HomeAlone: Co-Residency Detection in the Cloud via Side-Channel Analysis

Authors:
Yinqian Zhang Ari Juels Alina Opera Michael K. Reiter

Presented by:
Michael Christakos

2011 IEEE Symposium on Security and Privacy

Basics - Cloud Computing

- Computing resources are available as Virtual Machine (VM) instances
  - These VMs are managed by a hypervisor
    - Analogous to OS managing applications
  - Hypervisor handles I/O, core migration, time slots, etc.

Basics - Cloud Computing

- Private
  - Intended for only a single tenant (single organization)
- Public
  - Intended for multiple tenants (provided by Amazon, IBM, etc.)
    - Vulnerable to side-channel attacks, particularly using the L2 cache
      - L2 cache is a widely known and used vulnerability
      - Anything using the same cache can read the cache

Basics

- Cache - Smaller, much faster than main memory
  - L1 cache - fastest, smallest, most expensive ~ few ns
  - **L2 cache - slower, larger than L1**
  - L3 cache - slower, larger than L2 - (not always available)
  - main memory - significantly largest, slowest ~ 100’s of ns

Basics - Cloud Computing

- Many organizations have Service Level Agreements (SLA) that guarantee physical isolation
  - The entire physical machine is dedicated to a single organization
  - PROBLEM: How can the organization verify that they have sole access to the physical device?
  - SOLUTION: The HomeAlone VM

Basics - HomeAlone Approach

- PRIME-PROBE detection
  - PRIME - Read large section of main memory to fill up a section of the cache
  - IDLE - Wait a period of time to allow other VM's to run and potentially use the cache
  - PROBE - Reread the same section and compare access time to determine if the watched section of cache was overwritten
- Uses 1/16th of available cache to test

HomeAlone VM Classifications

- Friendly
  - Another VM running by the same organization.
  - It is expected and wanted.

- Foe
  - Another VM not from the same organization.
  - It should not be on the same physical machine

HomeAlone: Foe VMs

- Benign
  - Not actively attacking
  - May or may not be aware of other VMs
  - Result of accidental or purposeful breach of SLA physical isolation
- Adversarial
  - Actively attempting to exploit co-residency
  - Attempting to disrupt or gather information through L2 cache side-channel

Default Cache Usage

RESERVED

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Cache Usage - Prepare to Monitor

1. Choose set to monitor (#7) - Tell Friendly VMs

RESERVED

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Cache Usage - Prepare to Monitor

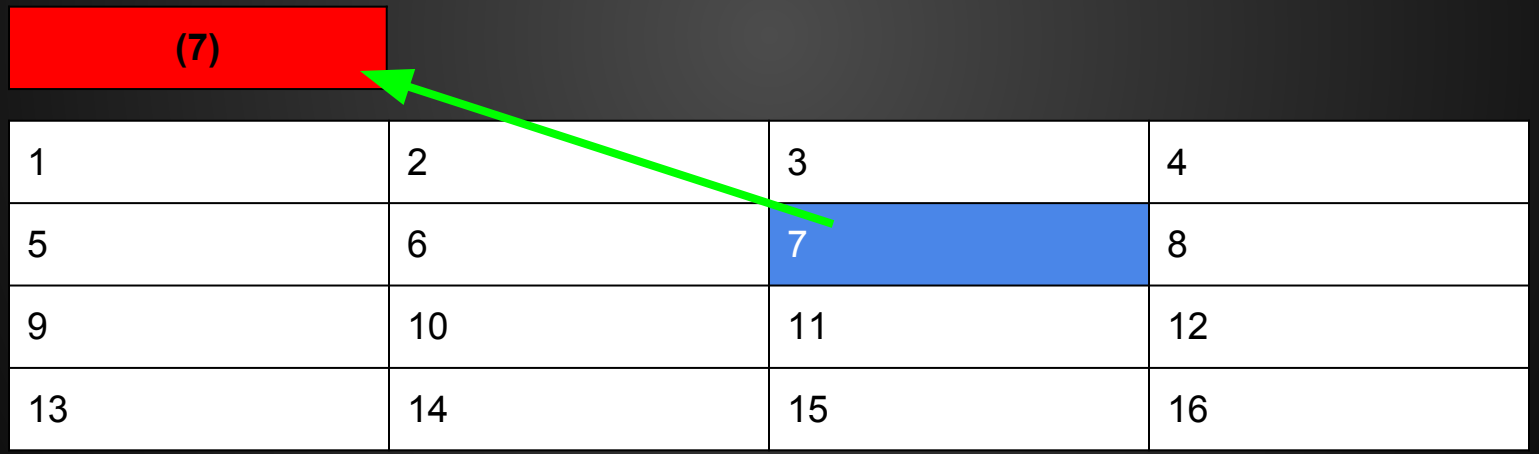
1. Choose set to monitor (#7) - Tell Friendly VMs
2. Gather activity profiles from Friendly VMs (Total # I/O bytes)

