Exploiting Out-of-Order-Execution
Processor Side Channels to Enable Cross VM Code Execution

Sophia D’Antoine
REcon 2015
The Cloud

DEMAND A SAFER CLOUD

client-side encryption, server-side computation
beginning with encrypted search

LEARN MORE
Cloud Computing (IaaS)

- Virtual instances
- Hypervisors

Dynamic allocation

=> Reduces cost
Everyone’s Happy
Problems with the Cloud

Security issues with cloud computing

- Sensitive data stored remotely
- Vulnerable host
- Untrusted host
- Co-located with foreign VM's
Physical co-location leads to side channel vulnerabilities.
Cloud Hardware

Exploiting Out-of-Order-Execution
Universal Vulnerabilities

1) **Translation** between physical and virtual hardware based on need

2) Allocation causes **contention**

3) Private VM activities **not opaque** to co-residents
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the processor
4. Importance of memory models
5. Design of an Out-of-Order-Execution channel
6. Demo
7. Conclusion
Side Channel Attack

“In cryptography, a side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem”

Cloud Computing

- Hardware side channel
- Cross virtual machine
- Information gained through recordable changes in the system
Classification S/R Model

- **Hardware agnostic**
- **Two** methods of interacting
  - Transmit
  - Receive
Possible Exploits

- **Receive (exfiltrate)**
  1. crypto key theft
  2. process monitoring
  3. environment keying
  4. broadcast signal

- **Transmit (infiltrate)**
  1. DoS
  2. co-residency

- **Transmit & Receive (network)**
  1. communication (C&C)
Communication

Virtual Allocations

Shared Hardware

Communication Medium

VM1

S
R

VM2

S
R

Client

S
R

Master VM

06/19/2015

Exploiting Out-of-Order-Execution
Cache Side Channel Example [3]

Flush+Reload targets the L3 Cache Tier

- Receiving Mechanism (Adversary)
  - Flushes & queries
- Transmitting Mechanism (Victim)
  - Accesses same L3 line
- Leaked GnuPG Private Key
Pipeline vs Cache Channel

**Benefits:**
- Quiet, **covert** channel
- Not affected by **cache misses**, etc.
- Channel & noise **amplifies** in a crowded cloud environment
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the pipeline
4. Importance of memory models
5. Design of an Out-of-Order-Execution channel
6. Demo
7. Conclusion
The Attack Vector

Side Channels which Exploit Hardware Vulnerabilities Inherent to Modern Cloud Computing Systems

Requirements:

- **Shared** hardware
- **Dynamically** allocated hardware resources
- **Co-Location** with adversarial VMs or infected VMs
Pipeline Side Channel

We chose to target the processor as the hardware medium.

=> CPU's pipeline
=> System artifacts queried dynamically

- Instruction order
- Results from instruction sets
Out-of-Order-Execution

Exploiting Out-of-Order-Execution
Processor Pipeline Contention

- VM
  - Process01
  - SMT Optimizes Shared Hardware
- VM
  - Process02
- VM
  - Process03
- VM
  - Process04

Core01
Core02

Processor

Pipeline Executing Instructions From Foreign Applications

Exploiting Out-of-Order-Execution
RECEIVER
8.2.3.4 Loads May Be Reordered with Earlier Stores to Different Locations

The Intel-64 memory-ordering model allows a load to be reordered with an earlier store to a different location. However, loads are not reordered with stores to the same location.

The fact that a load may be reordered with an earlier store to a different location is illustrated by the following example:

**Example 8.3. Loads May be Reordered with Older Stores**

<table>
<thead>
<tr>
<th>Processor 0</th>
<th>Processor 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov [ _x], 1</td>
<td>mov [ _y], 1</td>
</tr>
<tr>
<td>mov r1, [ _y]</td>
<td>mov r2, [ _x]</td>
</tr>
</tbody>
</table>

Initially x = y = 0  
r1 = 0 and r2 = 0 is allowed
Record Out of Order Execution

<table>
<thead>
<tr>
<th>THREAD 1</th>
<th>THREAD 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synched</strong></td>
<td><strong>Asynched</strong></td>
</tr>
<tr>
<td><code>store [X], 1</code></td>
<td><code>store [Y], 1</code></td>
</tr>
<tr>
<td><code>load r1, [Y]</code></td>
<td><code>load r2, [X]</code></td>
</tr>
</tbody>
</table>

