PHMon: A Programmable Hardware Monitor and Its Security Use Cases

Leila Delshadtehrani, Sadullah Canakci, Boyou Zhou, Schuyler Eldridge, Ajay Joshi, and Manuel Egele

delshad@bu.edu

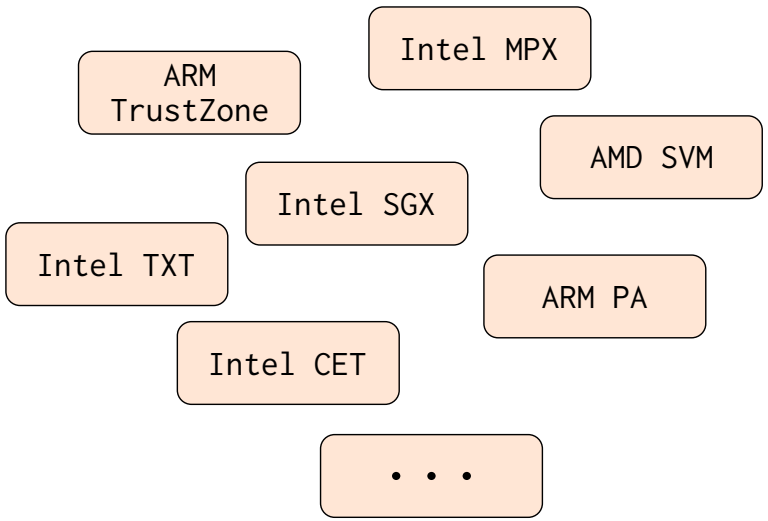
Boston University

August 12, 2020

Motivation

Current Trend

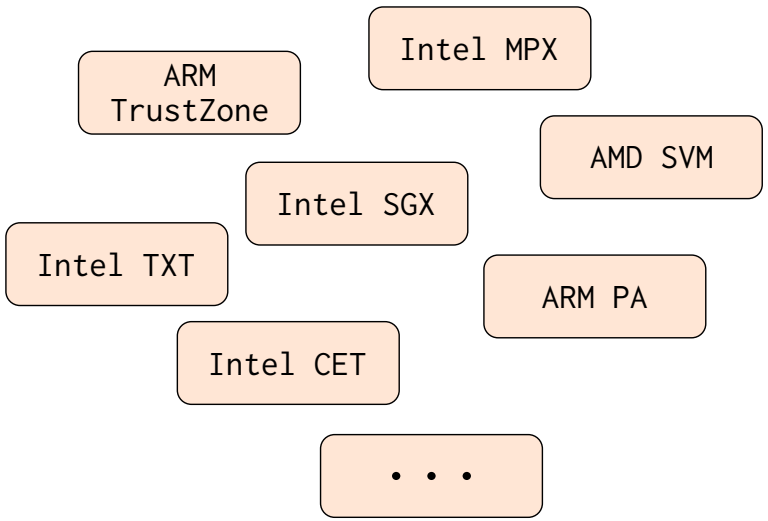
- Growing demand to enforce security policies in hardware

