Damian Barabonkov

MI6 Secure Enclaves

... in a Speculative Out-of-Order Processor

Overview

- Goals of MI6
- Big Ideas
- Paper Feedback
- Motivation of MI6
- Threat Model
- Implementation
- Performance Analysis
- Discussion Questions

Goals of MI6

- Provide a processor specification capable of speculative and out-of-order execution

  AND

- Protect process isolation against microarchitectural side channels

Big Ideas

- Secure Enclave
  - With protection domains
- *Trusted* Security Monitor
  - Mediates enclave entry/exit
  - Verifies resource allocation
- Hardware Modifications
  - LLC set-partitioning
  - Separate memory pipelines per core to avoid data leak from resource contention
  - Speculation guard for security monitor
  - *purge* hardware instruction

Paper Feedback (Positive)

- Explains how cache queues (MSHR) work for uninformed reader
  - Upgrade Queue (UQ)
  - Downgrade Queue (DQ)
  - Downgrade-L1 Logic
- Provides proof-of-concept implementation on FPGA

What do you think?

Paper Feedback (Needs Improvement)

- Was confused whether MI6 enclave was separate piece of secure hardware such as SGX and Apple enclave
- Definition of “protection domain” is relatively short for how important a concept it is to the paper

What do you think?

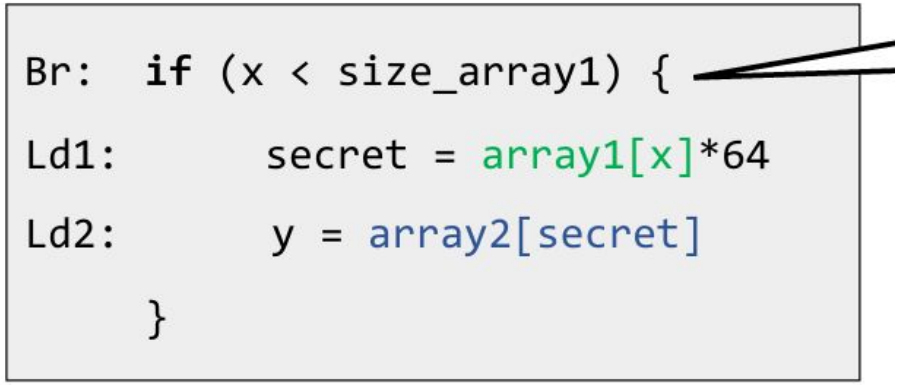
Motivation of MI6

- Attacks such as Spectre and Meltdown use microarchitectural side channels to leak data
- Breaking process isolation posses massive security threats
- Eliminating microarchitectural side channels is large value add
  - Minimal/acceptable performance loss
  - Software and hardware utilized to provide targeted solution

Motivation of MI6 (Example Side Channel)

Attacker would:

- prepare branch misprediction
- access a secret value in *array1*
- transmit the secret via a cache side channel through *array2*



Credit: Mengjia Lec 6

Motivation of MI6 (Current Status of Tech)

- No production processor has any strong defences against microarchitectural side-channel data leaks

- Precursor research *Sanctum* presents security monitor
  - Memory/cache hierarchy unrealistic
  - Does not support complex processor microarchitecture

MI6
Solves *all* of these shortcomings

Threat Model

Attacker reach:

- Compromise any operating system and hypervisor present
- Launch malicious enclaves
- Has complete knowledge of microarchitecture design

Attacker can:

- Analyse passively observed data (page fault addresses)
- Launch active attacks (memory probing)
- Exploit speculative state (branch prediction)

Not in Threat Model

- Attacker does not have physical present to hardware
- Attacks that rely on sensor data are considered physical
- No Denial-of-Service protection
- No protection against hardware bugs

Poll Question

What breaks timing independence when using network card (NIC)? (Select all that apply)

1) Processor LLC Cache
2) Hardware mapped memory
3) NIC Queue Latency
4) Security Monitor verifying NIC resources
5) NIC Queue Size

Implementation

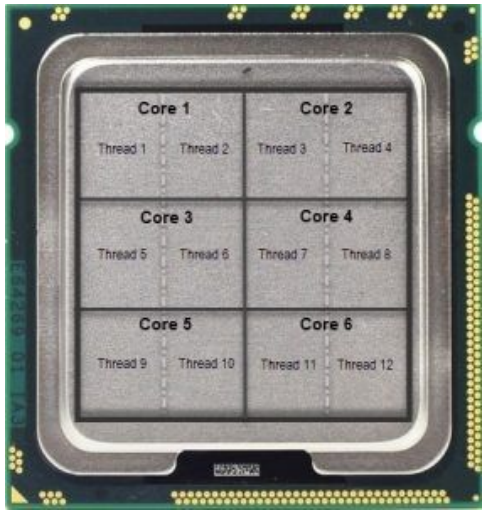
Note: Enclave is not a separate, physical piece of hardware on processor. Simply a terminology for a process isolated from rest.

Main Implementation Points:

- LLC partitioned cache sets per core
- Security monitor ensures validity and isolation of hardware resources
- Dedicated cache pipeline queues per core (MSHR partitioning)
- DRAM controller constant latency
- *purge* instruction to clear enclave state before context switch

Implementation (LLC set partitioning)

- Each core can run a single enclave at a time
- Each enclave owns predetermined sets of LLC



Prevents cache line contention between enclaves

Eliminates cache timing side-channels

Implementation (Security Monitor)

- Trusted software
  - Can use hardware to authenticate its own integrity
- Resides in highest level of security permission
  - Interposes scheduling and physical resource allocation decisions made by (possibly untrusted) OS
  - Asserts that one enclave's resources do not overlap with another's
  - Scrubs resources before they are available for reallocation
  - Facilitates messaging between enclaves
- Speculative execution disabled to prevent hijacking and misuse

Implementation (Security Monitor cont.)

