CUPID: Automatic Fuzzer Selection for Collaborative Fuzzing

Annual Computer Security Applications Conference (ACSAC), Virtual

December 10, 2020

Emre Güler*, Philipp Görz*, Elia Geretto†, Andrea Jemmett†, Sebastian Österlund†, Herbert Bos†, Cristiano Giuffrida†, and Thorsten Holz*

*Chair for Systems Security
Ruhr-Universität Bochum

†Systems and Network Security Group
Vrije Universiteit Amsterdam
Motivation
The Necessity for Collaborative Fuzzing

- Goal: find bugs
The Necessity for Collaborative Fuzzing

- Goal: find bugs
- Scenario: maybe once, maybe periodically
The Necessity for Collaborative Fuzzing

• Goal: find bugs
• Scenario: maybe once, maybe periodically
• Dedicated machine for fuzzing / limited resources
The Necessity for Collaborative Fuzzing

- Goal: find bugs
- Scenario: maybe once, maybe periodically
- Dedicated machine for fuzzing / limited resources
  - i.e. find as many bugs as possible, in the shortest possible time

Answer: parallel fuzzing!

But which fuzzers?
The Necessity for Collaborative Fuzzing

• Goal: find bugs
• Scenario: maybe once, maybe periodically
• Dedicated machine for fuzzing / limited resources
  • i.e. find as many bugs as possible, in the shortest possible time
• Answer: parallel fuzzing!
The Necessity for Collaborative Fuzzing

- Goal: find bugs
- Scenario: maybe once, maybe periodically
- Dedicated machine for fuzzing / limited resources
  - i.e. find as many bugs as possible, in the shortest possible time
- Answer: parallel fuzzing!
- But ... which fuzzers?
Collaborative Fuzzing
Why not 4 x AFL?

• What if AFL is bad at a particular problem?
Why not 4 x AFL?

- What if AFL is bad at a particular problem?
- More of the same might not be better
Why not 4 x AFL?

- What if AFL is bad at a particular problem?
- More of the same might not be better
- EnFuzz: use a diverse set of fuzzers
Complementary Fuzzers
Complementary Fuzzers

LIBFUZZER

HONGGFUZZ

6
Complementary Fuzzers

LIBFUZZER  HONGGFUZZ

LIBFUZZER  HONGGFUZZ

LIBFUZZER

LIBFUZZER

+LIBFUZZER  LIBFUZZER

+HONGGFUZZ  HONGGFUZZ
Complementary Fuzzers

LIBFUZZER + LIBFUZZER

LIBFUZZER + HONGGFUZZ

HONGGFUZZ + HONGGFUZZ
Cupid
• No human expert needed
Improvements

• No human expert needed
• Automatically collect performance data on single, isolated fuzzers
Improvements

• No human expert needed
• Automatically collect performance data on single, isolated fuzzers
• Linear, instead of exponential growth
Improvements

• No human expert needed
• Automatically collect performance data on single, isolated fuzzers
• Linear, instead of exponential growth
• Target-independent: predict one fuzzer combination, use on all future binaries
Data Collection
Data Collection

• Find fuzzers that are complementary.
Data Collection

• Find fuzzers that are complementary.
  • Fuzzers are complementary when they solve different branches
Data Collection

• Find fuzzers that are complementary.
  • Fuzzers are complementary when they solve different branches
• Ideal, theoretical scenario:
Data Collection

• Find fuzzers that are complementary.
  • Fuzzers are complementary when they solve different branches
• Ideal, theoretical scenario:
  • Test fuzzer on all possible branches
Data Collection

• Find fuzzers that are complementary.
  • Fuzzers are complementary when they solve different branches
• Ideal, theoretical scenario:
  • Test fuzzer on all possible branches
  • Compare two fuzzers on the same branches: how often did they solve different branches?
Data Collection

- Find fuzzers that are complementary.
  - Fuzzers are complementary when they solve different branches
- Ideal, theoretical scenario:
  - Test fuzzer on all possible branches
  - Compare two fuzzers on the same branches: how often did they solve different branches?
- Real-world scenario:
Data Collection

• Find fuzzers that are complementary.
  • Fuzzers are complementary when they solve different branches.

• Ideal, theoretical scenario:
  • Test fuzzer on all possible branches.
  • Compare two fuzzers on the same branches: how often did they solve different branches?

• Real-world scenario:
  • Find diverse set of binaries.
Data Collection

- Find fuzzers that are *complementary*.  
  - Fuzzers are complementary when they solve *different branches*
- Ideal, theoretical scenario:
  - Test fuzzer on *all* possible branches  
  - Compare two fuzzers on the same branches: how often did they solve different branches?
- Real-world scenario:
  - Find diverse set of binaries  
  - Fuzz binaries for a short time, but many times
Data Collection

• Find fuzzers that are *complementary*.
  • Fuzzers are complementary when they solve *different branches*

• Ideal, theoretical scenario:
  • Test fuzzer on *all* possible branches
  • Compare two fuzzers on the same branches: how often did they solve different branches?

• Real-world scenario:
  • Find diverse set of binaries
  • Fuzz binaries for a short time, but many times
  • Use different seeds to explore more of the program space
Overview

![Diagram of Cupid and Fuzzing Process]

- **Seeds**
- **Fuzzers**
- **Training Binaries**

**Offline phase**
- Cupid
- Prediction Engine

**Online phase**
- Shared Corpus
- Fuzzing Target

13
Evaluation
• How well does CUPID predict real-world rankings?
Evaluation

• How well does CUPID predict real-world rankings?
  • Strong positive correlation
Evaluation

• How well does CUPID predict real-world rankings?
  • Strong positive correlation
Evaluation

• How well does CUPID predict real-world rankings?
  • Strong positive correlation
• Does the automatic selection by CUPID beat two different combinations by ENFuzz?
Evaluation

• How well does CUPID predict real-world rankings?
  • Strong positive correlation
• Does the automatic selection by CUPID beat two different combinations by ENFuzz?
  • Yes: slightly more code coverage, faster, more bugs.
Conclusion
Conclusion

- Use complementary fuzzers to maximize return on investment given fixed resources
Conclusion

• Use complementary fuzzers to maximize return on investment given fixed resources
• Data on single, isolated fuzzer runs can be used to predict well-performing fuzzer combinations
Conclusion

- Use complementary fuzzers to maximize return on investment given fixed resources
- Data on single, isolated fuzzer runs can be used to predict well-performing fuzzer combinations
- Data-driven approach of CUPID beats expert guided fuzzer picks
Thanks!

Emre Güler

emre.xguler

emre.gueler@rub.de

www.emre.re

https://github.com/RUB-SysSec/cupid