RESERVED

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Cache Usage - Prepare to Monitor

1. Choose set to monitor (#7) - Tell Friendly VMs
2. Gather activity profiles from Friendly VMs (Total # I/O bytes)
3. Copy data in #7 to Reserved Space



Cache Usage - Prepare to Monitor

1. Choose set to monitor (#7) - Tell Friendly VMs
2. Gather activity profiles from Friendly VMs (Total # I/O bytes)
3. Copy data in #7 to Reserved Space
4. Whenever #7 is needed for normal use, map to reserved

(7)

1	2	3	4
5	6	7 (TEST AREA)	8
9	10	11	12
13	14	15	16

Cache Usage - Monitor

1. Read data from main memory into 7 (completely fill 7) <PRIME>

(7)

1	2	3	4
5	6	7 (TEST DATA)	8
9	10	11	12
13	14	15	16

Cache Usage - Monitor

1. Read data from main memory into 7 (completely fill 7) <PRIME>
2. WAIT long enough to allow other VMs to access cache (~30ms)

(7)

1	2	3	4
5	6	7 (TEST DATA)	8
9	10	11	12
13	14	15	16

Cache Usage - Monitor

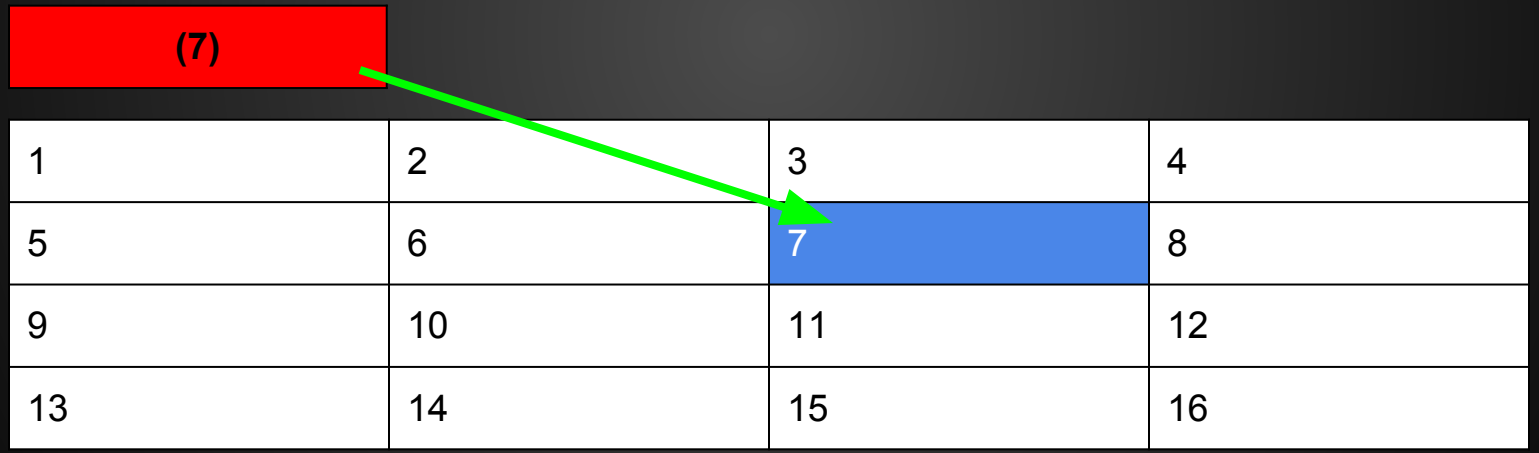
1. Read data from main memory into 7 (completely fill 7) <PRIME>
2. WAIT long enough to allow other VMs to access cache (~30ms)
3. Read same data from step 1 and measure access time <PROBE>