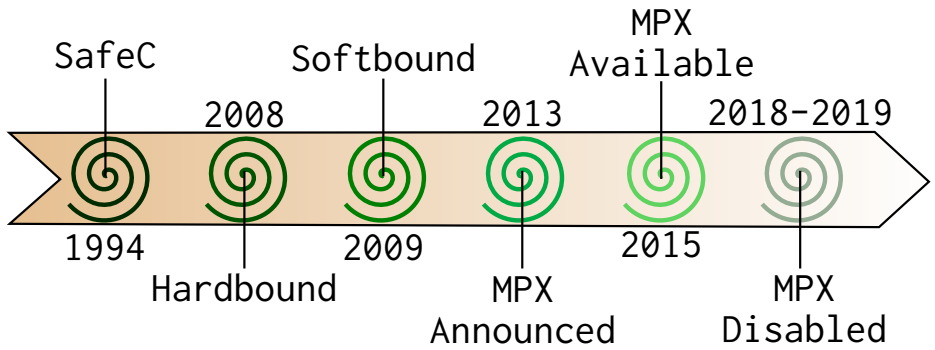


Motivation

Current Trend

- Growing demand to enforce security policies in hardware



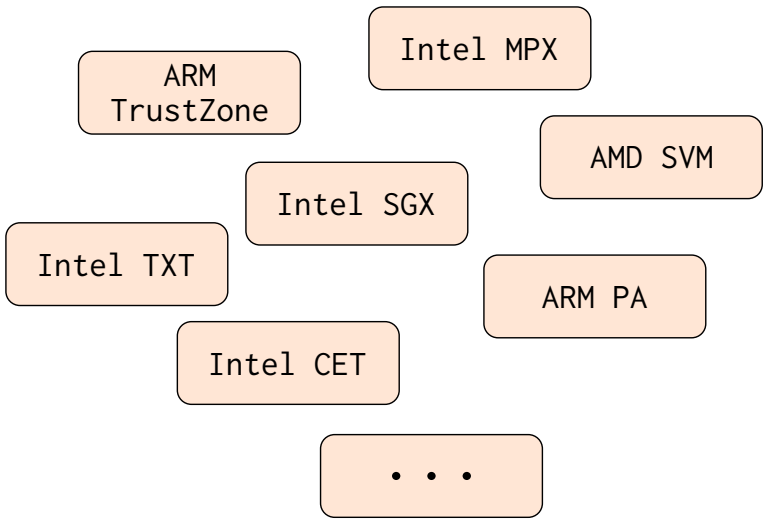


[AUSTIN, PLDI'94] [DEVIETTI, ASPLOS'08] [NAGARAKATTE, PLDI'09]

Motivation

Current Trend

- Growing demand to enforce security policies in hardware

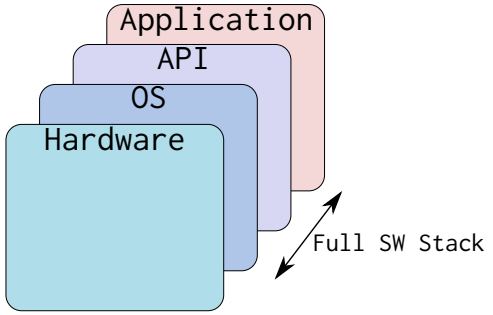


What if
we could have a flexible hardware implementation that could enhance and enforce a variety of security policies as security threats evolve?!

Our Proposal - PHMon

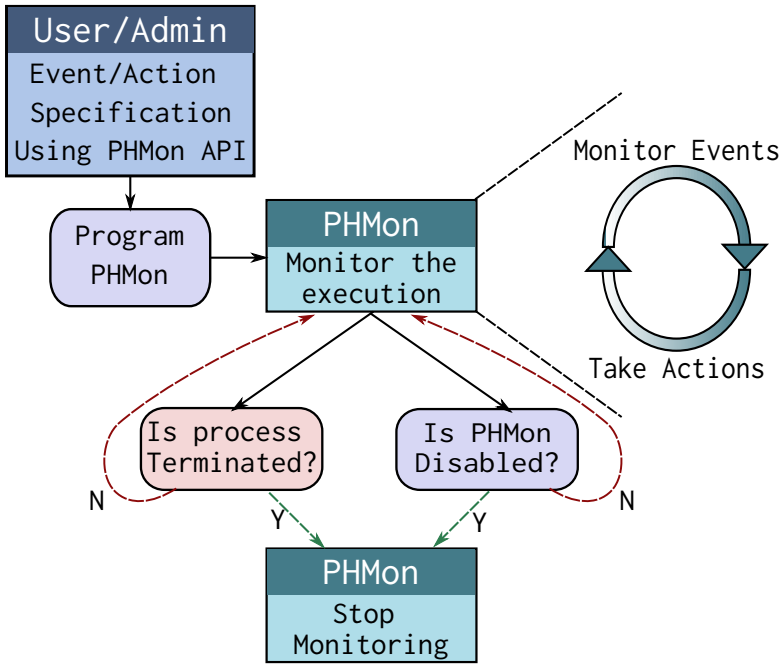
PHMon

- A hardware monitor and the full software stack around it
  - A programmable hardware monitor interfaced with a RISC-V Rocket processor on an FPGA
  - OS support
  - Software API
  - Security use cases



How Does It Work?