When is it invoked?

- Upon enclave creating/destruction
- When an enclave is scheduled in/out
- When memory is granted to an enclave

Also

- When an enclave performs an OS system call
- When an enclave needs to communicate with another

Implementation (Security Monitor cont.)

How does the Security Monitor handle communication?

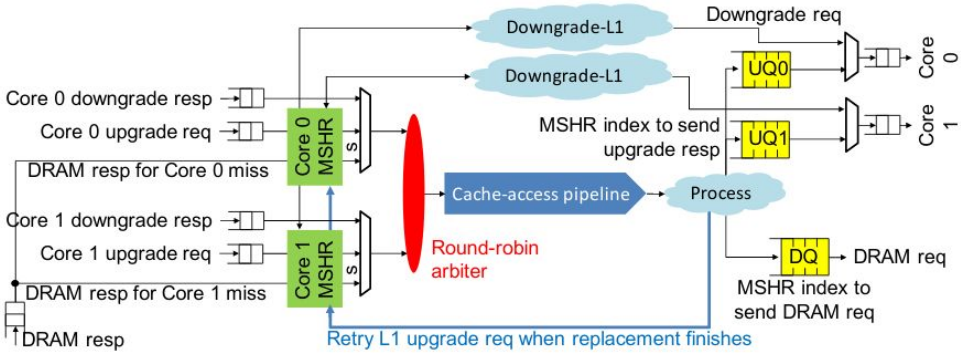
- Implements primitive to share 64B messages between enclaves
- Implements privileged memcopy between buffers of equal size of two enclaves
- Responds to “read/write” of OS buffer using memcopy

Comm timings are padded to a constant latency (zero leakage) or a fixed set of latencies (limited leakage).

Implementation (MSHR partitioning)

- Each core will have dedicated MSHR and upgrade queue for memory requests to cache
- Downgrade queue takes 1 cycle per MSHR index, therefore never blocks

Prevents contention for cache accessing among enclaves



Credit: MI6 paper, pg 48

Implementation (DRAM constant latency)

- Memory accesses to DRAM are aggregated from all cores into DRAM controller
- Controller usually reorders accesses to group ones that target same memory banks

Simple Solution:

- Each access to DRAM should take constant time, regardless of grouping
- Eliminates controller timing-based side-channels

Implementation (*purge* instruction)

Problem

- Upon context switch, swapping an enclave out of a core may leave residual side-channel state
  - Branch prediction trained
  - Cache buffer queues may be non-empty

Solution

- *purge* instruction clears all side-channel state before enclave leaves
- L1 and TLB caches flushed
- Note: L2 does not need to be flushed since enclaves do not share cache sets

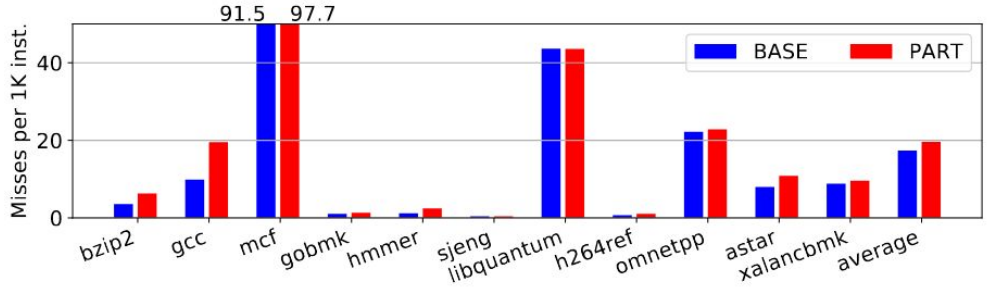
Performance Analysis

- Implemented on FPGA emulator (AWS F1 FPGAs)
- Tiered performance analysis
- 16.4% average slowdown for protected programs

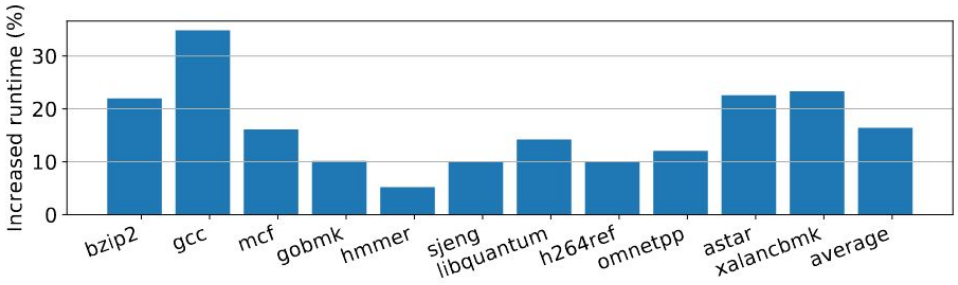
Measure performance hits for every MI6 overhead variable

- BASE — baseline
- FLUSH — flushes per-core microarchitectural states at every context switch
- PART — set-partition LCC of BASE processor
- MISS — changes in organization of LLC MSHRs of BASE processor
- ARB — increase latency of LLC pipeline for BASE processor to simulate round-robin arbiter
- NONSPEC — executes memory instructions non-speculatively on BASE processor
- F+P+M+A — FLUSH + PART + MISS + ARB

Performance Analysis (cont.)



LLC misses in
BASE and PART



Performance
Overhead of MI6

Discussion Questions

1. This method for securing side-channels is patchwork approach, targeting specific weak areas of architecture. Is this approach fool proof and enough?
2. Can contention in the security monitor itself due to simultaneous requests from multiple different enclaves leak information?
3. When is it simply cheaper/easier to run secure software on a dedicated CPU vs. sharing a CPU and using secure enclaves?