Poster: *HyPFuzz*: Formal-Assisted Processor Fuzzing

Chen Chen†, Rahul Kande†, Nathan Nguyen†, Flemming Andersen†, Aakash Tyagi†,
Ahmad-Reza Sadeghi*, and Jeyavijayan Rajendran†
†Texas A&M University, USA, *Technische Universität Darmstadt, Germany
†{chenc, rahulkande, nathan.tm.nguyen, flandersen, tyagi, jv.rajendran}@tamu.edu,
*{ahmad.sadeghi}@trust.tu-darmstadt.de

Abstract

Recent research has shown that hardware fuzzers can effectively detect security vulnerabilities in modern processors. However, existing hardware fuzzers do not fuzz well the hard-to-reach design spaces. Consequently, these fuzzers cannot effectively fuzz security-critical control- and data-flow logic in the processors, hence missing security vulnerabilities.

To tackle this challenge, we present *HyPFuzz*, a hybrid fuzzer that leverages formal verification tools to help fuzz the hard-to-reach part of the processors. To increase the effectiveness of *HyPFuzz*, we perform optimizations in time and space. First, we develop a scheduling strategy to prevent under- or over-utilization of the capabilities of formal tools and fuzzers. Second, we develop heuristic strategies to select points in the design space for the formal tool to target. We evaluate *HyPFuzz* on five widely-used open-source processors. *HyPFuzz* detected all the vulnerabilities detected by the most recent processor fuzzer and found three new vulnerabilities that were missed by previous extensive fuzzing and formal verification. This led to two new common vulnerabilities and exposures (CVE) entries. *HyPFuzz* also achieves $11.68\times$ faster coverage than the most recent processor fuzzer.

I. Main Content

This research [1] is recently published in USENIX Security 2023. The original abstract and author list are shown above. We post the paper link with the conference version[1].

References

[1] C. Chen, R. Kande, N. Nguyen, F. Andersen, A. Tyagi, A.-R. Sadeghi, and J. Rajendran, "HyPFuzz: Formal-Assisted Processor Fuzzing," *USENIX Security Symposium*, pp. 1361–1378, 2023.

[1] https://www.usenix.org/conference/usenixsecurity23/presentation/chen-chen

HyPFuzz: Formal-Assisted Processor Fuzzing

TEXAS A&M UNIVERSITY
TECHNISCHE UNIVERSITÄT DARMSTADT

Chen Chen†, Rahul Kande†, Nathan Nyugen†, Flemming Andersen†, Aakash Tyagi†, Ahmad-Reza Sadeghi*, and Jeyavijayan Rajendran†
†Texas A&M University, *Technische Universitat Darmstadt

intel
usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Introduction

- Hardware vulnerabilities are difficult to be patched.
- Hardware vulnerabilities emerge at an alarming rate.

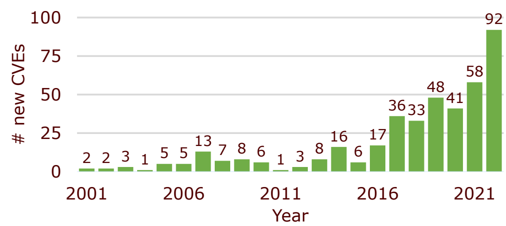


Figure 1. Exponential increase in the number of hardware vulnerabilities [1].

Research Objective

- *HyPFuzz*, a hybrid fuzzer that leverages formal verification tools to help fuzz the hard-to-cover part of the processors.

Technique	Automated	Scalable	Efficient	Coverage
Manual Inspection	✗	✗	✗	✗
Formal verification	✗	✗	✓	✗
Random regression	✓	✓	✗	✗
Hardware Fuzzing	✓	✓	✓	✗
HyPFuzz	✓	✓	✓	✓

Table 1. Comparison between various hardware verification techniques.

Methodology

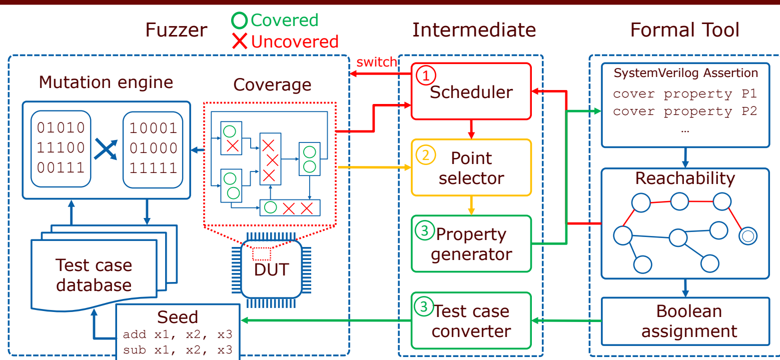


Figure 2. Framework of *HyPFuzz*.

