Spectrum: Classifying, Replicating and Mitigating Spectre Attacks on a Speculating RISC-V Microarchitecture

Abraham Gonzalez, Ben Korpan, Ed Younis, Jerry Zhao

Berkeley UNIVERSITY OF CALIFORNIA

ADEPT

Background

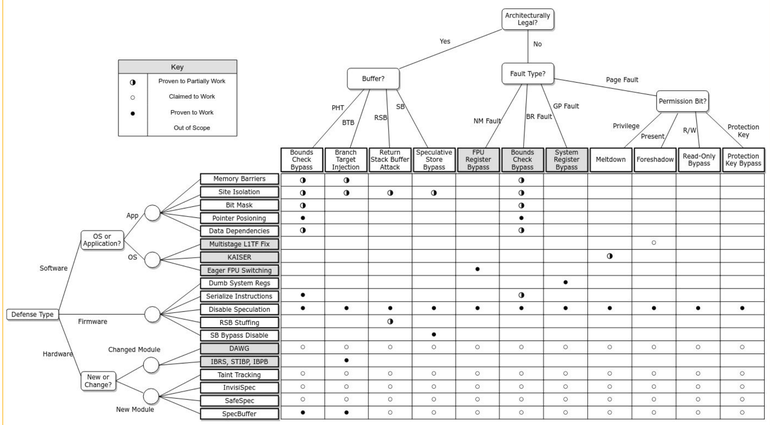
- Spectre and Meltdown
  - Affect modern out-of-order processor designs
  - Abuse speculation to retrieve secret data
  - More attacks being discovered
- Berkeley Out-of-Order Machine (BOOM)
  - Open source RV64G RISC-V core written in Chisel
  - Good platform for microarchitecture research
- BOOM lacks exposure to Spectre style attacks
- Standard taxonomy has not been adopted

Objectives

- Create taxonomy for researchers to understand attacks and defenses
- Replicate subset of attacks
  - Spectre Variants 1 and 2
- Create speculative buffer
  - Prevent misspeculated refills in L1 Data Cache (L1D$)

Taxonomy

- Taxonomy classifies both attacks and defenses on a single core
- Threat model only considers an attacker who can read speculative data from the cache
- Attacks are split into architecturally legal and illegal actions
  - Illegal actions raise a fault and are further split on the type
  - Plenty of attacks raise a page fault but differ in permission bit abused, so there is a last split for this
- Defenses are organized on implementation location.
  - Hardware is categorized into new or changed modules
  - Firmware is separate because implementation cost is drastically less
  - Software is split into OS and application due to the scope of the implementation



Taxonomy of Attacks and Defenses

Speculative Attack Replication

- Bounds Check Bypass and Branch Target Injection
  - Trained Branch Target Buffer and GShare Predictor
  - Small speculation window
    - No deep memory hierarchy resulted in fast load misses
    - Bypassed using multiple dependent long operations (*fdiv*'s) to expand the attack window
  - Handmade *clflush* since RV64G does not have one

Parameter	Value
Fetch Width	2
Decode Width	2
Issue Width	4
PRF Size	100
ROB Size	100
L1 Sets	64
L1 Ways	8
L1 Linesize	64 bytes
BTB Sets	512
BTB Banks	2
BTB Ways	4
GShare History Bits	23
GShare Counter Table Entries	4096

BOOM Parameters for Replicated Attacks

Parameter	Value
Cache Hit Threshold	50 cycles
Amount of runs on same byte	10 rounds
Training rounds for BPU	6 training rounds
Cache flush hits on same set	4 * L1_WAYS
GShare Counter Table Entries	4096

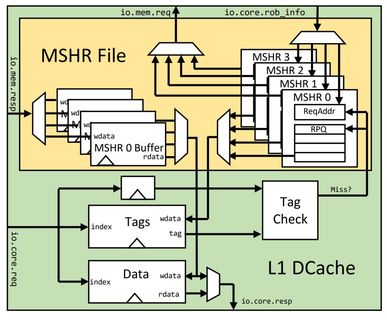
Attack Run Parameters

```
want(B) =?= 1.(B) 2.(<9f>)
want(a) =?= 1.(a) 2.(^B)
want(b) =?= 1.(b) 2.(^A)
want(y) =?= 1.(y) 2.()
want(B) =?= 1.(B) 2.( )
want(o) =?= 1.(o) 2.(^H)
want(o) =?= 1.(o) 2.(4)
want(m) =?= 1.(m) 2.(k)
want(e) =?= 1.(e) 2.(^C)
want(r) =?= 1.(r) 2.(^A)
want(T) =?= 1.(T) 2.(^C)
want(e) =?= 1.(e) 2.(^D)
want(s) =?= 1.(s) 2.(^D)
want(t) =?= 1.(t) 2.(^A)
```

