Scheduler-based Defenses against Cross-VM Side-channels

Venkat(athan) Varadarajan,
Thomas Ristenpart,
and Michael Swift
Shared Resources and Isolation

- IaaS Public clouds (Amazon EC2, Azure, etc.)
  - Multi-tenancy
- VMs share many resources
  - CPU, cache, memory, disk, network, etc.
- Virtual Machine Managers (VMM)
  - Goal: Provide Isolation
- Deployed VMMs don’t perfectly isolate VMs
  - Side-channels [Ristenpart et al. ’09, Zhang et al. ’12]
  - Other attacks: Performance Degradation, RFA [Varadarajan et al. ‘12]
Example Cache Side-channel*

Control-flow Side-channel
– secret key bits directly affect instruction sequence executed

I-cache usage leaks secret
– Operations: Square (S), Reduce (R), and Multiply (M).
– $e_i = 1$ bit: $S \rightarrow R \rightarrow M \rightarrow R$
– $e_i = 0$ bit: $S \rightarrow R$ (and, NOT followed by $M \rightarrow R$).

Modular Exponentiation Algorithm:

\[
\text{SQUAREMULT}(x, e, N):
\]

Let $e_n, ..., e_1$ be the bits of $e$
\[
y \leftarrow 1
\]
for $i = n$ down to 1 do
\[
y \leftarrow \text{SQUARE}(y)
y \leftarrow \text{MODREDUCE}(y, N)
\]
if $e_i = 1$ then
\[
y \leftarrow \text{MULT}(y, x)
y \leftarrow \text{MODREDUCE}(y, N)
\]
end if
end for
return $y$

* Zhang, Juels, Reiter and Ristenpart: Cross-VM Side-channel Attack, CCS 2012
Cache-based Side-channels

Translation & logical isolation layer

Core: VM VM VM VM

Preemption Relinquish

Cache sets

cache ways

Attacker Timing Profile

Extract secret information
Defenses against Side-channels

Access-driven side-channel attacks rely on:

1. **Sharing**
   - 1.1. Resource Partitioning or Hard isolation
     Problems: low utilization, high service cost
   - 1.2. Specialized Hardware
     Problems: high cost, non-commodity

2. **Access to high-resolution timers**
   - Reduce resolution, add noise
   - Problems: Loss of feature or high overhead

3. **Vulnerabilities in CPU scheduler**
   - Managing preemptions – *soft isolation*

Scheduler is *exploited* in side-channels →
NO prior research on secure scheduler designs!
Soft Isolation Mechanisms

Goals:
1. Reduce risk of sharing
2. Monotonically improve security
3. Low performance overhead

Challenges:
- Unintuitive impacts of scheduler changes
- No standard benchmarks
- No security evaluation methods
Prime-Probe Side-channel Attack

- Core: VM VM VM VM
- Time: Prime Probe
- Preemption interval
- Shorter the preemption interval \(\rightarrow\) more (or any) information leakage bandwidth

- cache sets
- cache ways
Background: Xen CPU Scheduler

Throughput-oriented: Benefits from longer scheduler timeslices

Batch VM

Interactive VM

State-of-art CPU schedulers

Latency-oriented: Benefits from quick wakeups, BOOST priority

Prime-probe attacker: Abuses BOOST priority, using interrupts (IPIs).

Malicious VM
Soft-Isolation: Minimum Runtime Guarantee

Under Zhang et al. attack setting:

Core: VM VM VM VM

Time

IPI (boosted) < 10µs

Under Minimum RunTime (MRT) guarantee:

Core: VM VM VM VM

Time

IPI (boosted)

Min. runtime (scheduler parameter)

Introduced in Xen and Linux for performance improvement for batch VMs

What about security properties?
Evaluation of MRT

1. Does MRT make existing side-channels harder?

2. What is the scope of security against side-channels for all victims?

3. How much performance overhead for latency-sensitive applications with MRT?
**Experiment Setting**

**Machine Configuration:**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Intel Xeon E5645, 2.4GHz, 6 cores, single package</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Hierarchy</strong></td>
<td>Private 32KB L1 (I- and D-Cache), 256KB unified L2, 12MB shared L3 &amp; 16GB DDR3 RAM.</td>
</tr>
</tbody>
</table>

**Xen Configuration:**

<table>
<thead>
<tr>
<th>Xen Version</th>
<th>4.2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduler</strong></td>
<td>Credit Scheduler 1</td>
</tr>
<tr>
<td><strong>Configuration (Non-work conserving)</strong></td>
<td>40% cap on DomU VCPUs with equal weight</td>
</tr>
<tr>
<td># VMs</td>
<td>6</td>
</tr>
<tr>
<td># VCPUs per VM</td>
<td>2</td>
</tr>
</tbody>
</table>

**Similar to setting used by Zhang et al.**
Prime-Probe Timing Profile

A Sample Side-channel Victim

Victim Pseudo Code

```plaintext
if subset(secret)= X
  then
    for( sometime )
    do
      instr. in green
    endfor
  fi
if subset(secret)= Y
  then
    for( sometime )
    do
      instr. in blue
    endfor
  fi
```

For simplicity:
secret = XYXY...
Prime-Probe Timing Profile

Sample probe (time series)

I-cache set number

Cache Timing per iCache set probe (0 to 200 cycle range)

Idle Victim VM

Model Victim VM Under Zero-MRT
Prime-Probe Timing Profile

A simple scheduler mechanism → known attacks are harder
Elgamal Victim: Information Leakage

Elgamal Side-channel rely on consecutive redundant observations for noise-reduction
Security Limitations of MRT

1. Slower victims could still leak!

2. Only applicable to sub-class of side-channels, and to virtualized setting,

3. Interactive VMs that voluntarily relinquish the CPU are still vulnerable!

Modular Exponentiation Algorithm:

\[
\text{SQUAREMULT}(x, e, N):
\text{Let } e_n, ..., e_1 \text{ be the bits of } e
\]

\[
y \leftarrow 1
\]

\[
\text{for } i = n \text{ down to } 1 \text{ do}
\]

\[
y \leftarrow \text{SQUARE}(y)
\]

\[
y \leftarrow \text{MODREDUCE}(y, N)
\]

\[
\text{if } e_i = 1 \text{ then}
\]

\[
y \leftarrow \text{MULT}(y, x)
\]

\[
y \leftarrow \text{MODREDUCE}(y, N)
\]

\end if

\end for

\text{return } y

Per-core Shared State-Cleansing
Performance Evaluation

1. What is the overhead of turning on MRT?
   – A 0.3% improvement for batch workloads
   – On average 4% and at worst 7% overhead on 95th percentile latency

2. What is the overhead of cleansing on latency sensitive real-world applications?
   – adds a overhead of 10µs for latency sensitive workloads,
   – At worst a 80-100µs on 95th percentile latency
Conclusion

• Current state-of-the-art CPU schedulers do not account for malicious users,
• First-of-its-kind security analysis of schedulers
• Introduce new design paradigm: soft-isolation

Future work

– Model preemption-driven side-channels and estimate theoretical strength of MRT mechanism
– MRT-like mechanism for other system-level shared resources.
A Simple, Secure Scheduler Design

Scheduler VM-classification

Batch VM (CPU hungry)

Involuntary Context-switches

All context-switches leak information through shared hardware state

Voluntary Context-switches

Interactive VM (I/O intensive)

State-Cleansing

Protection Mechanisms

MRT Mechanism

Secures *most* victims involved in *involuntary switches*

Secures all victims involved in *voluntary switches*
Related Work


Questions?

Contact: venkatv@cs.wisc.edu