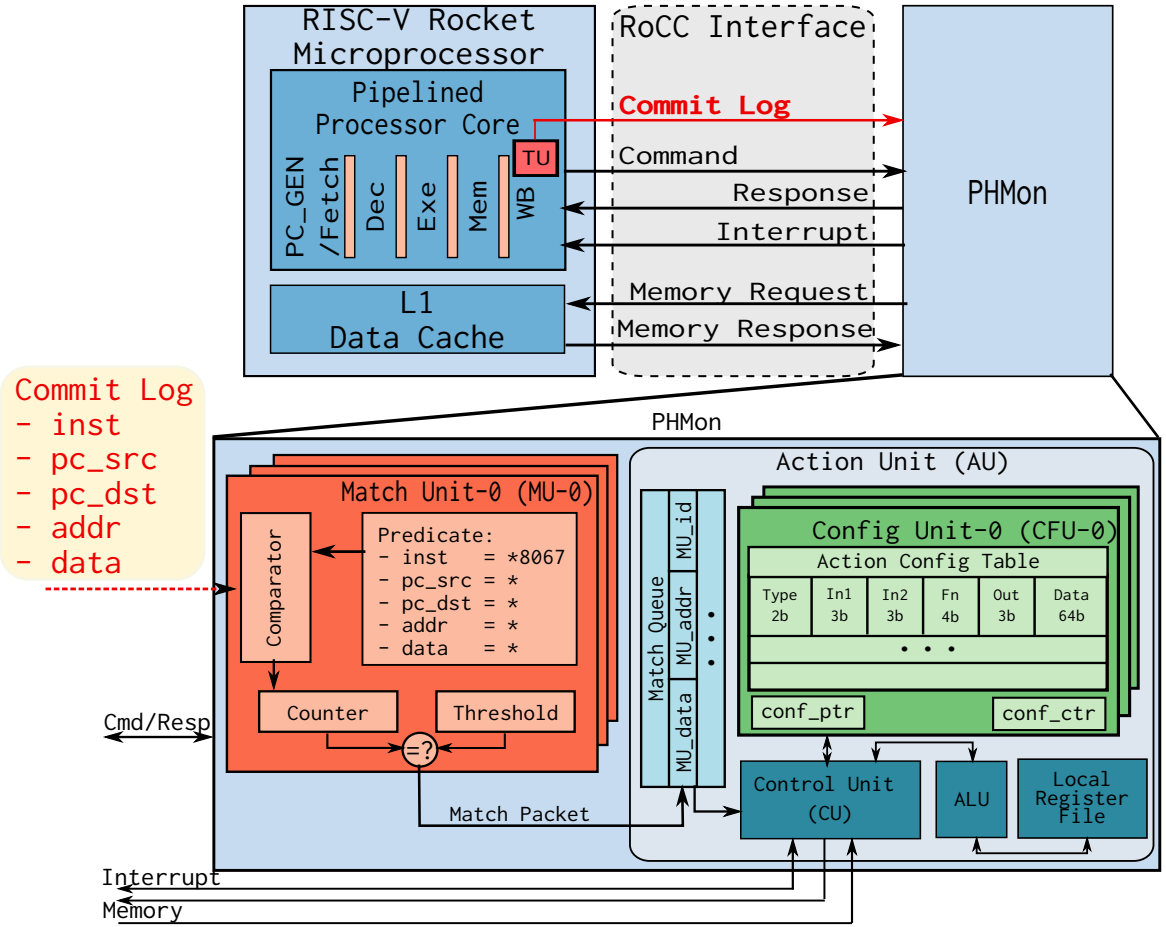
- A user/admin configures the hardware monitor
- The hardware monitor collects the runtime execution information
  - Checks for the specified events, e.g., detects branch instructions
  - Performs follow-up actions, e.g., an ALU operation



Hardware Overview

HW Functionality

- Collect the instruction trace
- Find matches with programmed events
- Take follow-up actions



Software Overview

Software Interface

- A list of functions that use RISC-V's standard ISA extension
  - Configure PHMon
  - Communicate with PHMon

Reset MU-0 and configure the match pattern

```
phmon_reset_val(0);
phmon_pattern(0, &mask_inst0)
```

Compare pc_dst and pc_src, and trigger an interrupt

```
action_mu0.op_type = e_OP_ALU; //ALU operation
action_mu0.in1 = e_IN_DATA_RESP; //MU_resp
action_mu0.in2 = e_IN_LOC3; //Local3
action_mu0.fn = e_ALU_SEQ; //Set Equal
action_mu0.out = e_OUT_INTR; //Interrupt reg
phmon_action_config(0, &action_mu0);
```

OS Support

- Per process OS support
  - Maintain PHMon information during context switches
- Interrupt handling OS support
  - Delegate interrupt to OS
  - Terminate the violating process