Output from Spectre V1

SpecBuf: Speculative Buffer

- Extends existing mechanisms in BOOM's cache
  - BOOM uses the non-blocking RocketChip L1D$
  - SpecBuf modifies Miss Status Holding Registers (MSHRs)
    - MSHRs handle misses in a non-blocking cache
- Protects cache state from speculation while preserving performance



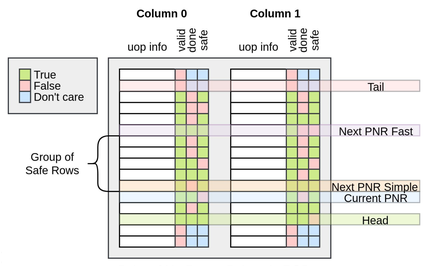
Modified MSHR File SpecBuf Implementation

Modifications:

- Added a cache line buffer to each MSHR
- Buffered line not written to cache until corresponding instruction is speculatively safe
- Allow load forwarding from MSHR buffer to execution pipeline prior to cache refill

Added a Point-of-No-Return (PNR) pointer to the Re-Order Buffer (ROB):

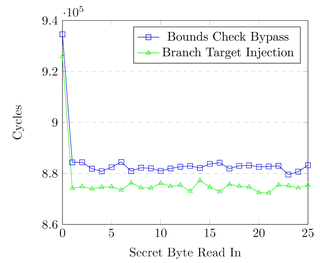
- Helps to commit speculative cache refills faster
- Points at oldest speculatively unsafe instruction
- Two versions: simple and fast



Point-of-No-Return Pointer in ROB

Results

- Attacks and SpecBuf tested on FireSim
- Benchmark tests for SpecBuf
  - Stressed SpecBuf MSHR blocking and eviction limitations
  - Able to run Dhrystone
  - Unable to boot Linux - No SPEC2017 and CoreMark results
- Preliminary 45nm synthesis in HAMMER
  - 2.5% increase in area
  - 0.36% decrease in clock frequency



Speed of Replicated Attacks Per Byte

Attack	Cycles for Secret Byte	Bytes per Second 100 MHz	Bytes per Second 3.2 GHz
Bounds Check Bypass	884485	113 B/s	3618 B/s
Branch Target Injection	876602	114 B/s	3650 B/s

Speed of Replicated Attacks

Benchmark	Version of BOOM Normal	SpecBuf	% Diff.
Non-spec. LD misses to same sets	540 cycles	640 cycles	-19%
Non-spec. LD misses to diff. sets	264 cycles	297 cycles	-11%
MSHR evicted spec. LD miss	48 cycles	67 cycles	-40%
Dhrystone	2176 Dhrystones/s	2216 Dhrystones/s	+2%

Microbenchmark Results

Future Work

- Replicated Attacks
  - Test on Linux
  - Improve and test RSB attacks
- Run more evaluations
  - SPEC2017
  - CoreMark Pro
  - Memory Centric Workloads
- SpecBuf improvements
  - Security enhancements
  - Reduce physical impact
- BOOM improvements
  - Branch Predictor bugfixes
  - Multiported L1 cache
  - L2 integration

Conclusion

- Created taxonomy to more easily understand Spectre-style attacks
- Showed BOOM's vulnerability to Spectre-style attacks
  - Implemented Bounds Check Bypass and Branch Target Injection attacks
- Created initial implementation of SpecBuf
  - Prevents cache side channel attacks
- Demonstrated BOOM's capability for doing HW security research using RISC-V
  - Less than 100 person-hours to implement and debug using Chisel

Acknowledgments

Thanks to Chris Celio for his thoughts on hardware mitigations for Spectre and for building BOOM. Additionally, we would like to thank the ADEPT lab and FireSim/HAMMER teams for providing helpful resources.