SecDir: A Secure Directory to Defeat Directory Side Channel Attacks [ISCA’19]

Mengjia Yan, Jen-Yang Wen, Christopher W. Fletcher, and Josep Torrellas

{myan8,jwen11,cwfletch,torrella}@illinois.edu
University of Illinois at Urbana-Champaign

1. INTRODUCTION

Directory-based Side Channel Attacks:
- Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World [S&P’19]; Mengjia Yan, Read Sprabery, Bhargava Gopireddy, Christopher W. Fletcher, Roy Campbell, Josep Torrellas
- A spy process forces directory conflicts on shared directory to observe a victim program’s cache behavior to infer details about the program’s private information.
- A serious security threat in cloud. It bypasses software security policies and leaves no trace.

Contributions:
1) Design the first scalable secure directory, SecDir.
2) An analysis of the characteristics of SecDir, as well as the performance and cost evaluation.

2. CHALLENGES

Root Cause:
The limited set associativity of a directory slice.

Existing solutions are not effective.
1) Increase the associativity of the directory
   Unrealistic. Each directory slice requires:
   \[ W_{TD} + W_{ED} > W_{L2} \times (N_{core} - 1) + W_{L2} \]
   E.g., in an Intel Skylake-X processor with 8 cores, this requires a directory slice with more than 123 ways.
2) Way-partition the directory
   Inflexible, low performing, and limited. It requires:
   \[ W_{ED} \geq N_{core} \]

2. SecDir DESIGN

Key Idea:
Take part of the extended directory (ED) and configure it as a per-core victim directory (VD).

Corresponding directory operations:
1) core access
   Main Memory
   \[ (a) \text{Traditional.} \]
   \[ (b) \text{SecDir.} \]

SecDir Attributes:
1) Provides Isolation Inexpensively and Scalably.
2) Provides High Associativity.
3) Uses Low Area.
4) Delivers Efficient Directory Lookup.

3. VICTIM DIRECTORY DESIGN

Victim directory features:
- VD bank organization as a cuckoo directory.
- Early detection of VD Misses.
- Low area since there is no need to store sharer vector.

4. PERFORMANCE EVALUATION

We evaluated the following benchmarks:
- 12 mixes of single-threaded SPEC applications
- 10 multi-threaded PARSEC applications

Evaluation of the SPEC mixes
Evaluation of the PARSEC applications

Overall, <1% performance overhead.