
Fangfei Liu, Qian Ge, Yuval Yarom, Frank McKeen, Carlos Rozas, Gernot Heiser, Ruby B. Lee


Presented by

RAVIKUMAR SELVAM
Motivation

Hardware

VMM

Guest OS #1

VM

Guest OS #2

VM

Hardware
Motivation
Motivation
## CROSS-VM SIDE CHANNEL ATTACK

**FLUSH–RELOAD ATTACK**

*Phase–I:* Flush the cache hierarchy.  
*Phase–II:* Attacker waits idle.  
*Phase–III:* Reload the memory line and measure access time.  

If the memory line available in the memory, the reload takes shorter time, otherwise it has to fetch content from memory.

**PRIME–PROBE ATTACK**

*Phase–I:* Fill the cache set with known data.  
*Phase–II:* Attacker became idle  
*Phase–III:* Probe the cache set by measuring access time.  

If the victim access the target cache line the target cache line will evict an attacker line from the cache, resulting in a longer time than otherwise.
ILLUSTRATION – *PRIME PROBE ATTACK*

Victim Core

Attacker Core

L1–Cache

L2–Cache

L3–Cache
ILLUSTRATION – **PRIME PROBE ATTACK**

Victim Core  
Attacker Core

L1–Cache

L2–Cache

L3–Cache
ILLUSTRATION – *PRIME PROBE ATTACK*

Victim Core

L1–Cache

Attacker Core

L2–Cache

L3–Cache

12/8/2016
ILLUSTRATION – PRIME PROBE ATTACK

Victim Core

Attacker Core

L1–Cache

L2–Cache

L3–Cache
THINKING ABOUT SOLUTION....!

1. Scalable Solution
2. Practical Solution
3. Solution should not increase the risk of other attacks

Other factor like Performance cost.

“Goal is providing a system-level mechanism that offers strong security guarantees to cloud providers and their customers.”
CACHE ALLOCATION TECHNOLOGY

• Introduced in Intel Haswell Processors in 2014.

• Supports configurable partitioning of the LLC.

• A bitmask defines the LLC ways to access Class of Service (COS).

• Configured in software, to one or four model-specific registers (MSRs).

• Application running on a logical processor share the same COS.
CACHE ALLOCATION TECHNOLOGY

• Cache Partition Mechanism.
SOLUTION – CATalyst

• CAT as a pseudo-locking mechanism, which pin certain page frames in the LLC – Secure pages.

• Idea is to create two partition of LLC
  1. Secure Partition - managed by Software (at VMM boot time)
  2. Non-Secure Partition - managed by Hardware

• VMM guarantees – secure page is not shared by multiple VMs.
• Secure pages are fully managed by guest OS.
SOLUTION – CATalyst

• Security guarantees:
  1. No eviction of secure pages by malicious code.
  2. No overlapping of secure pages between different active VMs (security domains).

• Scalability
  • Secure partition contains $32 \times k \times N_{\text{slice}}$; $k \geq 2$

![Diagram](image-url)
SOLUTION – CATalyst

• CATalyst system consists of 4 phases:
  1. VMM boot time
  2. VM launch time
  3. VM run time
  4. VM terminate time

  How the secure pages managed, how to expose the secure pages to the guest VM, and how to reclaim the secure pages
VMM boot time

• Pseudo-locking mechanism

<table>
<thead>
<tr>
<th>Example – bit mask configuration for CAT as a pseudo locking mechanism</th>
</tr>
</thead>
</table>
| \( \begin{array}{cccccccccc}
M0 & M1 & M2 & M3 & M4 & M5 & M6 & M7 & M8 & M9 \\
\hline
\text{COS 0} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\text{COS 1} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\text{COS 2} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\text{COS 3} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \) |

• ‘SET’ – MSR associated with the logical processor
VMM boot time

- Secure pages always hit in the secure partition of LLC.
- All the memory traffic initiated by the logical processor will follow the cache allocation policy of COS.
- Pre-loading routine and any non-sensitive data, must never be loaded into the secure partition, to avoid evicting the secure partition.
- No interrupt is allowed while the preloading routine is executing.
- No TLB miss is allowed while the preloading routine executing.
- To guarantee that the secure pages are never cached in any part of the non-secure partition.

**Preloading routine**

Input: Secure page P to be pinned.