(7)

1	2	3	4
5	6	7 (TEST DATA)	8
9	10	11	12
13	14	15	16

Cache Usage - Monitor

1. Read data from main memory into 7 (completely fill 7) <PRIME>
2. WAIT long enough to allow other VMs to access cache (~30ms)
3. Read same data from step 1 and measure access time <PROBE>
4. Restore data from reserved space back to 7 (tell Friendly VMs)



Cache Usage - Monitoring Complete

1. Cache is back to normal use (RESERVED is off-limits again)

RESERVED

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Interpret Access Time

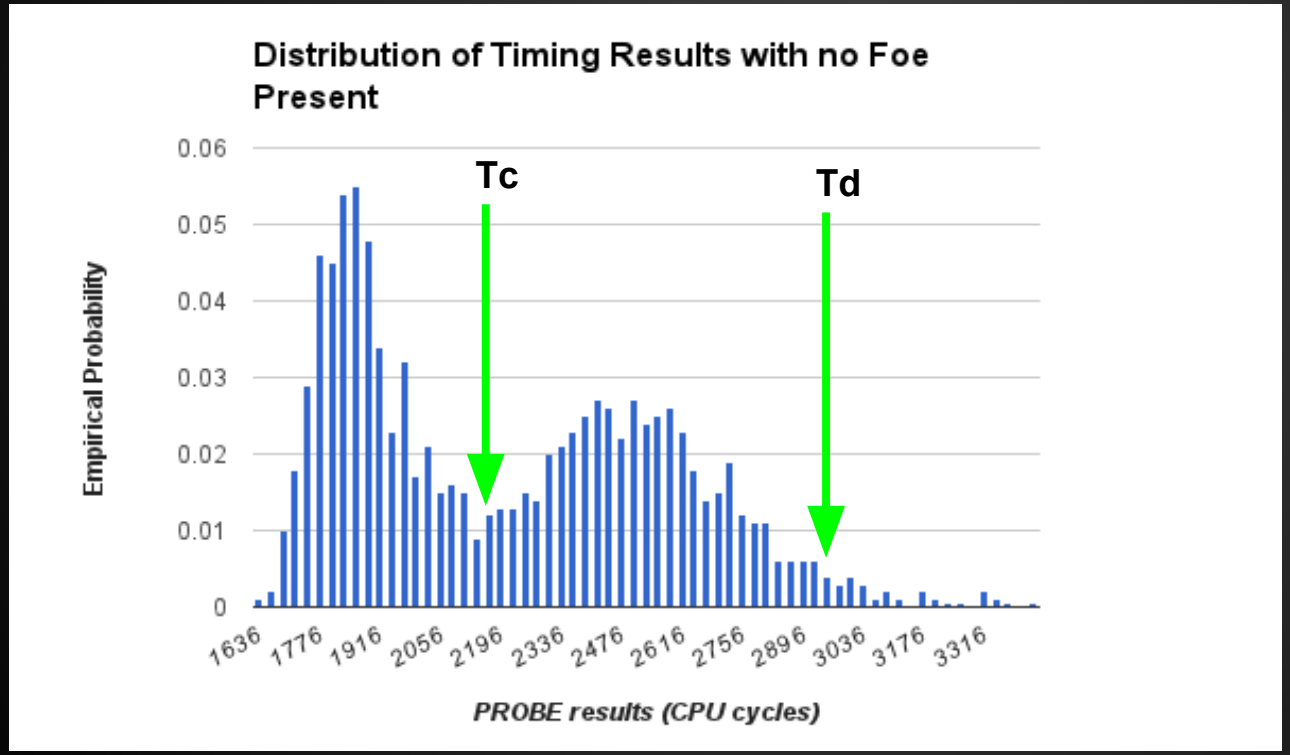
1. Use activity profiles to determine expected level of cache usage by hypervisor
   a. The hypervisor handles all I/O operations and may use area of cache (if the VM is on the same cache)
   b. If access time is above thresholds, mark as foe present
2. Repeat test n times
3. Determine probability of foe being present

