegel, Nils Bars, Nico Schiller, Lukas Bernhard, Tobias Scharnowski, Addison Crump, Arash Ale Ebrahim, Nicolai Bissantz, Marius Muench, and Thorsten Holz

CISPA Helmholtz Center for Information Security
Ruhr University Bochum
University of Birmingham

Fuzzing Papers are Still Popular



Are we evaluating fuzzers right?

SoK: Prudent Evaluation Practices for Fuzzing

Moritz Schloegel[1], Nils Bars[1], Nico Schiller[1], Lukas Bernhard[1], Tobias Scharnowski[1]
Addison Crump[1], Arash Ale-Ebrahim[1], Nicolai Bissantz[2], Marius Muench[3], Thorsten Holz[1]

[1]*CISPA Helmholtz Center for Information Security, {first.lastname}@cispa.de*
[2]*Ruhr University Bochum, nicolai.bissantz@ruhr-uni-bochum.de*
[3]*University of Birmingham, m.muench@bham.ac.uk*

***Abstract*—Fuzzing has proven to be a highly effective approach to uncover software bugs over the past decade. After AFL popularized the groundbreaking concept of lightweight coverage**

1. Introduction

Fuzzing, a portmanteau of "fuzz testing", has gained

Study reproducibility

Goals

Study reproducibility

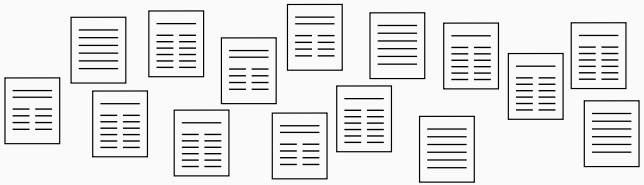
... and other evaluation pitfalls

Approach

1. Find all fuzzing papers on 7 top-tier venues between 2018 and 2023

Approach

1. Find all fuzzing papers on 7 top-tier venues between 2018 and 2023



Approach

1. Find all fuzzing papers on 7 top-tier venues between 2018 and 2023
2. Literature analysis of 150 of these papers

Approach

1. Find all fuzzing papers on 7 top-tier venues between 2018 and 2023
2. Literature analysis of 150 of these papers
3. Case studies: Reproducing experiments of 8 papers

Approach

1. Find all fuzzing papers on 7 top-tier venues between 2018 and 2023
2. Literature analysis of 150 of these papers
3. Case studies: Reproducing experiments of 8 papers
4. Update recommendations (where needed)

How to make sure fuzzing is reproducible?

Klees et al. – “Evaluating Fuzz Testing”, ACM CCS, 2018

Metzmann et al. – “FuzzBench: An Open Fuzzer Benchmarking Platform and Service”, ESEC/FSE, 2021

Böhme et al. – On the Reliability of Coverage Testing, ICSE, 2022

How to make sure fuzzing is reproducible?

Klees et al. – “Evaluating Fuzz Testing”, ACM CCS, 2018

Metzmann et al. – “FuzzBench: An Open Fuzzer Benchmarking Platform and Service”, ESEC/FSE, 2021

Böhme et al. – On the Reliability of Coverage Testing, ICSE, 2022

Disclaimer

No intention of finger pointing!

Recommendations

1. Document setup and parameters
2. Sample relevant targets
3. Pick a good baseline
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. Conduct a statistical evaluation

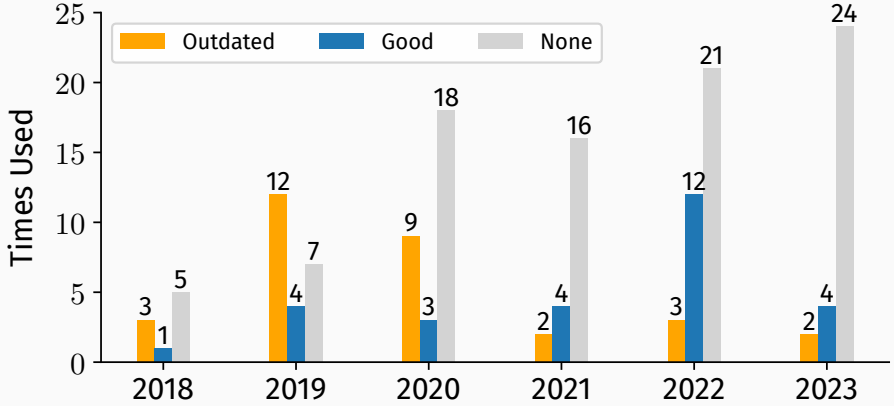
Recommendations

1. **Document setup and parameters**
2. Sample relevant targets
3. Pick a good baseline
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. Conduct a statistical evaluation

