Cloud security

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu
Announcements

• Take-home final versus in-class
• Homework 3 problem 4
Cloud computing

NIST: Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
A simplified model of public cloud computing

Users run Virtual Machines (VMs) on cloud provider’s infrastructure

Virtual Machine Manager (VMM) manages physical server resources for VMs

To the VM should look like dedicated server

Multitenancy (users share physical resources)
Trust models in public cloud computing

Users must trust third-party provider to

- not spy on running VMs / data
- secure infrastructure from external attackers
- secure infrastructure from internal attackers
Trust models in public cloud computing

Users must trust third-party provider to:

- Not spy on running VMs / data
- Secure infrastructure from external attackers
- Secure infrastructure from internal attackers

Threats due to sharing of physical infrastructure?

- Your business competitor
- Script kiddies
- Criminals
  ...

User A

Bad guy
A new threat model:

Attacker identifies one or more victims VMs in cloud

1) Achieve advantageous placement via launching of VM instances

2) Launch attacks using physical proximity

   Exploit VMM vulnerability            DoS            Side-channel attack
1 or more targets in the cloud and we want to attack them from same physical host

Launch lots of instances (over time), with each attempting an attack

Can attackers do better?
Outline of a more damaging approach:

1) Cloud cartography
   map internal infrastructure of cloud
   map used to locate targets in cloud

2) Checking for co-residence
   check that VM is on same server as target
   - network-based co-residence checks
   - efficacy confirmed by covert channels

3) Achieving co-residence
   brute forcing placement
   instance flooding after target launches

4) Location-based attacks
   side-channels, DoS, escape-from-VM

Placement vulnerability: attackers can knowingly achieve co-residence with target
Case study with Amazon’s EC2

1) given no insider information
2) restricted by (the spirit of) Amazon’s acceptable use policy (AUP)
   (using only Amazon’s customer APIs and very restricted network probing)

We were able to:

- Pick target(s)
- Choose launch parameters for malicious VMs
- Each VM checks for co-residence
- Cross-VM side channel attacks to spy on victim’s computational load
- Frequently achieve advantageous placement
Some info about EC2 service (at time of study)

Linux-based VMs available
Uses Xen-based VM manager

Launched parameters

3 “availability zones” (Zone 1, Zone 2, Zone 3)
5 instance types (various combinations of virtualized resources)

<table>
<thead>
<tr>
<th>Type</th>
<th>gigs of RAM</th>
<th>EC2 Compute Units (ECU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small (default)</td>
<td>1.7</td>
<td>1</td>
</tr>
<tr>
<td>m1.large</td>
<td>7.5</td>
<td>4</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>c1.medium</td>
<td>1.7</td>
<td>5</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>7</td>
<td>20</td>
</tr>
</tbody>
</table>

1 ECU = 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

Limit of 20 instances at a time per account.
Essentially unlimited accounts with credit card.
Our experiments indicate that internal IPs are **statically assigned** to physical servers.

Co-residence checking via Dom0: only hop on traceroute to co-resident target.
Cloud cartography

Pick target(s) ➔ Choose launch parameters for malicious VMs

3 “availability zones”
(Zone 1, Zone 2, Zone 3)

5 instance types
(m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)

User account
Cloud cartography

Pick target(s)

Choose launch parameters for malicious VMs

3 “availability zones”
(Zone 1, Zone 2, Zone 3)

5 instance types
(m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)

launch parameters

User account

Launch parameters

User account

Account B Account A

Internal IP address

c1.medium ☐ c1.xlarge ☓ m1.large ☐ m1.small ☐ m1.xlarge ☓
Associate to each /24 an estimate of **Availability zone** and **Instance Type**

External IP → **DNS** → Internal IP → /24 → **Availability zone**
**Instance Type**

Mapping 6,577 public HTTP servers running on EC2 (Fall 2008)
Achieving co-residence

“Brute-forcing” co-residence

Attacker launches many VMs over a relatively long period of time in target’s zone and of target type

Experiment:

1,686 public HTTP servers as stand-in “targets” running m1.small and in Zone 3 (via our map)

1,785 “attacker” instances launched over 18 days

Each checked co-residence against all targets using Dom0 IP

Results:

78 unique Dom0 IPs

141 / 1,686 (8.4%) had attacker co-resident

Sequential placement locality lowers success

Lower bound on true success rate
Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

Launch many instances in parallel near time of target launch
Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

Launch many instances in parallel near time of target launch

Experiment:

Repeat for 10 trials:

1) Launch 1 target VM (Account A)

2) 5 minutes later, launch 20 “attack” VMs (alternate using Account B or C)

3) Determine if any co-resident with target using Dom0 IP

4 / 10 trials succeeded
Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

How long is parallel placement locality good for?

Experiment:

40 “target” VMs (across two accounts)
20 “attack” VMs launched hourly
Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

What about commercial accounts?