Training

- Run on same architecture many times to determine normal thresholds
  - Run in absence (or assumed absence) of Foe VM
  - Threshold values dependent on activity profiles

Training

- **Tc** - Threshold when hypervisor is on a different cache
- **Td** - Threshold when hypervisor is on same cache



Training

- Run on same architecture many times to determine normal thresholds
  - Run in absence (or assumed absence) of Foe VM
  - Threshold values dependent on activity profiles

- Determine Tc and Td such that they are at (100 - α) percentile
  - α = desired level of false positives
  - Tc is for distribution with hypervisor on separate cache, in CPU cycles
  - Td is for distribution with hypervisor on this cache, in CPU cycles

Implementation - No Foe

- Cache test size ALWAYS 256 sets (1/16)

- 4 Friendly VMs
  - 1 running apache2 server
  - 3 running one of PARSEC benchmark applications

Implementation - Benign Foe

- 4 Friendly VMs
  - 1 running apache2 server
  - 3 running one of PARSEC benchmark applications

- 1 Foe VM
  - 1 running one of PARSEC benchmark applications

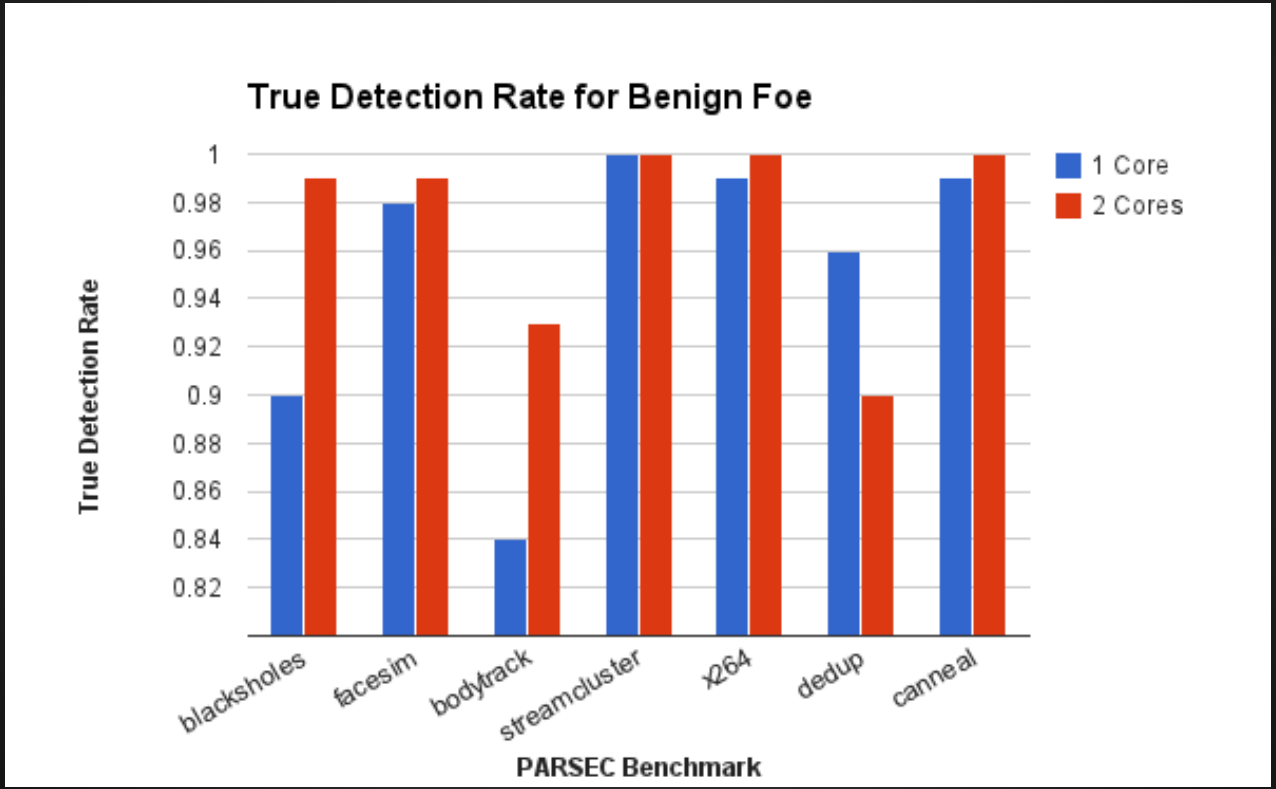
Implementation - Adversarial

- 4 Friendly VMs
  - 1 running apache2 server
  - 3 running one of PARSEC benchmark applications
- 1 Foe VM
  - Toy program allocates buffer much larger than size of cache
  - Randomly reads from locations in buffer
  - Access frequency can be changed to test different range of foe actions