Challenge 1: Scheduling of Fuzzer and Formal Tool

- Switch from fuzzer to formal when $r_{fuzz}(w) < r_{fml}$.
- Rate of fuzzer of the recent w tests: $r_{fuzz}(w) = \frac{total\ new\ cov.(w)}{total\ sim.time\ (w)}$.
- Rate for formal tool of $|C|$ points selected: $r_{fml} = \frac{num.of\ points\ in\ C}{total\ proof\ time}$.

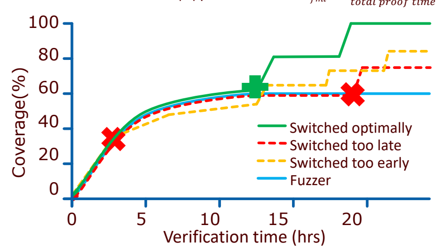


Figure 3. Switch illustration from fuzzer to formal tool.

Challenge 2: Selection of Uncovered Points

- ***BotTop*** selects modules with the deepest distance to the top module.
- ***MaxUncovd*** selects modules with the maximal number of uncovered points.
- ***ModDep*** selects modules with the highest logic dependence.

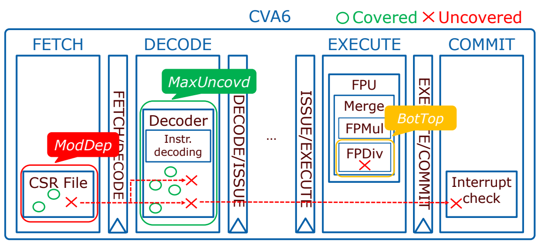
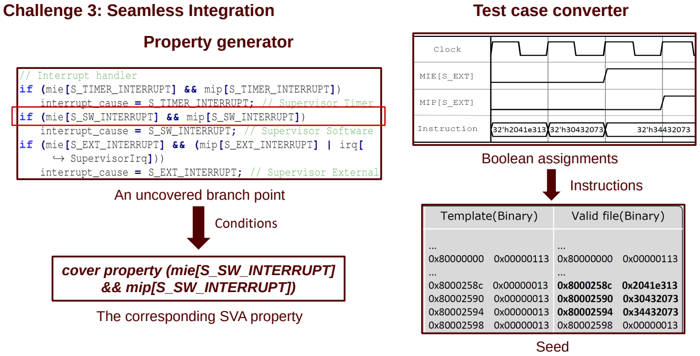


Figure 4. Different point selection strategies on CVA6[3].

Challenge 3: Seamless Integration



Experiment

Vulnerability detection

- Detected existing **11** vulnerabilities **3.06** × less time.
- Detected **three** new vulnerabilities.
- Resulted **two** CVEs: CVE-2022-33021, CVE-2022-33023.

Branch coverage achievement compared to:

- *TheHuzz*[2]: **41.24×** speedup, **6.84%** more total coverage.
- Random regression: **239.93×** speedup, **12.70%** more total coverage.

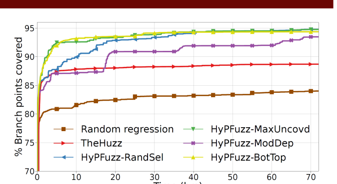


Figure 5. Branch points achieved by random regression, *TheHuzz*[2], and *HyPFuzz* on CVA6[3].

Reference

1. National Vulnerability Database. https://nvd.nist.gov/, 2023.
2. Kande, Rahul, et al. "TheHuzz: Instruction Fuzzing of Processors Using Golden-Reference Models for Finding Software-Exploitable Vulnerabilities." USENIX Security, 2022.
3. Zaruba, Florian, et al. "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology." VLSI, 2019.

Contact

- Chen Chen, Ph.D. student in Computer Engineering, Texas A&M University
- Email: chenc@tamu.edu Lab Website: https://seth.engr.tamu.edu/

Published at USENIX Security Symposium 2023