Secure Hierarchy-Aware Cache Replacement Policy (SHARP): Defending Against LLC-Based Side Channel Attacks

Mengjia Yan, Bhargava Gopireddy, Thomas Shull, Josep Torrellas
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

INTRODUCTION

LLC-Based Side Channel Attacks
- A spy process uses LLC-based side channel attacks to observe a victim program’s cache behavior to infer details about the program's security policies and leaves no trace.

Contributions
- The design of SHARP consists of:
  1) A new cache line replacement scheme to defend against conflict-based attacks
  2) A slightly modified clflush instruction to defend against flush-based attacks
- SHARP has desirable characteristics:
  • Prevents all known cross-core cache-based side channel attacks
  • Minimal performance overhead
  • No programmer intervention
  • Minor hardware modifications

BACKGROUND

Cache-based Side Channel Attacks
- An attack generally consists of:
  1) An offline phase to identify probe addresses
  2) An online phase to monitor victim’s execution. It repeats three steps: Eviction, Wait, and Analysis.

Eviction Strategies
- Conflict-based attacks: The spy creates cache conflicts to evict probe addresses, by accessing multiple cache lines in the same cache set as the probe address.
- Flush-based attacks: The spy shares cache addresses with the victim, and simply executes clflush instructions to evict the probe addresses from the cache.

SHARP DESIGN

Secure Hierarchy-Aware Replacement Policy
- The goal is to prevent a spy process from replacing shared-cache lines from a victim process that would create inclusion victims in the victim’s private caches.

MOTIVATION: ATTACK ANALYSIS

Conflict-Based Attacks
- All successful conflict-based attacks share two traits:
  1) They generate inclusion victims in the private cache of the core running the victim thread
  2) They exploit modern cache line replacement policies that do not properly defend against malicious creation of inclusion victims

Hardware Modifications
- 3 mechanisms to obtain private cache presence information with varied performance and scalability characteristics:
  1) Using core valid bits
  2) Using queries
  3) Using core valid bits and queries

A Modified clflush Instruction
- SHARP only allows clflush to be performed on pages with write permissions. With this restriction, sharing library code between processes and supporting page deduplication do not open up vulnerabilities to flush-based attacks.
  • Shared libraries ➔ Exceptions
  • Page deduplication ➔ Copy-on-write

EXPERIMENTAL SETUP

- MarxoX86 cycle-level full system simulator
- 2 to 16 out of order cores
- Private DL1, IL1, L2 (32KB, 32KB, 256KB)
- Shared Inclusive L3 cache (256MB slice per core)
- Baseline replacement policy: pseudo LRU

RESULTS

Defense Analysis
- Defending against attacks on GnuPG

Performance Evaluation
- 4-core PARSEC benchmark

CONCLUSIONS

- We make an important observation that all existing cross-core cache-based side channel attacks rely on creating “inclusion victims”.
- We propose the design of SHARP, which consists of a new cache line replacement scheme that prevents inclusion victims on other processes, and a slightly modified clflush instruction.
- We provide a simulation-based evaluation of SHARP that shows that it is effective against real-world attacks, and induces negligible average performance degradation.

ACKNOWLEDGEMENTS

This work was supported in part by NSF under grants CCF 1536795 and CCF 1649432.