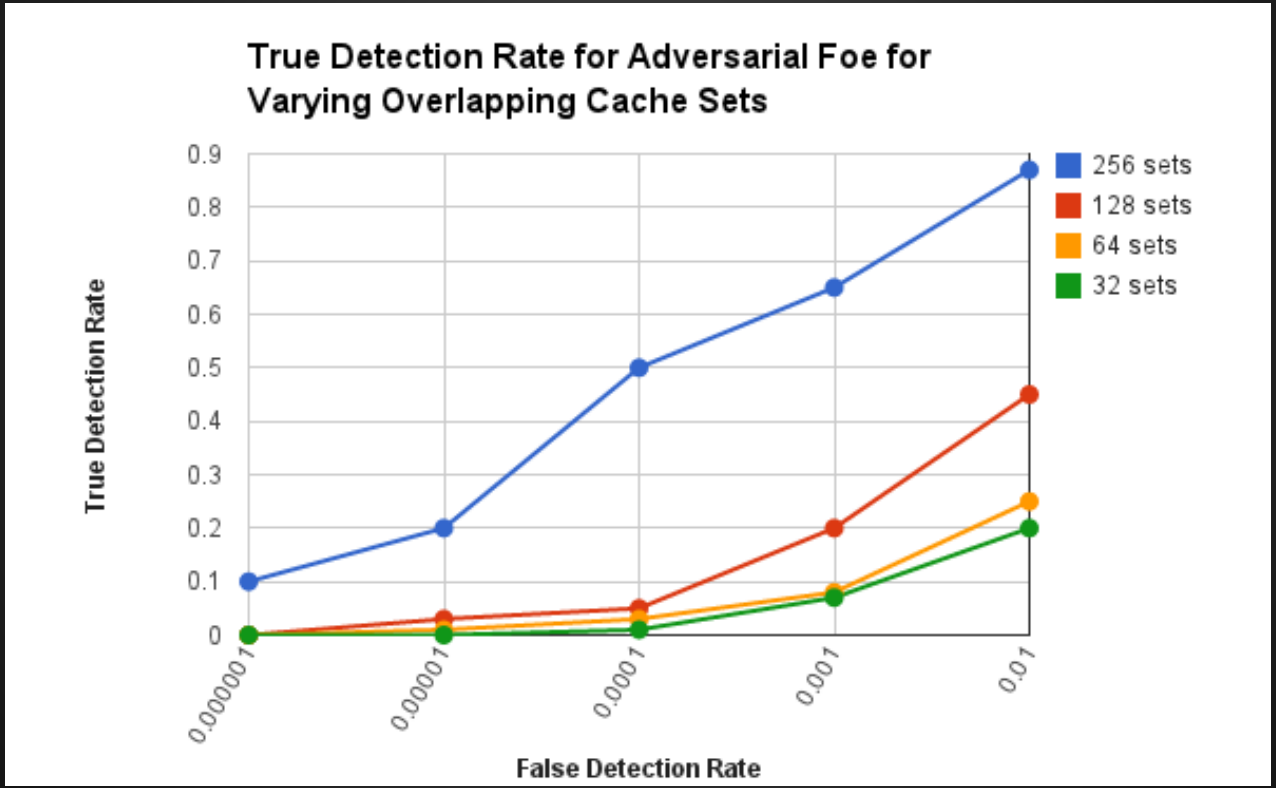
PARSEC Benchmark Applications

- Blackscholes - financial analysis
- Bodytrack - video/animation applications
- Canneal - engineering applications
- Dedup - next gen. backup storage
- Facesim - computer games
- Streamcluster - data mining
- x264 - next gen. video systems