Recommendations

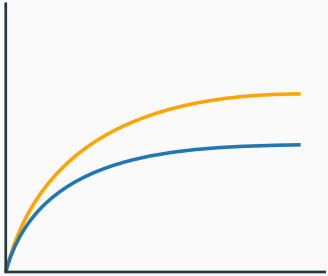
1. Document setup and parameters
2. **Sample relevant targets**
3. Pick a good baseline
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. Conduct a statistical evaluation

Benchmarks


Outdated
Good
None
Times Used
25
20
15
10
5
0
3
1
5
12
4
7
9
3
18
2
4
16
3
12
21
2
4
24
2018
2019
2020
2021
2022
2023

Recommendations

1. Document setup and parameters
2. Sample relevant targets
3. **Pick a good baseline**
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. Conduct a statistical evaluation

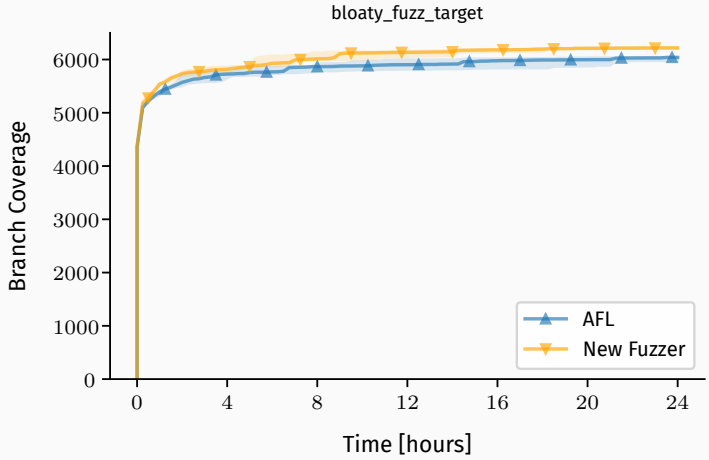


❸ Pick a Good Baseline

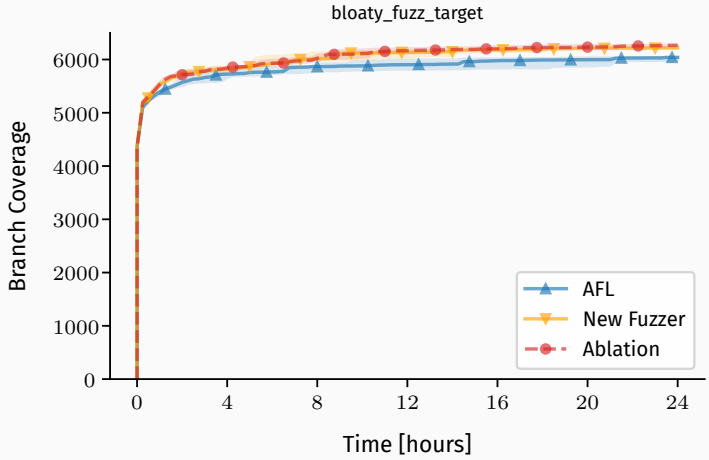
A new fuzzer that proposes to:

1. *Dynamically* adapt probabilities with which mutations are chosen
2. Use an *evolutionary strategy* to optimize these probabilities

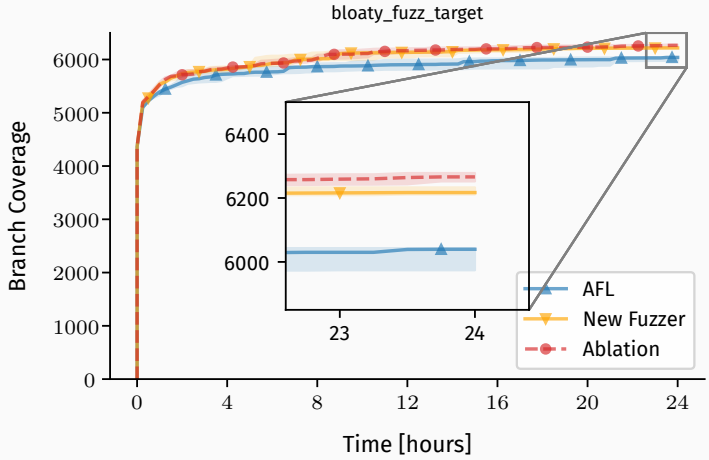
❸ Pick a Good Baseline



❸ Pick a Good Baseline



❸ Pick a Good Baseline

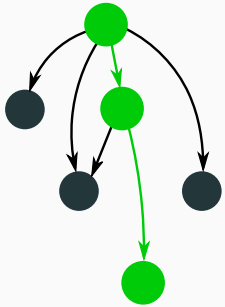


❸ Pick a Good Baseline

Lesson learned: ablation studies are important

Recommendations

1. Document setup and parameters
2. Sample relevant targets
3. Pick a good baseline
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. Conduct a statistical evaluation

