CheckMate: Automated Synthesis of Hardware Exploits and Security Litmus Tests

Naorin Hossain  
ctrippel@princeton.edu  
Margaret Martonosi  
mmr@princeton.edu  
Princeton University

- Widely implemented microarchitectural features such as speculative execution have been found to be exploitable using well-known side-channel attacks resulting in new exploits such as Spectre and Meltdown.
- Our automated approach uses microarchitectural happens-before analysis to determine whether a given microarchitectural design is vulnerable to a given security exploit class.
- If a vulnerability is found, our tool outputs synthesized code for a possible attack.

2018: The Year of the Hardware Security Exploit

Starting with Meltdown and Spectre, a recent wave of exploits leverage the effects of speculative execution on non-architectural state to make sensitive information available to software via some well-known side-channel attack. What is novel and surprising about these attacks is their ability to create practical working exploits out of a variety of widely-implemented x86 features. We present CheckMate, an approach & automated tool for determining x86 vulnerabilities for formalized security exploit classes, and for synthesizing proof-of-concept exploit code where applicable.

Augmenting µbGraphs for Security Verification

Exploit Patterns
- Add µb patterns to our modeling framework (i.e., µb sub-graphs)
- Exploit pattern + µarch execution patterns indicative of some exploit class
- Abstract yet still expressive

Exploit Program Execution
- µb graph that features an exploit pattern of interest

Security Litmus Tests
- Compact program that can induce exploit execution
- Abstracted for efficient analysis; path to full exploit clear

Relational Model Finding (RMF)
For Security Litmus Test Synthesis
A relational model (e.g., a µb graph) is a set of constrains on an abstract system of atoms and N-dimensional relations.
- Constraints: x[i] + µarch patterns
- Abstract system: µb graph nodes (atoms) + µb graph edges (2D relations)

Case Study 1: Evaluating Susceptibility of a Speculative OoO Processor to Cache Side-Channel Attacks

Given this exploit pattern, CheckMate will generate programs representing all of the ways in which the input µarch (+ related OS support) could induce a mis in the probe on the access.

Case Study 2: Evaluating a SandyBridge Processor Model

- Expand CheckMate’s capabilities and techniques with a more modern, complex microarchitectural model, resembling that of Intel Sandy Bridge.
- SandyBridge model handles both virtual and physical addressing and supports microarchitectural events leading from hardware-OS interactions.

Results: Automatically Generated Exploits

Testing on Real Hardware

We extended the SpectrePrime security litmus test and demonstrated SpectrePrime on a MacBook with a 2.4 GHz Intel Core i7 Processor running MacOS Sierra, Version 10.12.6.

SandyBridge Program Execution Model

sb Litmus Test

Highlighted instructions are “ghost” instructions. In this litmus test, they are page table walks.

Ghost instructions are in the process of being synthesized with our models. We will soon be interleaving them with user level instructions. They will include page table walks, dirty bit status updates, INVPG calls, etc.

In order to use CheckMate with this model to find security exploits, our ongoing work includes adding ghost instructions to our security litmus tests.

Ongoing Work

- Extended this litmus test generation (for hardware-OS interactions)
- Extend CheckMate’s capabilities in the following ways: 
  - Branch Target Buffer Security Enclaves (Intel SGX)
  - Floating Point Registers
- Next steps include extending our methodology with performance modeling capabilities (edges labeled with thermal or timing info)
- Automatically synthesize new exploit patterns

Conclusions

- Demonstrated value of SM and RM techniques for hardware-aware analysis
- Interactive runtime in minutes or hours: dozens of unique exploits in a lunch break
- Early-stage verification: synthesize real-world exploits on abstract representations of hardware; abstract exploit representations can synthesize a variety of exploits