Hardware-based Cache Auditing to Aid Cache Timing Channel Detection

Fan Yao, Hongyu Fang, Guru Venkataramani and Miloš Doroslovački
The George Washington University | Washington, DC

1. Background

- Timing channel attacks illicitly leak sensitive secrets to malicious parties.
  - In covert channels, Trojan (sender) and Spy (receiver) collude to subvert system security policy.
  - In side channels, a benign victim unknowingly leaks sensitive data to a malicious spy.

2. Cache Timing Channel Attacks

- Cache Timing Channel Attacks
  - Cache timing channels are extremely stealthy.
  - Very challenging to detect due to various communication protocols involved (parallel, serial, single group/multiple groups etc.).

3. Existing Detections and Motivation

- Software-based detection mechanism\(^1\)
  - Based on high level statistics from Performance Counters (LLC misses)
  - No need for architecture supports
  - May be subject to high false negatives/positives

- Hardware-based detection mechanism\(^2\)
  - Finer-grained statistics and higher effectiveness
  - Do not provide high coverage and/or incur non-trivial overheads

- A solution that captures the fundamental characteristic of cache timing channels (high coverage) with minimal design cost.

4. Hardware-based Cache Auditing

- Capture the building block for communicating secrets on caches
  - Cache conflicts occur at set level. To ensure conflicts, Trojan and/or Spy have to fill all the ways in a targeted set.

- Hardware-based Cache Auditing
  - Capture the atomic communicating semantic: Spy access → Trojan
  - Track a single event: when the Trojan’s full way occupancy is destroyed in the spy (or vice versa), WOE
  - No reliance on communication protocol modeling, fundamentally hard to eliminate the events.

5. System Design & Exp. Setup

- Cache Auditing Design Overview
  - Bookkeeping ownership of cache lines (a few extra bits, may already be maintained by modern processors)
  - Recording WOE occurrence for each pair (e.g., a pair of cores)
  - Interfacing with Operating System. Provide statistics for further diagnosis

- Experimental Setup
  - All experiments run with cycle-accurate simulation on Gem5
  - Simulate a 4-core OoO processor with 32KB private L1 Caches and one shared 512KB L2 Cache
  - Full system mode with Linux kernel version 2.6.32

6. Cache Auditing Traces – Way Occupancy History

- All figures show the breakdown of way occupancy on a single hot set for the four applications running on distinct cores.

7. WOE Statistics

- Histograms of WOE trigger ratio for benign workloads and the attack. (Trigger ratio = WOE_counts / total_evictions for each pair)

8. Conclusion

- We observed a way occupancy event that is fundamentally related to cache channel attacks.
- We proposed a cache auditor that collect WOE statistics to aid cache channel detection.
- Our results showed that the proposed method is effective in identifying cache timing channels.

References


This work was supported in part by Semiconductor Research Corporation (SRC).