HyPFuzz: Formal-Assisted Processor Fuzzing

Chen Chen†, Rahul Kande†, Nathan Nguyen†, Flemming Andersen†, Aakash Tyagi†, Ahmad-Reza Sadeghi*, Jeyavijayan Rajendran†

†Texas A&M University, College Station, USA,
*Technische Universität Darmstadt, Germany
Massive Growth in Hardware Vulnerability

- Hardware vulnerabilities emerge at an alarming rate

  Total documented hardware vulnerabilities (CVEs) by year

  100

  92

- Hardware vulnerabilities are difficult to be patched
  - Pentium FDIV: $475 million, 1994
  - Meltdown and Spectre: biggest patch coordination
  - Xilinx Starbleed: vulnerability is in the silicon

Source: National Vulnerability Database NVD (08/2023)
## Existing Hardware (HW) Fuzzers

<table>
<thead>
<tr>
<th>Technique</th>
<th>Coverage Metric</th>
<th>Largest Design</th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFUZZ</td>
<td>Select signals of some MUXs</td>
<td>5-stage Sodor core</td>
<td>0</td>
</tr>
<tr>
<td>DifuzzRTL</td>
<td>Registers driving select signals of some MUXs</td>
<td>Boom</td>
<td>16</td>
</tr>
<tr>
<td>HyperFuzzing</td>
<td>Inserted properties</td>
<td>SHA crypto engine</td>
<td>0</td>
</tr>
<tr>
<td>Trippel et al.</td>
<td>SW FSM, line, edge, HW toggle, functional</td>
<td>KMAC</td>
<td>5</td>
</tr>
<tr>
<td>TheHuzz</td>
<td>HW FSM, line, edge, branch, toggle</td>
<td>Boom</td>
<td>11</td>
</tr>
</tbody>
</table>
Limitations of Existing HW Fuzzers

- Hard-to-reach design spaces due to specific conditions
- Example:

```c
// CVA6 Interrupt handler
if (mie[S_TIMER_INTERRUPT] && mip[S_TIMER_INTERRUPT])
    interrupt_cause = S_TIMER_INTERRUPT;  // Supervisor Timer
if (mie[S_SW_INTERRUPT] && mip[S_SW_INTERRUPT])
    interrupt_cause = S_SW_INTERRUPT;  // Supervisor Software
if (mie[S_EXT_INTERRUPT] && (mip[S_EXT_INTERRUPT] |
    irq[SupervisorIrq]))
    interrupt_cause = S_EXT_INTERRUPT;  // Supervisor External
```

After 72 hours, 200K tests, TheHuzz did not trigger it
Probability of Triggering the Branch

If (mie[S_SW_INTERRUPT] && mip[S_SW_INTERRUPT])
interrrupt_cause = S_SW_INTERRUPT; // Supervisor Software

\[
P(S\_SW\_INTERRUPT == 1) = \frac{1}{2}
\]

\[
P(\text{instr} == \text{CSRRS}) = \left(\frac{4}{1146}\right)^2
\]

\[
P(\text{CSR} == \text{mip}) = \frac{1}{229}
\]

Overall \( P(\text{Branch} == \text{True}) = 1.16 \times 10^{-10} \)
Hybrid Fuzzing: Fuzzing + Formal Verification
HyPFuzz: Framework

**Fuzzer**
- Mutation engine
- Test case database
- Seed:
  - add x1, x2, x3
  - sub x1, x2, x3
  - return

**Coverage**
- 01010
- 11100
- 00111
- 10001
- 01000
- 11111

**Intermediate**
1. Scheduler
2. Point selector
3. Property generator
4. Test case converter

**Formal Tool**
- SystemVerilog Assertion
  - cover property P1
  - cover property P2
  - ...

**Boolean assignment**
- Reachability
Challenge: Schedule of Fuzzer and Formal

When should HyPFuzz switch from Fuzzing to Formal?

Illustration

- Switched optimally
- Switched too late
- Switched too early
- Fuzzer

Coverage (%)

Verification time (hrs)
Scheduler

- Switch from fuzzer to formal tools when $r_{fuzz} < r_{fml}$

- $r_{fuzz}$: coverage increment rate of the recent $w$ tests

\[ r_{fuzz}(w) = \frac{\text{total new cov.}(w)}{\text{total sim. time}(w)} \]

- $r_{fml}$: moving average rate on hard-to-cover point set $C$

\[ r_{fml} = \frac{\text{num. of points in } C}{\text{total proof time}} \]
Evaluation

• **Benchmarks:** CVA6, BOOM, Rocket Core, OR1200, mor1kx
• **Coverage metric:** Branch

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

• **Coverage achievement compared to:**
  • *TheHuzz*: **41.24 ×, 6.84%**
  • Random regression: **239.93 ×, 12.70%**
Vulnerability Found

- **CVA6** returns unknown value (‘X’) when accessing some control and status register (CSRs)
- Example: Fuzzer alone: hard to generate CSR addresses
- Formal tool alone: hard to explore all CSRs

Example:

```c
CSRRC, GPR18, GPR0, CSR31:
GPR18=XXXXXXXXXXXXXXXXXXXXX
```

```c
Module 1

```c
case (CSR_address):
  CSR0: data = CSR_value;
  CSR1: data = CSR_value;
  ...
  CSR31: data = CSR_value;

Module 2

```c
reg [16][63:0] CSRs;
```
Compatibility of **HyPFuzz**

- **FSM**: state transitions in a DUT
- **Condition**: all combinations of values of signals in branch statements
Conclusion

- **HyPFuzz:**
  - Automatically integrates fuzzer and formal tool
  - Dynamically schedules use of fuzzer and formal tool
  - Outperforms existing processor fuzzers
  - Found new vulnerabilities that are difficult for fuzzer and formal tool to find
  - Is compatible to different coverage metrics

---

Thank you!

[chenc@tamu.edu](mailto:chenc@tamu.edu)

Lab Website: [https://seth.engr.tamu.edu/](https://seth.engr.tamu.edu/)

LinkedIn: [https://www.linkedin.com/in/chen-chen-127235170/](https://www.linkedin.com/in/chen-chen-127235170/)