Implementation and Evaluation Framework

Implementation

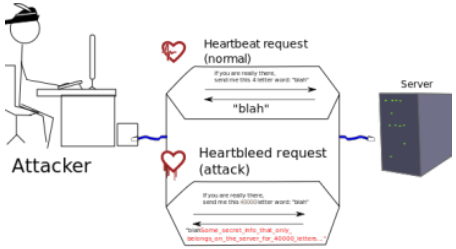
- PHMon as a RoCC, written in Chisel HDL
  - Interfaced with the in-order RISC-V Rocket core
- Linux kernel v4.15
- RISC-V gnu toolchain for cross-compilation

Evaluation

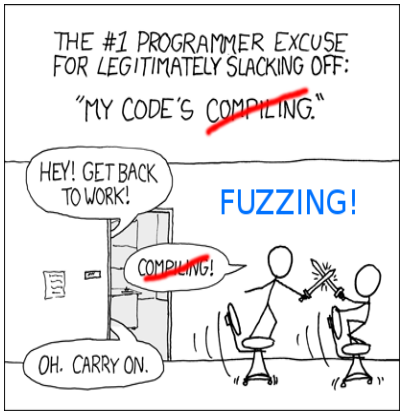
- Prototyped on Xilinx Zynq Zedboard
  - Rocket core + PHMon
- Open-sourced at https://github.com/bu-icsg/PHMon

Use Cases

Shadow Stack



Preventing Information Leakage



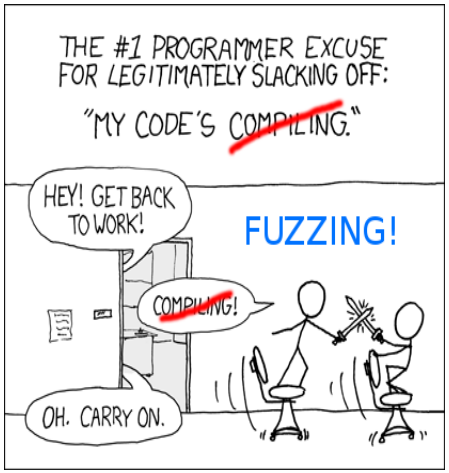
Hardware Accelerated Fuzzing

Watchpoints and Accelerated Debugger

https://security.googleblog.com/2019/10/protecting-against-code-reuse-in-linux30.html,
https://www.darkreading.com/attacks-breaches/heartbleed-attack-targeted-enterprise-vpn-/d/d-id/1204592,
https://medium.com/@dieswaytoofast/fuzzing-and-deep-learning-5aae84c20303,
https://hackernoon.com/professional-debugging-in-rails-1yr2bnz

Use Cases

Shadow Stack



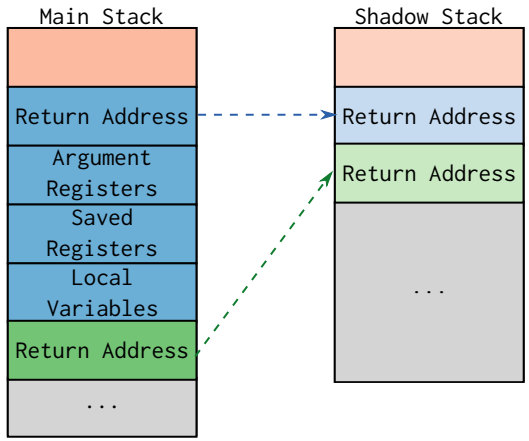
Hardware Accelerated Fuzzing

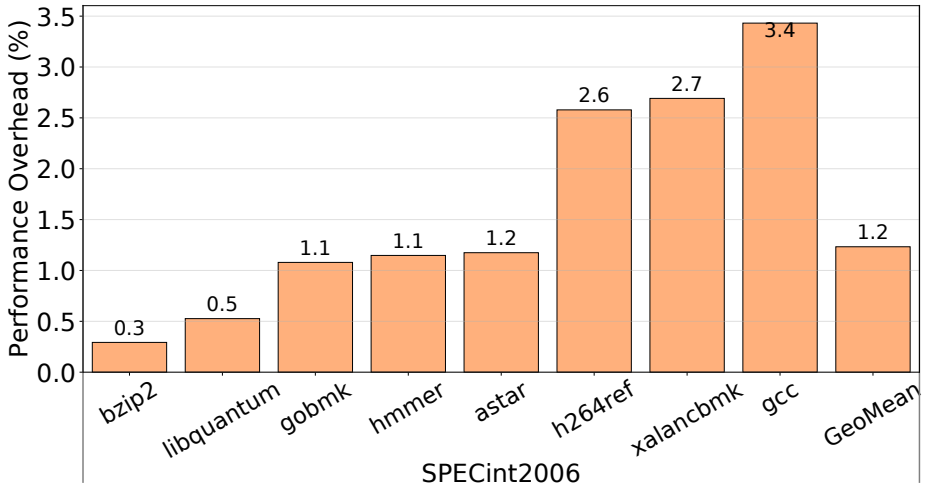
https://security.googleblog.com/2019/10/protecting-against-code-reuse-in-linux30.html,
https://medium.com/@dieswaytoofast/fuzzing-and-deep-learning-5aae84c20303,