Free demos of Internet appliances powered by EC2

2 attempts

1\textsuperscript{st} – coresident w/ 40 VMs

2\textsuperscript{nd} – 2 VMs coresident w/ 40 launched

Several attempts

1\textsuperscript{st} – coresident w/ 40 VMs

Subsequent attempts failed
Checking for co-residence

How do we know Dom0 IP is valid co-residence check?

Use simple covert channel as ground truth:

- **Sender** transmits ‘1’ by franticly reading random locations
- **Sender** transmits ‘0’ by doing nothing
- **Receiver** times reading of a fixed location

Covert channels require control of both VMs: we use only to **verify** network-based co-residence check
Checking for co-residence

Experiment

Repeat 3 times:
1) 20 m1.small Account A
2) 20 m1.small Account B
3) All pairs w/ matching Dom0 → send 5-bit message across HD covert channel

Ended up with 31 pairs of co-resident instances as indicated by Dom0 IPs

Result: a correctly-received message sent for every pair of instances

During experiment also performed pings to:
* 2 control instances in each zone
* co-resident VM

| Zone 1 Control 1 | 1.164 |
| Zone 1 Control 2 | 1.027 |
| Zone 2 Control 1 | 1.113 |
| Zone 2 Control 2 | 1.187 |
| Zone 3 Control 1 | 0.550 |
| Zone 3 Control 2 | 0.436 |
| Co-resident VM   | 0.242 |
So far we were able to:

- Pick target(s)
- Choose launch parameters for malicious VMs
- Each VM checks for co-residence
- Frequently achieve advantageous placement

This shouldn’t matter if VMM provides good isolation!
Violating isolation

- Hard drive covert channel used to validate Dom0 co-residence check already violated isolation
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability
- Escape-from-VM vulnerabilities
- Side-channel attacks
Cross-VM side channels using CPU cache contention

1) Read in a large array (fill CPU cache with attacker data)
2) Busy loop (allow victim to run)
3) Measure time to read large array (the load measurement)
Cache-based cross-VM load measurement on EC2

Repeated HTTP get requests

Running Apache server

Performs cache load measurements

3 pairs of instances, 2 pairs co-resident and 1 not
100 cache load measurements during HTTP gets (1024 byte page) and with no HTTP gets
Cache-based load measurement of traffic rates on EC2

3 trials with 1 pair of co-resident instances:
1000 cache load measurements during
0, 50, 100, or 200 HTTP gets (3 Mbyte page) per minute for ~1.5 mins
More on cache-based physical channels

Prime+Trigger+Probe combined with differential encoding technique gives high bandwidth cross-VM covert channel on EC2
See [Xu et al., “An Exploration of L2 Cache Covert Channels in Virtualized Environments”, CCSW 2011]

Keystroke timing in experimental testbed similar to EC2 m1.small instances

AMD Opterons

CPU 1
Core 1
Core 2

CPU 2
Core 1
Core 2
More on cache-based physical channels

Prime+Trigger+Probe combined with differential encoding technique gives high bandwidth cross-VM covert channel on EC2

See [Xu et al., “An Exploration of L2 Cache Covert Channels in Virtualized Environments”, CCSW 2011]

Keystroke timing in experimental testbed similar to EC2 m1.small instances

AMD Opterons

CPU 1

Core 1

Core 2

CPU 2

Core 1

Core 2
More on cache-based physical channels

Prime+Trigger+Probe combined with differential encoding technique gives high bandwidth cross-VM covert channel on EC2
See [Xu et al., “An Exploration of L2 Cache Covert Channels in Virtualized Environments”, CCSW 2011]

Keystroke timing in experimental testbed similar to EC2 m1.small instances
More on cache-based physical channels

Prime+Trigger+Probe combined with differential encoding technique gives high bandwidth cross-VM covert channel on EC2

See [Xu et al., “An Exploration of L2 Cache Covert Channels in Virtualized Environments”, CCSW 2011]

Keystroke timing in experimental testbed similar to EC2 m1.small instances

We show that cache-load measurements enable cross-VM keystroke detection

Keystroke timing of this form might be sufficient for the password recovery attacks of [Song, Wagner, Tian 01]
What can cloud providers do?

1) **Cloud cartography**

2) **Checking for co-residence**

3) **Achieving co-residence**
   - Isolate each user’s view of internal address space
   - Random Internal IP assignment
   - Hardware or software countermeasures to stop leakage
   - Hide Dom0 from traceroutes
   - Allow users to opt out of multitenancy

4) **Side-channel information leakage**

Possible counter-measures:
- Random Internal IP assignment
- Isolate each user’s view of internal address space
- Hide Dom0 from traceroutes
- Allow users to opt out of multitenancy
- Hardware or software countermeasures to stop leakage

Amazon provides dedicated instances now. They cost a lot more.
Untrusted provider

• A lot of work aimed at untrustworthy provider
• Attestation of cloud:
  – Homealone: use L2 cache side-channels to detect presence of foreign VM
  – RAFT: Remote Assessment of Fault Tolerance to infer if data stored in redundant fashion
  – Keep data private: searchable or fully-homomorphic encryption