Transient Execution Emulator

Meltdown and Spectre Behind the Scenes

Felix Betke, Lennart Hein, Melina Hoffmann, Jan-Niklas Sohn

2022-04-01

Rheinische Friedrich-Wilhelms-Universität Bonn

Structure

- Topic
- Background
- Our task
- Our approach
- Implementation
- Demo
- Conclusion

Topic

- Lab builds on SCA lecture
- Meltdown and Spectre mostly patched
- Difficult to experiment with
  - Personal computer often times not usable
- Goal: Vulnerable CPU Emulator that runs on many systems
  - Should offer a gdb-like interface

Background

CPU

- Frontend:
  - Fetches/Decodes instructions, maintains queue
  - Branch prediction
- Execution Engine:
  - Multiple sets of execution units
- Memory Subsystem:
  - Handles memory operations
  - Maintains L1 cache
  - Ensures data is loaded from other caches/memory

Out-of-order execution

- Independent instruction streams
- Tomasulo algorithm:
    - Reservation stations
    - Common Data Bus
- Rollbacks

Speculative execution

- Predict results of branch instructions
- Prevent stalls
- BPU maintains counters
- Rollbacks

Meltdown

- Abuses out-of-order execution
- Meltdown-US-L1:
  - Define oracle array
  - Perform illegal read to steal secret
  - Embded secret-dependent oracle entry into cache
  - Await rollback and measure oracle access times
- Small time window

Spectre

- Abuses speculative execution
- Different variations. Here: prediction of branch instrs.
- Spectre v1: Deliberately train BPU used by victim process
- Make victim leak secret into cache
- Direct consequence of speculative execution

Mitigations: Meltdown

- Disable out-of-order execution
- Intel's microcode mitigation
  - Microprograms
- OS mitigations

Mitigations: Spectre

- Disable speculative execution:
    - Completely disable
    - fence instructions
- Flush entire cache after rollback

Our task

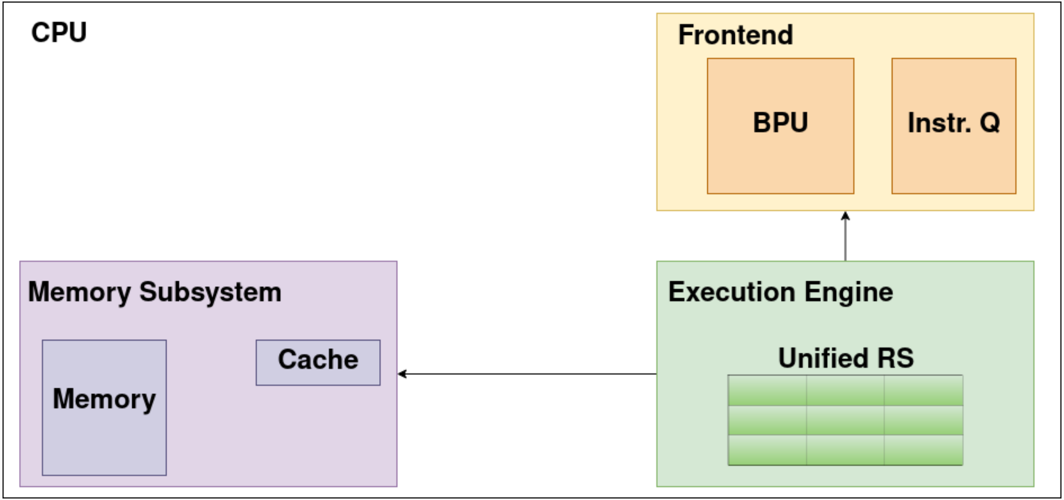
- Develop graphical CPU emulator vulnerable to:
  - Our version of Meltdown-US-L1
  - Spectre v1
- Must support single step, out-of-order, and speculative execution
- Implement Intel's microcode mitigation
- Other mitigations via microprograms
- Target audience: SCA students
  - Or anyone with basic knowledge of TE attacks

Our approach

How we started

- Must-haves, nice-to-haves, future work
- At time of Meltdown/Spectre publication: Skylake
- Filter components needed for our Meltdown/Spectre versions
- Build simplified CPU

Our version


CPU
Frontend
BPU
Instr. Q
Memory Subsystem
Memory
Cache
Execution Engine
Unified RS

References

- ja