Use Cases: Shadow Stack

PHMon-based Shadow Stack

- Simple and flexible
  - Two MUs
  - Shared memory space
    - Allocated by OS as a user-space memory
- Secure
- Efficient
  - For SPECint2000, SPECint2006, and MiBench benchmarks, on average, 0.9% performance overhead





Use Cases: Hardware Accelerated Fuzzing

American Fuzzy Lop (AFL)
[Zalewski, 2013]

- A state-of-the-art fuzzer
- Two main units
  - The fuzzing logic
  - The instrumentation suite
    - Compiler-based
    - **QEMU-based**

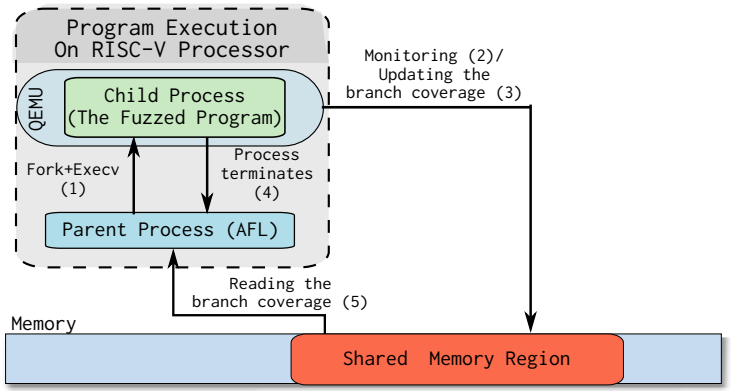
https://rabbitbreeders.us/american-fuzzy-lop-rabbits/

Use Cases: Hardware Accelerated Fuzzing

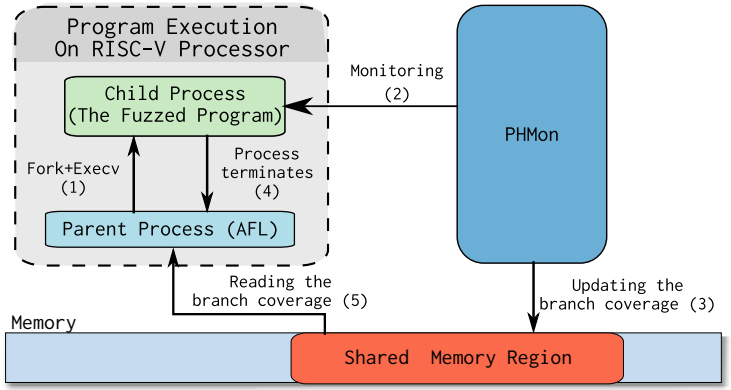
American Fuzzy Lop (AFL) [Zalewski, 2013]

- A state-of-the-art fuzzer
- Two main units
  - The fuzzing logic
  - The instrumentation suite
    - Compiler-based
    - **QEMU-based**