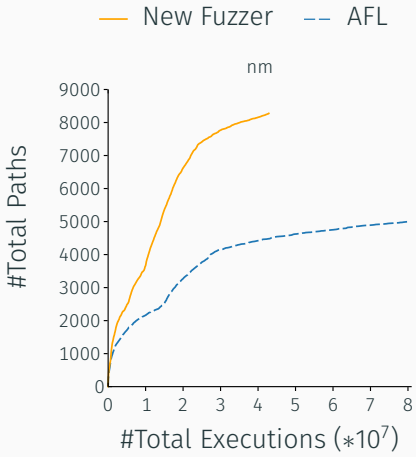


❹ Suitable Metrics: Code Coverage

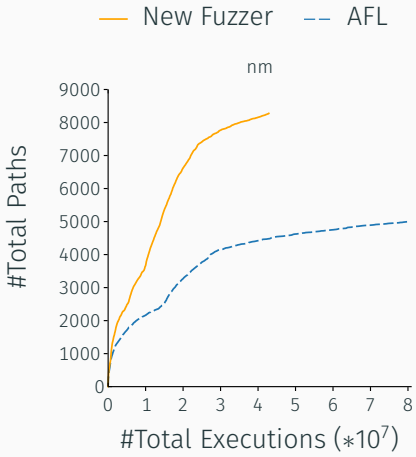
New fuzzer:

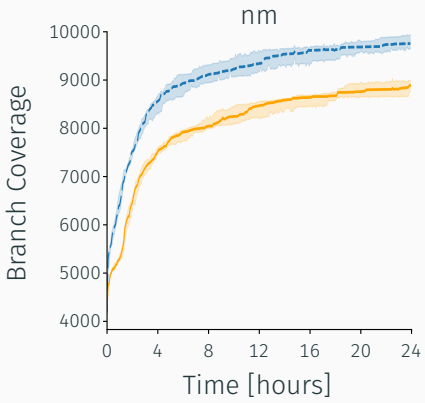
- Improves input scheduling
- Aims to covers more *paths* with fewer inputs

❹ Suitable Metrics: Code Coverage



❹ Suitable Metrics: Code Coverage





❹ Suitable Metrics: Code Coverage

Lesson learned: new metrics may mislead readers

$\Rightarrow$ include known metrics!

Recommendations

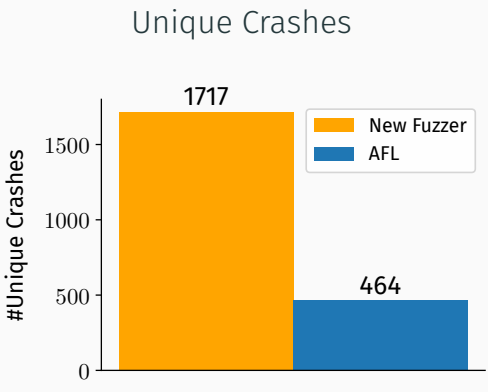
1. Document setup and parameters
2. Sample relevant targets
3. Pick a good baseline
4. **Choose suitable evaluation metrics**
   - Code coverage
   - **Bugs**
5. Conduct a statistical evaluation

❹ Suitable Metrics: Unique Crashes

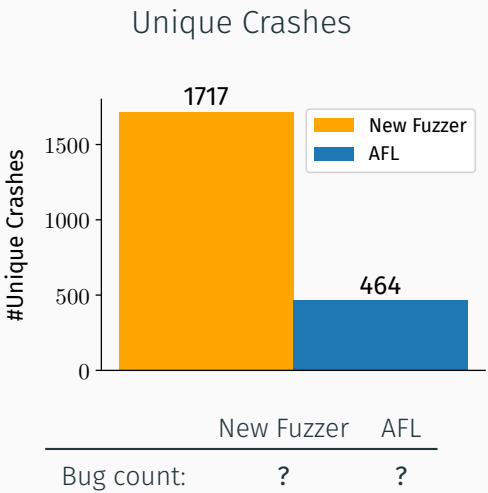
A new fuzzer that:

