CC-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware

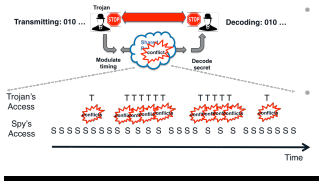
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Jie Chen | Guru Venkataramani | George Washington University | Washington DC, USA

1. Problem

- **Covert Channels illicitly leaks sensitive secrets to malicious parties**
  - Trojan (sender) and Spy (receiver) collude to subvert system security policy
- **Covert Timing Channels**
  - Covert Timing Channels are extremely stealthy
  - Very Challenging to detect and prevent

2. Covert Timing Channel on Hardware



- **Combinational structures**
  - E.g., compute logics and buses
  - Conflict ➔ contention patterns
- **Memory structures**
  - E.g., caches
  - Conflict ➔ block replacement

3. How to Detect Covert Timing Channels?

Detection Framework

Identify the event behind conflicts (contention)

↓

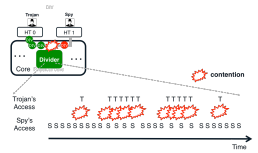
Construct event train

↓

Apply Pattern Detection Algorithms

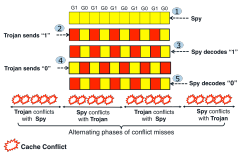
4. Design of Pattern Detection Algorithms

- **Detection on Combinational Structures**



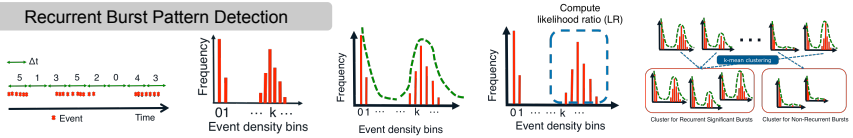
An Integer Divider Covert Timing Channel example

- **Detection on Memory Structures**



A Cache Covert Timing Channel example

Recurrent Burst Pattern Detection



Step 1 Determine Δt for counting the # of contention events

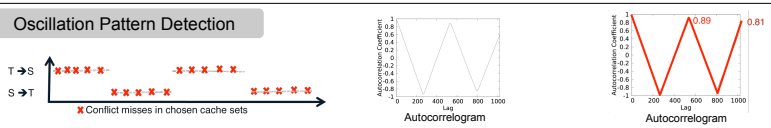
Step 2 Construct event density histogram

Step 3 Find bursts of contention events

Step 4 Identify significant bursts

Step 5 Check for recurrence of significant bursts

Oscillation Pattern Detection



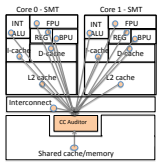
Step 1 Identify conflict miss events, and construct conflict miss event train for one OS time quantum

Step 2 Apply autocorrelation to event train, and construct autocorrelogram

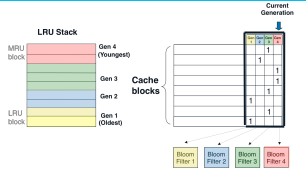
Step 3 Look for oscillation in the autocorrelogram, that is, autocorrelation coefficients show significant periodicity.

5. Hardware Support

- **Dedicated auditor unit: CC Auditor**
  - Gathers covert timing channel related events
  - Audits two hardware events at any give time



- **Conflict Miss Tracker**
  - Tracks cache conflict miss events
  - A practical design based on generation bits



- **Cache conflicts recorder**
  - Two alternating 128-byte vector registers
  - record the hardware context ID of the replacer and victim
- **Hardware histogram buffer**
  - 32-bit count-down register for Δt
  - 16-bit register for event density in each Δt
  - 128-entry histogram

6. Software Support

- **Software API**
  - Places a microarchitectural unit under audit
  - OS does privilege checks before letting the user to monitor the unit
- **Software monitor**
  - Accumulates all data from hardware auditor
  - Could be scheduled to run on un-audited cores

7. Experimental Setup

- **Cycle accurate full system simulator MARSSx86**
  - Simulates a Quad core processor, 2.5 GHz, with two hyperthreads
- **Test on two realistic covert timing channels**
  - Integer divider and Shared L2 cache
- **Evaluation uses combinations of**
  - I/O-intensive Filebench server benchmark
  - Memory-intensive Stream benchmarks
  - SPEC2006 CPU benchmarks

8. Experimental Results

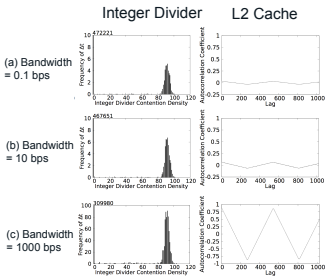


Fig. 1 Varying bandwidth Rates

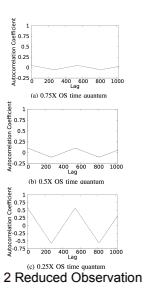


Fig. 2 Reduced Observation Window for 0.1bps Cache Channels

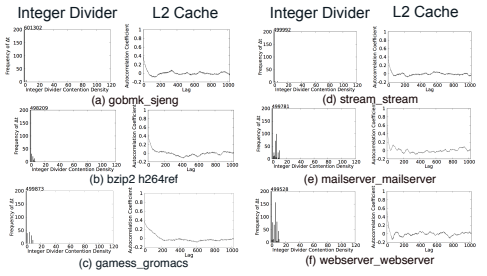


Fig. 3 SPEC'06, Stream & FileBench Benchmark False Alarm Test