https://rabbitbreeders.us/american-fuzzy-lop-rabbits/

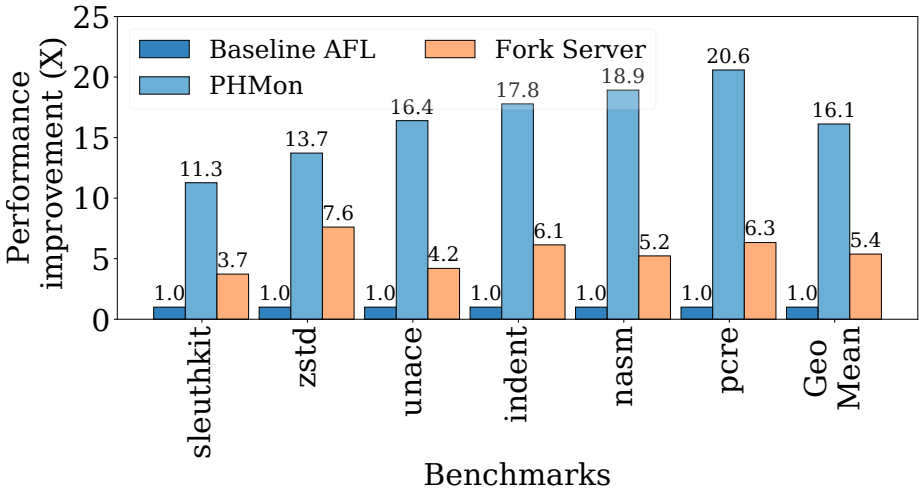


QEMU-based AFL

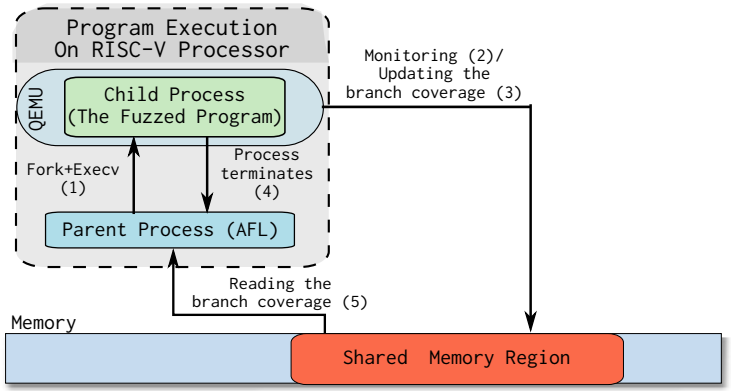


PHMon-based AFL

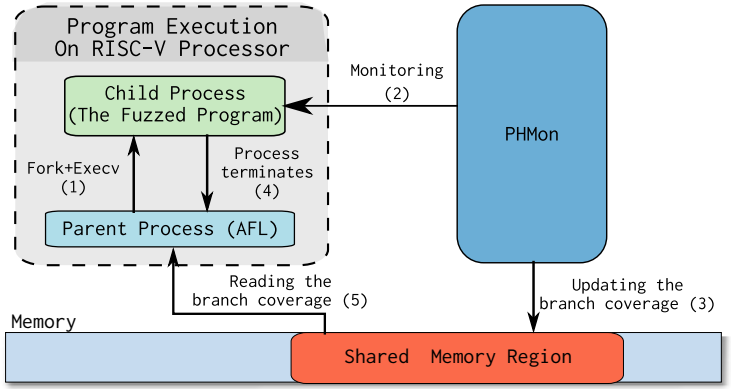
Use Cases: Hardware Accelerated Fuzzing



- PHMon improves AFL's performance by 16× over the baseline
- Power overhead: 5%
- Area overhead: 13.5%



QEMU-based AFL



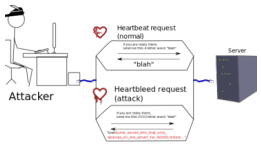
PHMon-based AFL

Conclusion

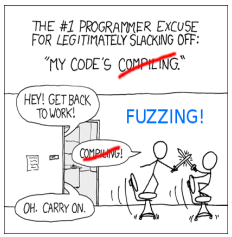
Ref: https://www.linux-apps.com/p/1082429/

A hardware monitor with full software stack

Shadow Stack



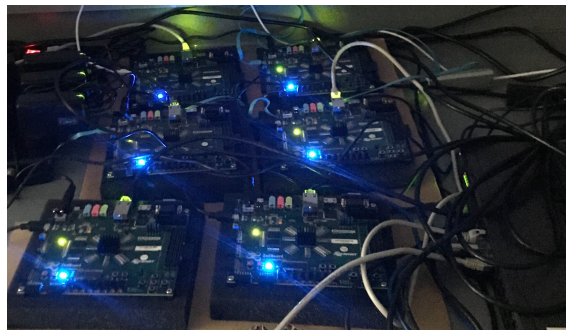
Preventing Information Leakage



Hardware Accelerated Fuzzing

Watchpoints and Accelerated Debugger

Versatile and easily adopted



FPGA prototype

ARTIFACT EVALUATED usenix ASSOCIATION PASSED

https://www.usenix.org/system/files/sec20spring_delshadtehrani_prepub.pdf

https://github.com/bu-icsg/PHMon

Thanks! You can reach me at delshad@bu.edu for follow-up questions.

More information