- Proposes memory usage as additional feedback
- Uses *unique crashes* as a metric

❹ Suitable Metrics: Unique Crashes

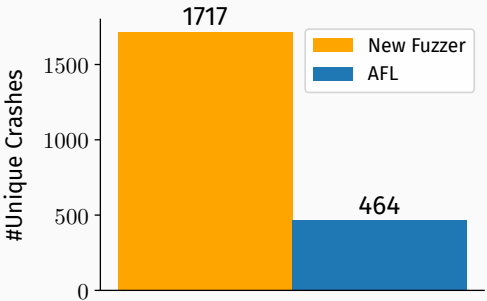


❹ Suitable Metrics: Unique Crashes

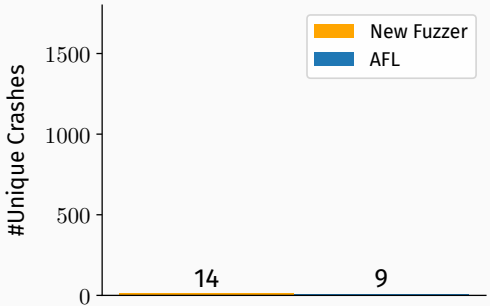


❹ Suitable Metrics: Unique Crashes

Unique Crashes

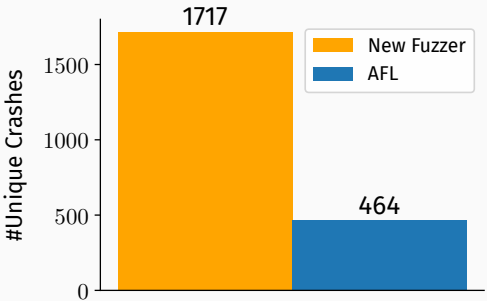


… after one patch

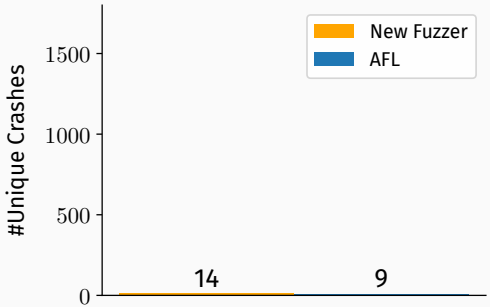


❹ Suitable Metrics: Unique Crashes

Unique Crashes



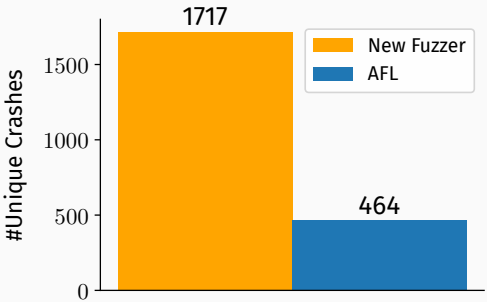
… after one patch



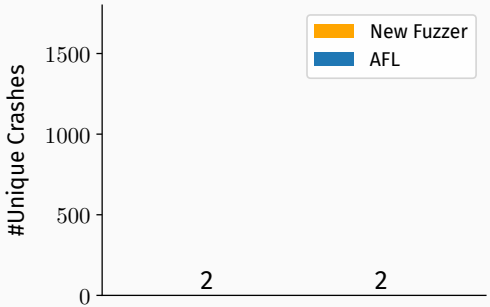
	New Fuzzer	AFL
Bug count:	1+?	1+?