C: code that loads a secure page, P

1) Disable local interrupts
2) Access C
3) Access one word in P
4) Set current logical processor to COS3
5) clflush P
6) Access P
7) Set current logical processor back to COS0
8) Enable local interrupts
VMM boot time

• Preloading secure pages without self-conflicts.
  • Physically contiguous chunk with size equal to the size of the secure partition.
  • Problem: when the core count is not a power of two.
  • VMM needs to know whether adding the page frame into secure partition will cause eviction
    1. Reverse Engineer the hash function.
    2. PRIME + PROBE technique.

• DDIO – enables direct communication to host processor’s cache.
VMM boot time – other issues

• Special instruction explicitly flush the cache.
  • Instruction “invd” and “wbinvd”.
  • Instruction “clflush” The loading of secure pages at boot time, reloading them requires special care.

• Loading of secure pages at boot time, reloading them requires special care.
  • Avoid concurrent access of cache across VM.
  • Stop all logical processors while VMM reloading the secure pages
VMM launch time & VMM terminate time

• VMM allocates secure pages to the guest VM as required.
  
  • long do_alloc_secure_pages(unsigned long start_gpfn, int nr_pages)
    1. Free the original page frames.
    2. Allocate secure pages.
    3. Modify the EPT entries to map to the secure pages.

• Secure pages are reclaimed and reloaded into the secure partition.

  • long do_release_secure_pages(void)
    1. Allocate a free machine page frame and change the EPT to point to the new machine.
    2. Free secure pages.
    3. Ensure that the secure page in pinned in the cache.
VMM run time

- VM’s kernel manages the secure pages by multiplexing them to all the process.
  - Virtualizing the secure pages for users processes.
  - A user process can exclusively hold secure pages until they are explicitly release.
  - `long sys_map_sp(unsigned long start, size_t len);`
    Uses mlock to make sure the memory region has the VM_LOCKED flag set.
  - `long sys_unmap_sp(unsigned long start, size_t len);`
SECURITY EVALUATION

Experimental environment

<table>
<thead>
<tr>
<th>Processor Model</th>
<th>Intel Xenon E5 2618L v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microarchitecture</td>
<td>Haswell</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>2.3GHz</td>
</tr>
<tr>
<td># cores</td>
<td>8(8 slices)</td>
</tr>
<tr>
<td>LLC</td>
<td>20-way 20MB</td>
</tr>
<tr>
<td># way (secure partition)</td>
<td>2</td>
</tr>
<tr>
<td>VMM</td>
<td>Xen 4.5.0 (HVM)</td>
</tr>
<tr>
<td>Dom0 and DomU</td>
<td>CentOS 6.6</td>
</tr>
<tr>
<td>Kernel Version</td>
<td>Linux 3.14.0</td>
</tr>
<tr>
<td>Legacy VM</td>
<td>4 VCPUs, 4GB RAM</td>
</tr>
<tr>
<td>Secure VM</td>
<td>4 VCPUs, 2GB RAM, 8 secure pages</td>
</tr>
<tr>
<td>Algorithm</td>
<td>GNUPG 1.4.13</td>
</tr>
<tr>
<td></td>
<td>Square and Multiply exponentiation</td>
</tr>
</tbody>
</table>

12/8/2016
PERFORMANCE EVALUATION
PERFORMANCE EVALUATION

<table>
<thead>
<tr>
<th>Time to encrypt a 5MB file using AES</th>
<th>5KB</th>
<th>50 KB</th>
<th>500 KB</th>
<th>5 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Size</td>
<td>5KB</td>
<td>50 KB</td>
<td>500 KB</td>
<td>5 MB</td>
</tr>
<tr>
<td>Baseline (ms)</td>
<td>81.04</td>
<td>79.91</td>
<td>79.83</td>
<td>82.87</td>
</tr>
<tr>
<td>CATalyst (ms)</td>
<td>95.26</td>
<td>81.96</td>
<td>79.96</td>
<td>82.83</td>
</tr>
<tr>
<td>Slowdown</td>
<td>17.54%</td>
<td>2.56%</td>
<td>0.16%</td>
<td>-0.05%</td>
</tr>
</tbody>
</table>

With the small buffer, the overhead of the map and unmap system calls is significant resulting in a slowdown of 17.5%.

A programmer can evoke the mapping at library initialization, therefore avoiding the overhead for every encryption.
CONCLUSION

• CATalyst, a lightweight system mechanism.

• CATalyst builds on existing commodity hardware, and can be easily deployed.

• Users can load and lock the security-sensitive code and data by mapping them to the cache-pinned pages.

• CATalyst – Xen hypervisor and the Linux OS.

• Increase security with small performance degradation.
REFERENCE


THANKS FOR YOUR ATTENTION

ALL THE BEST

FOR FINALS