Detection Rates for Benign Foe



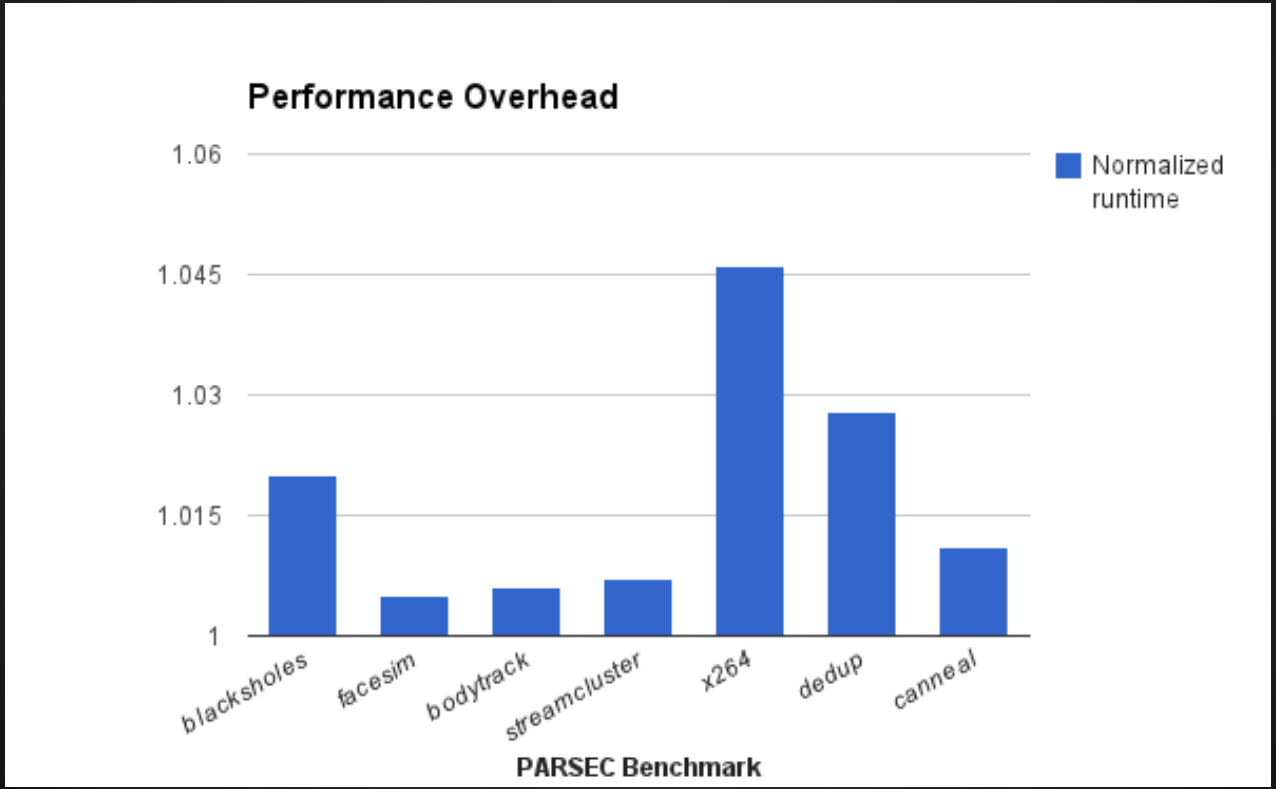
Detection Rates for Adversarial Foe



Detection Against Adversarial Foe

- Detection related to number of overlapping sets of cache
- Detection related to rate of false positives
  - False positive rate determines thresholds Tc and Td
- Detection related to frequency of Foe VM probing
  - Higher probing rate generates more cache corruption and easier to detect

Computational Cost



Limitations

- Only measures a section of the cache at a time
  - If this section is not used by the Foe VM during this period, it will not be detected
  - Measuring this section requires Friendly VMs to not use this section during this period
  - Multi-core/multi-cache - can only detect Foe if on the same cache
    - Only vulnerable if on same cache

Conclusions

- HomeAlone uses vulnerability for defence
  - Can determine SLA violations (benign or adversarial)
- Modest computational cost (<5%)
- No change to hypervisor required
- Foe VM can avoid detection by not using the L2 cache
  - This robs Foe of a major attack avenue