Side Channels in the Cloud: Good News and Bad News

Michael Reiter
University of North Carolina at Chapel Hill

July 16, 2012
Side Channels

Extracts information from a computation based on its *implementation* ...

... despite the fact that its implementation is faithful to its design (and the design is correct)

Usually (but not always!) used to attack cryptographic implementations

The target value is a cryptographic key
The attacker runs a program on the system that is performing the cryptographic operation of interest.

Basic idea: observe computation’s effects on the system, and learn information from that.

Recent attacks are *asynchronous*, in that they do not require the attacker to achieve precisely timed observations of the victim.

Utilize SMT or ability to game the OS scheduler
None shown to work in virtualized SMP settings
CPU Cache

4-way set associative cache

Physical Address

Cache Set

Cache Line
**PRIME-PROBE Protocol**

**PRIME**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PRIME-PROBE Protocol

PRIME

PRIME-PROBE Interval
PRIME-PROBE Protocol

PRIME

PROBE

PRIME-PROBE Interval

500 400 600 500 400 700 400 500
Virtualization adds a layer of software between the attacker and the victim
Cross-VM Side Channels
Challenge #1: Observation Granularity

- Scheduling quantum of Xen is 30ms
- Does not permit many observations of a crypto op
Numerous HW and SW sources of cache noise

Hardware: TLB misses, power saving, ...
Software: Hypervisor, Dom0, ...

Cross-VM Side Channels
Challenge #2: Observation Noise
Cross-VM Side Channels
Challenge #3: Core Migration

Attacker VM and victim VM will migrate across cores over time
So, many observations might not be the victim
Attacker VM and victim VM will migrate across cores over time
So, many observations might not be the victim
1. Game the Xen scheduler to interrupt victim with sufficient frequency
2. Classify cache-pattern observations, first individually and then in sequence
   Yields a collection of “execution fragments”
3. Apply customized sequence reconstruction algorithms to fragments
   Corrects errors and assembles full key “template”
4. Exhaustively search remaining key possibilities
Attacked the implementation of ElGamal decryption in libgcrypt v.1.5.0
Specifically loaded the victim VM with Gnu Privacy Guard (GnuPG) v.2.0.19

Utilized the I-cache on a single-socket quad-core processor (Intel Core 2 Q9650)
Xen 4.0 as virtualization substrate
Each VM ran Ubuntu 10.04 server with Linux kernel 2.6.32.16

Scheduler was work-conserving
Non-work-conserving also possible, but harder
Victim utilized a 4096-bit ElGamal modulus
Private exponent was 457 bits

Victim repeatedly performed decryptions, as if they could be triggered by the attacker
Done simply to speed up the test

~30,000,000 prime-probe trials over ~6 hours

After of several hours of post-processing, key space narrowed to 9,862 possibilities
Exhaustive search easily identified the key
Physical Isolation

A natural defense is to *physically isolate* VMs
Customer has exclusive use of a physical machine

Amazon offers dedicated instances in virtual private cloud
Confirming Physical Isolation

Cloud provider may accidentally violate service level agreement or take shortcuts
Configuration error
Lower cost

Cloud customer has no control or visibility into the virtualization layer
Verification and auditing is difficult
Friendly VMs: VMs controlled by the legitimate tenant
Foe VMs: unexpected third party VMs
Friendly VMs

Xen Hypervisor
Home Alone
Address Remapping in PVM

Page Table Entries

Pseudo-physical Pages

Physical Pages

Physical Address

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>0x2000</td>
<td>0x3000</td>
<td>0x4000</td>
<td>0x5000</td>
</tr>
</tbody>
</table>
Home Alone
Address Remapping in PVM

Pseudo-physical Pages

Physical Pages

L2 Cache

Data copy

Avoided pages
Reserved pages
Home Alone
Distribution of PROBE Results

Empirical probability

PROBE results with NO foe present (CPU cycles)
Home Alone
Cache Activity Classifier

Different friend I/O level
Cache region
Select Monitoring VM
Xen Hypervisor
Xen Hypervisor

Home Alone
Putting Everything Together
Home Alone
Putting Everything Together

Does the PROBE result fall into class A or class B?

I am the next Monitoring VM

Select next Monitoring VM

Class A

Class B

Xen Hypervisor
“Cloud” platform: Intel Core 2 Quad processor, two shared L2 cache, no SMT

HomeAlone implemented in 64-bit PVOps Linux kernel 2.6.32 for Xen 4.0

Classifier parameters:
- Cache avoided: 1/16th
- PRIME-PROBE interval: 30 ms
- Detection period: 25 PRIME-PROBE trails
True detection rate (with 1% false positive)

Foe VM running cloud applications
- Simulated with PARSEC benchmarks: 84% - 100%
Foe VM running PRIME-PROBE protocol
- Less frequent, smaller cache region: 15%
- More frequent, larger cache region: 85%

Performance overhead
Address remapping: 150ms for remapping a 2GB memory (1/16 mapped to monitored cache region)
Less than 5% overhead during detection period
Conclusions

**Bad news:** Cross-VM side channels with sufficient fidelity to extract private keys are possible

**Good news:** Friendly VMs can use side channels to confirm that they are physically isolated from others