❹ Suitable Metrics: Unique Crashes

Unique Crashes

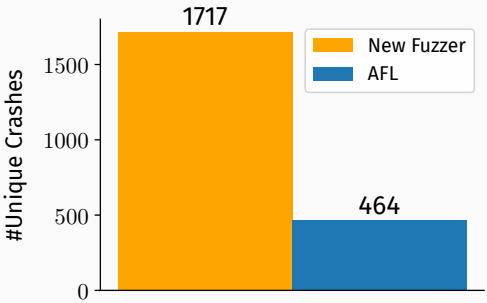


...after manual deduplication

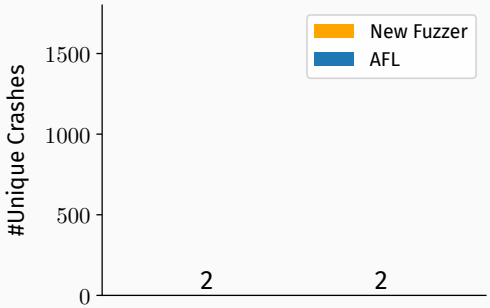


❹ Suitable Metrics: Unique Crashes

Unique Crashes



...after manual deduplication



	New Fuzzer	AFL
Bug count:	2	2

❹ Suitable Metrics: Unique Crashes

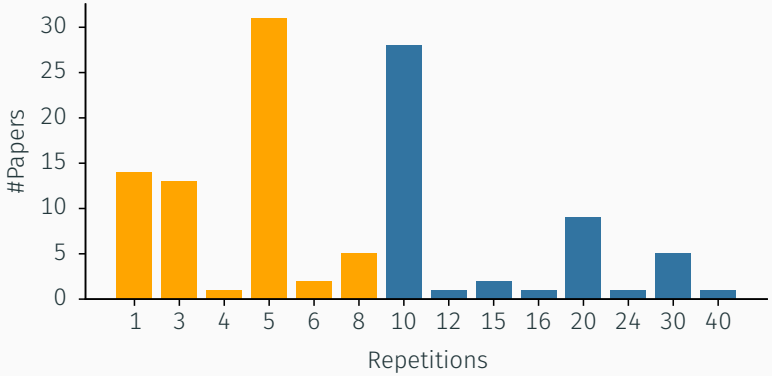
Lesson learned: “unique” crashes $\neq$ actual bugs

$\Rightarrow$ need deduplication or should use actual bugs

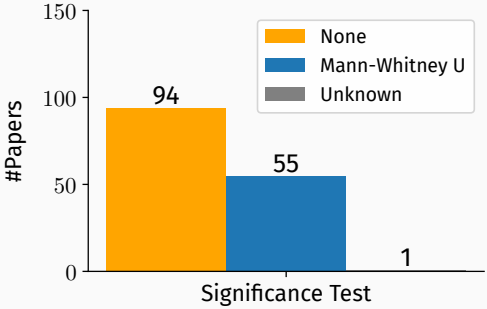
Recommendations

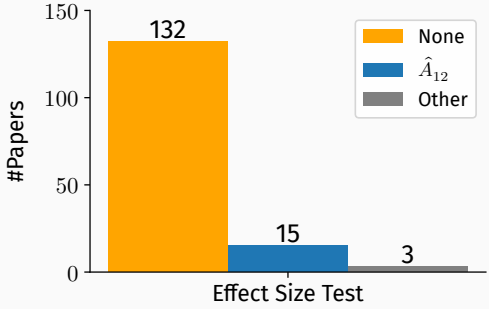
1. Document setup and parameters
2. Sample relevant targets
3. Pick a good baseline
4. Choose suitable evaluation metrics
   - Code coverage
   - Bugs
5. **Conduct a statistical evaluation**

❺ Statistically evaluate results



❺ Statistically evaluate results





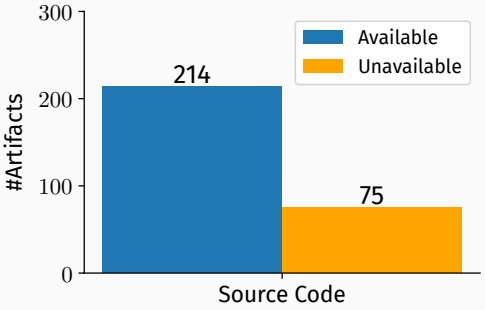
Recommendations

There's more beyond the evaluation itself:

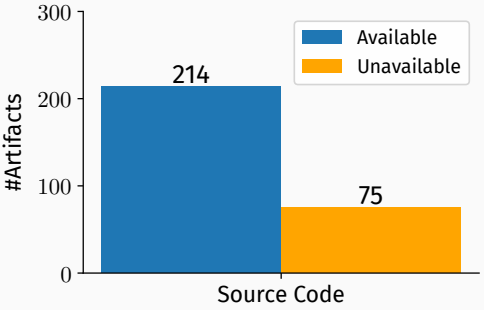
- What about the fuzzer source code?
- What about new bugs found during the evaluation?

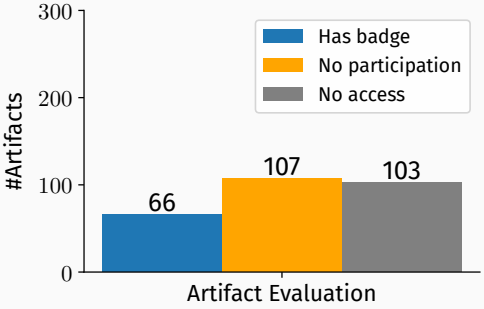
Beyond the paper: artifact availability

Artifact Availability



Artifact Availability





Artifact Availability

Good: much code is openly available!

