THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Hardware-based Cache Auditing to Aid Cache Timing Channel Detection*

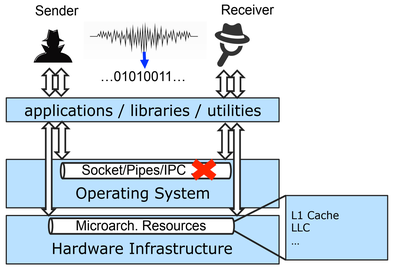
Fan Yao, Hongyu Fang, Guru Venkataramani and Miloš Doroslovački
The George Washington University | Washington, DC

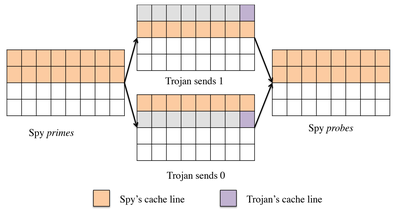
1. Background

❖ Timing channel attacks illicitly leaks 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





A typical cache timing channel: *Prime and Probe that communicates by creating mutual evictions.*

❖ 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]

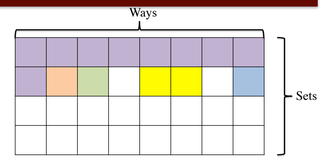
- ➕ Finer-grained statistics and higher effectiveness
- ➖ Do not provide high coverage and/or incur non-trivial overheads

❖ ***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



A simplified view of modern set-associative caches. Way replacement policy is typically unknown.

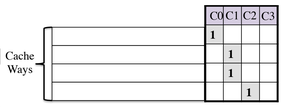
❖ Hardware-based Cache Auditing

- ***Capture the atomic communicating semantic***: Spy access ➔ Trojan Fill (full way occupancy) ➔ spy access (full occupancy destroyed)
- ***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

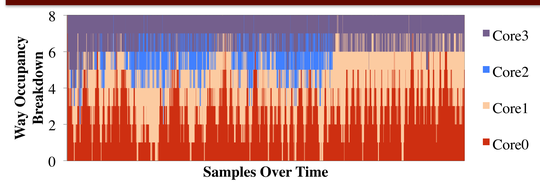


Way occupancy bookkeeping

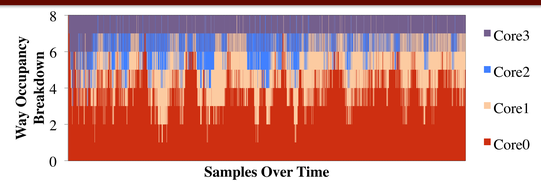
❖ Experimental Setup

- 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

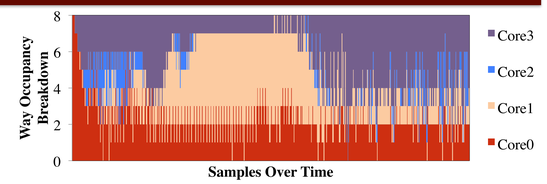
6. Cache Auditing Traces – Way Occupancy History



SPEC2006 wd: (*bzip2, lbm, hmmer, namd*)

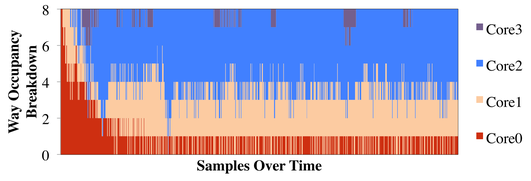


SPEC2006 wd: (*bzip2, libquantum, mcf, specrand*)

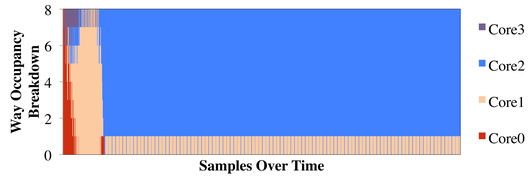


SPEC2006 wd: (*GemsFDTD, h264ref, omnetpp, xalancbmk*)

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



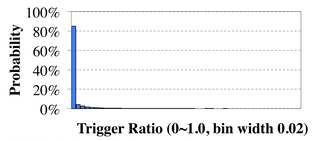
SPEC2006 wd: (*GemsFDTD, hmmer, xalancbmk, namd*)



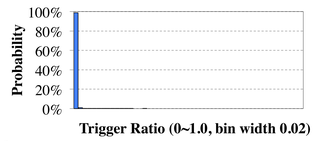
Timing Channel: (*bg1, Trojan, Spy, bg2*)

Trojan and spy are run together with two other background processes. Trojan is pinned on core2 and Spy is pinned on core 1.

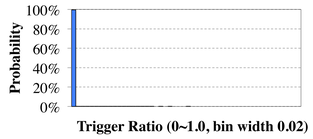
7. WOE Statistics



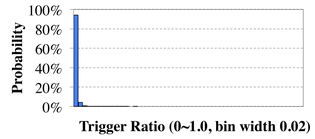
SPEC2006 wd: (*bzip2*, ***lbm***, *hmmer*, ***namd***)



SPEC2006 wd: (*bzip2*, *libq*, *mcf*, *specr*)

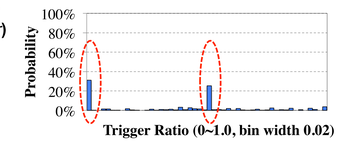


SPEC2006 wd: (*Gems*, *h264r*, *omn*, *xala*)



SPEC2006 wd: (*Gems*, ***hmmer***, ***xala***, *namd*)

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



Timing Channel: (*bg1*, ***Trojan*, ***Spy***, *bg2*)**

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

[1] M. Chiappetta, S. Erkay, and C. Yilmaz. "*Real time detection of cache-based side-channel attacks using hardware performance counters*." **Applied Soft Computing**, 2016.

[2] J. Chen, and G. Venkataramani. "*CC-hunter: Uncovering covert timing channels on shared processor hardware*." **MICRO**, 2014.

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