Scheduler-based Defenses against Cross-VM Side-channels

Venkat(anathan) Varadarajan, Thomas Ristenpart, and Michael Swift
Public Clouds (EC2, Azure, Rackspace, …)

Benefits:
1. High resource utilization,
2. Low service cost

Multi-tenancy
Different customers’ virtual machines (VMs) share same server
Shared Resources and Isolation

System shared resources (LLC, memory, disk, n/w etc.)
Problem: Cache-based Side-channels

\[ \text{CRYPTO\_FUNCTION}(s): \]
\[ s \leftarrow \text{secret bit} \]
\[ \begin{align*}
&\text{if } (s = 0) \{ \\
&\quad \text{OPERATION\_A}
\} \\
&\text{if } (s = 1) \{ \\
&\quad \text{OPERATION\_B}
\}
\end{align*} \]

*Zhang, Juels, Reiter, Ristenpart, “Cross-VM Side-channels …”, CCS’12
Requirements for Successful Side-channel VM

**Core:**
- Prime: A
- Probe: V
- A

Time

**Close cache set:**

**Quick preemption:**

**High-precision timer:**

**Secret:**

\[ \text{PROTOCOL}\_\text{FUNCTION}(s): s \leftarrow \text{secret bit} \]

\[ \text{if}\ (s = 0) \{ \]
  \[ \text{OPERATION}_A \]
\[ \}

\[ \text{if}\ (s = 1) \{ \]
  \[ \text{OPERATION}_B \]
\[ \}

**Attacker Timing Profile:**

**Extract secret information:**

*Zhang, Juels, Reiter, Ristenpart, “Cross-VM Side-channels …”, CCS’12*
Defenses against Side-channels

1. Sharing
   - Resource Partitioning [NoHype’10]
   - Specialized Hardware [RPcache’07]
   - Software-based partitioning [StealthMem’12]

2. Access to high-resolution timers
   - Reduce resolution [TimeWarp’12]
   - Removing timing channel [StopWatch’13]

No countermeasures deployed by providers!

3. Quick cross-VM preemptions
   - No prior work!
Our Solution: Soft Isolation

Allow sharing but limit frequency of dangerous VM interactions

Goals:
1. Secure: Controlled information leakage
2. Commodity: Easy to adopt
3. Efficient: Allow sharing, low overhead

... with simple changes to Hypervisor’s CPU scheduler
1. **Background:** Quick Preemptions & Schedulers

2. **Soft-Isolation:** Scheduler-based defense

3. **Evaluation:** Security and Performance
Requirement for Quick Preemptions

Core: V A V

Prime

Probe

Preemption Interval

Rate of preemption > Rate of event to measure

.crypto_function(s):
s ← secret bit

if (s = 0) {
    operation_A
}

if (s = 1) {
    operation_B
}

Next subsequent code/task execution ... (or noise)
Why do schedulers allow quick preemptions?

**Prime-probe attacker:** Abuses BOOST priority, using interrupts.

**Latency-oriented:** Benefits from *quick* wakeups, BOOST priority

**Throughput-oriented:** Benefits from longer scheduler timeslices

**State-of-art CPU schedulers**

**Batch VMs**

**Interactive VMs**

**Core:**

- V A V A

**< 10µs**

**Time**
Soft-Isolation: Ratelimit Preemptions

Available in Xen (and KVM)
- `ratelimit_us` (and `sched_min_granularity_ns`)
- Reduces VM-switches $\rightarrow$ Boosts batch-workload’s performance

Minimum RunTime (MRT) guarantee $\rightarrow$ soft-isolation
MRT Guarantee and Open Questions

1. Can MRT defend against Cross-VM Side-channels? *(security evaluation)*

2. Trade-off between security and performance? *(performance overhead)*
Experimental Methodology

Two VMs:
1. Attacker
2. Victim

Setting similar to public clouds (e.g. EC2)

<table>
<thead>
<tr>
<th>Xen Configuration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Xen Version</td>
<td>4.2.1</td>
</tr>
<tr>
<td>Scheduler</td>
<td>Credit Scheduler 1</td>
</tr>
<tr>
<td>Configuration (Non-work conserving)</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>

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

Prime-Probe Timing Profile

Sample probe (time series)

i-cache access timing

Alternating usage pattern

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

Idle Victim VM

Simple Victim VM Under Zero-MRT
Security Evaluation:
Prime-Probe Timing Profile

- Side-channel not discernible
- Alternating usage pattern

Simple Victim VM Under 1ms MRT

Simple Victim VM Under Zero-MRT
Security Evaluation:
ElGamal Victim

ElGamal Side-channel require multiple preemptions within single iteration for noise-reduction [Zhang et al’12]

Minimum number of iterations per preemption

<table>
<thead>
<tr>
<th>Xen MRT (ms)</th>
<th>0</th>
<th>0.1</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg: 0.096 ops</td>
<td>0</td>
<td>4</td>
<td>32</td>
<td>68</td>
<td>155</td>
<td>386</td>
<td>728</td>
</tr>
</tbody>
</table>

SQUAREMULT(x, e, N):
Let $e_n$, ..., $e_1$ be the bits of $e$
$y \leftarrow 1$
for $i = n$ down to 1 do
    $y \leftarrow$ SQUARE($y$)
    $y \leftarrow$ MODREDUCE($y$, N)
    if $e_i = 1$ then
        $y \leftarrow$ MULT($y$, x)
        $y \leftarrow$ MODREDUCE($y$, N)
    end if
end for
return $y$
MRT Guarantee and Open Questions

1. Can MRT defend against Cross-VM Side-channels? (security evaluation)

2. Trade-off between security and performance? (performance overhead)
Performance Evaluation: Overall System Performance

Measured workload:
1. *Interactive* ➔ memcached, cassandra, etc. and
2. *Batch* ➔ graph500, specJBB, etc.

Competing workloads:
microbenchmarks ➔ highly cache-thrashing + (interactive or batch)
Performance Evaluation: Overall System Performance

- **Overall System Performance**
- **Normalized to Zero-MRT**
- **All-Batch**
- **All-Interactive**
- **Interactive-Batch**
- **Idle**

- **Avg. 95th Percentile Latency** (interactive workloads)
- **Avg. Runtime** (batch workloads)

< 7% overhead

At 5ms MRT
More details in the paper …

• Per-core State-Cleansing
  – Interactive VMs may still leak information
  – MRT + State-cleansing incur low overhead

• Detailed Performance and Security Analysis
  – 20+ graphs in the paper

It is cheap and easy to deploy!
Conclusion

5ms MRT + selective state-cleansing
– known attacks no longer work
– negligible overhead
– easy to adopt

Introduce new scheduler principle
– soft-isolation = allow sharing + limit dangerous cross-VM interactions

https://bitbucket.org/vvaradarajan/robsched

contact: venkatv@cs.wisc.edu