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× 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†.

REFERENCES


†https://www.usenix.org/conference/usenixsecurity23/presentation/chen-chen
HyPFuzz: Formal-Assisted Processor Fuzzing

Chen Chen1, Rahul Kande1, Nathan Nyugen1, Flemming Andersen1, Aakash Tyagi1, Ahmad-Reza Sadeghi1, and Jeyavijayan Rajendran1
1Texas A&M University, *Technische Universität Darmstadt

Introduction

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

Research Objective

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

Table 1. Comparison between various hardware verification techniques.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Automated</th>
<th>Scalable</th>
<th>Efficient</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Inspection</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Formal verification</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Random regression</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Hardware Fuzzing</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>HyPFuzz</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Methodology

HyPFuzz consists of three main components: Fuzzer, Intermediate, and Formal Tool. The Fuzzer includes a Mutation engine, Coverage, and Test case database. The Intermediate involves a Scheduler, Point selector, Property generator, and Test case converter. The Formal Tool includes SystemVerilog Assertion, cover property P1, cover property P2, Reachability, Boolean assignment, and coverage.

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

Figure 2. Framework of HyPFuzz.

Challenge 1: Scheduling of Fuzzer and Formal Tool

- Switch from fuzzer to formal when \( r_{\text{fuzz}}(w) < r_{\text{form}} \).
- Rate of fuzzer of the recent \( w \) tests: \( r_{\text{fuzz}}(w) \).
- Rate for formal tool of \( |C| \) points selected: \( r_{\text{form}} \).

Challenge 2: Selection of Uncovered Points

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

Challenge 3: Seamless Integration

Property generator

- \( \text{Cover property (\text{milliSW\_INTERRUPT})} \)
- \( \text{& (\text{milliSW\_INTERRUPT})} \)

The corresponding SVA property

Figure 3. Switch illustration from fuzzer to formal tool.

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

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

Experiment

Vulnerability detection

- Detected existing 11 vulnerabilities 3.06 \times less time.
- Detected three new vulnerabilities.

Branch coverage achievement compared to:

- TheHuzz: 41.24x speedup, 8.04% more total coverage.
- Random regression: 239.93x speedup, 12.70% more total coverage.

Reference


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