**Synched**
- `r1 = r2 = 1`

**Asynched**
- `r1 = 0`
- `r2 = 1`

**Out of Order Execution**
- `r1 = r2 = 0`
int X, Y, count_OoOE;

....initialize semaphores Sema1 & Sema2...

pthread_t thread1, thread2;
pthread_create(&threadN, NULL, threadNFunc, NULL);

for (int iterations = 1; iterations++)

    X, Y = 0;

    sem_post(beginSema1 & beginSema2);

    sem_wait(endSema1 & endSema2);

    if (r1 == 0 && r2 == 0)
        count_OoOE ++;

Averages matter
TRANSMITTER

Exploiting Out-of-Order-Execution
Force Out of Order Execution

Memory Fences

Mfence:
- x86 instruction full memory barrier
  prevents memory reordering of any kind
- order of 100 cycles per operation

```
... mov dword ptr [_spin1], 0
  ... mfence

... mov dword ptr [_spin2], 0
  ... mfence
```
Force Out of Order Execution

THE PIPELINE

... ... NOP Store [X], 1 mfence Load r1, [X] NOP ...

06/19/2014

Exploiting Out-of-Order-Execution
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the processor
4. Importance of memory models
5. Design of an Out-of-Order-Execution channel
6. Demo
7. Conclusion
Types of Memory Reordering

- Compilation Time
  - GCC Multithreaded Programs
- Processor (Run) Time
  - OoOE Execution MultiCored (MultiExecution Processors) Computers

Exploiting Out-of-Order-Execution
Types of Memory Reordering

Dynamic side channel artifacts

Processor
(Run)
Time

OoOE Execution
MultiCored
(MultiExecution
Processors)
Computers
Weak Memory Models [7]
Types of Memory Reordering

4 types of run time reordering barriers

- Instruction A visible to all processes before B occurs
- #StoreLoad most expensive operation
Force Out of Order Execution

Memory Barrier

- ‘Lock-free programming’ on SMT multiprocessors
- #StoreLoad unique prevents r1=r2=0
- x86: mfence (effects the pipeline)
Channel Transmitter (Victim)

• Force Out-of-Order-Execution patterns
• Affect the order of stores and loads
• Time frame dependant
• x86: mfence
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the processor
4. Importance of memory models
5. Design of an Out of Order Execution channel
6. Demo
7. Conclusion
Lab Model

Scheduler **Xen** hypervisor
- Popular commercial IaaS platforms

**Xeon** Processors
- **Shared** multi-core/ multi-processor hardware
- **8** logical CPU’s/ **4** cores
- **6** virtual machines (VM’s)
- **Parallel Processing/ Simultaneous Multi-Threading** On (SMT)
Virtual Machines

- 6 Windows 7 VM’s
Virtual Machine S/R

Exploiting Out-of-Order-Execution

06/19/2015
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the processor
4. Importance of memory models
5. Design of an Out-of-Order-Execution channel
6. Demo
7. Conclusion
Demo Links

sophia.re/sender.py

sophia.re/receiver.py

Exploiting Out-of-Order-Execution
Overview

1. Introduction
2. Cloud exploitation techniques
3. Targeting the processor
4. Importance of memory models
5. Design of an Out-of-Order-Execution channel
6. Demo
7. Conclusion
Potential Channel Mitigation

Protected Resource Ownership

- Isolating VM’s
- Turn off hyperthreading
- Blacklisting resources for concurrent threads
- Downside: cloud benefits
In Conclusion...

Contribution:

We demonstrate a novel Out of Order Execution side channel.

- **Dynamic** querying/forcing method
- **Application** to cloud computing
- **Mitigation** techniques
Acknowledgements

- Jeremy Blackthorne
- RPISEC
- Trail of Bits
Any Questions?

IRC: quend (#rpisec, #pwning)
email: sophia@trailofbits.com

06/19/2015
References