BUT: low artifact evaluation participation

Recommendations

There's more beyond the evaluation itself:

- What about the fuzzer source code?
- **What about new bugs found during the evaluation?**

What happens to found bugs?

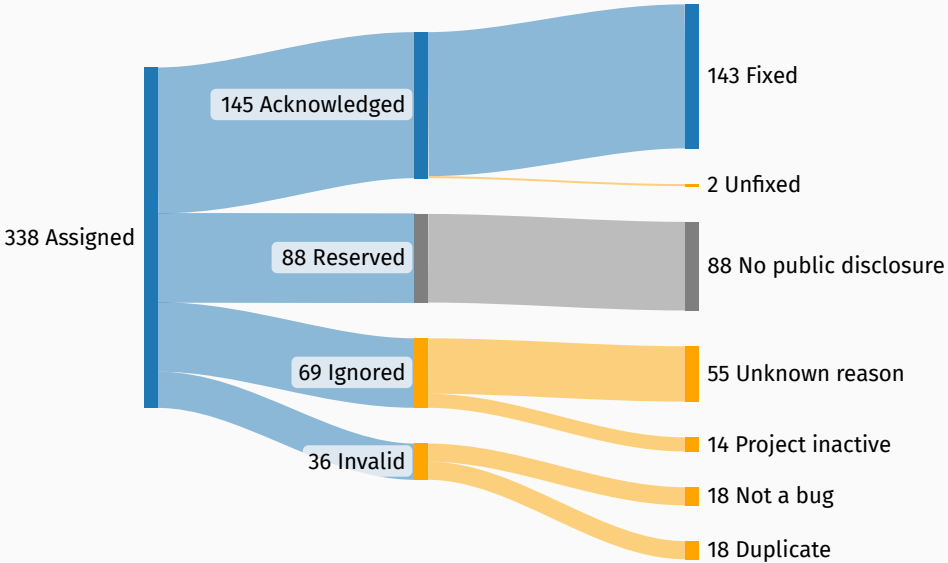
What happens to found bugs?

$\Rightarrow$ Responsible disclosure?

Approach

❶ Look for CVEs in fuzzing papers

❷ Check their outcome


338 Assigned
145 Acknowledged
88 Reserved
69 Ignored
36 Invalid
143 Fixed
2 Unfixed
88 No public disclosure
55 Unknown reason
14 Project inactive
18 Not a bug
18 Duplicate

Misaligned incentives

Misaligned incentives & no verification

Misaligned incentives & no verification

$\Rightarrow$ Easy to game the system

Recommendations

❶ Document setup and parameters

❷ Sample relevant targets

❸ Pick a good baseline

❹ Choose suitable evaluation metrics
- Code coverage
- Bugs

❺ Conduct a statistical evaluation

\+ Artifact availability

\+ CVE misuse

$\Rightarrow$ guidelines

Guidelines on GitHub

README

Fuzzing Evaluation Guidelines

Current version: 1.0.3

Proposals for changes welcome (please open an issue for discussion or a pull request for changes).

DISCLAIMER: These items represent are a best-effort attempt at capturing action items to follow during the evaluation of a scientific paper that focuses on fuzzing. **They do not apply universally to all fuzzing methods - in certain scenarios, techniques may wish to deviate for good reason from these guidelines. In any case, a case-by-case judgment is necessary.** The guidelines do not discuss many malicious choices that immediately negate any chance of a fair evaluation, such as giving your fuzzer an unfair advantage (e.g., by fine-tuning the fuzzer or its targets) or putting other fuzzers at a disadvantage.

A. Preparation for Evaluation

1. Find relevant tools and baselines to compare against
    - 1.1 Include state-of-the-art techniques from both academia and industry
    - 1.2 If your fuzzer is based on an existing fuzzer, include the baseline (to measure the delta of your changes, which allows attributing improvements to your technique)
    - 1.3 Use recent versions of fuzzers
    - 1.4 If applicable, derive a baseline variant of your technique that replaces core contributions by alternatives. For example, consider using a variant that replaces an informed algorithm with randomness.

Summary

1. Fuzzing evaluations are hard to get right
2. Join artifact evaluation
3. Help us shape the guidelines

Paper

Guidelines