HEY, YOU, GET OFF OF MY CLOUD

EXPLORING INFORMATION LEAKAGE IN THIRD-PARTY COMPUTE CLOUDS

Adam Bates, CIS607 Security in Systems, Storage and the Cloud
PREVIOUSLY ON CIS607 SECURITY SEMINAR

• Third-party cloud computing represents the promise of outsourced computation.
• It allows customers to purchase just the capacity they require, just when they require it.
• Cloud providers are able to maximize utilization of their capital investments by multiplexing many customer VMs across a shared physical infrastructure.
• It is a given that we need to be able to trust cloud providers to respect our private data...
... BUT CAN WE TRUST OTHER CUSTOMERS?

• We already know that 3rd Party cloud providers make their $$$ by multiplexing the machines in their monstrously large datacenters.
• This paper considers the threats of multi-tenancy, multiplexing the virtual machines of disjoint customers upon the same physical hardware.
• Could a customer be assigned to the same physical server as their adversary?
• Could the adversary exploit co-residency to extract confidential information?
RISTENPART & CO.’S PLAN OF ATTACK

1. Use Amazon EC2 as a case study.
   - U.S. Region
   - Linux Kernel

2. Achieve **PLACEMENT** of their malicious VM on the same physical machine as that of a target customer.
   - Determine where in the cloud an instance is likely to be located.
   - Determine if two instances are co-residents.
   - Intentionally launch an instance to achieve co-residence with another user.

3. Proceed to **EXTRACT** information and/or perpetrate all kinds of assorted nastiness.
THREAT MODEL FOR THIS PAPER

• We trust the provider, its infrastructure and its employees.
• Adversaries are non-provider-affiliated malicious parties.
• Victims are running confidentiality-requiring services in the cloud.
• Everyone is a customer; both groups can all run and control many instances.
• We are not concerned with traditional threats and exploits here, even though they are alive and well in the cloud environment.
• 3rd Party Cloud Providers give attackers novel abilities, implicitly expanding the attack surface of the victim.
• Two kinds of attackers
  1. Casts a wide net in an attempt to attack somebody
  2. Focuses on attacking a particular victim service
THE EC2 SERVICE: RELEVANT FACTS

- The Xen hypervisor makes use of the Domain0 (Dom0) privileged VM. It routes packets for guest images and is the first hop in traceroutes for those images.
- When an instance is launched on EC2, it is assigned to a single physical machine within the EC2 network for its lifetime.
- Three Degrees of freedom in specifying the physical infrastructure upon which instances should run.
  1. Region
  2. Availability Zone
  3. Instance Type
The EC2 Service: Region/Zone/Type Hierarchy

I. Regions: One of two, either the United States or Europe.

II. Availability Zones: Each have separate power and network connectivity, provide independent failure modes.

III. Instance Types: A particular combination of computational power, memory and persistent storage available to the VM.

How do these options affect our physical address?
NETWORK PROBING METHODS

• The writers used **nmap**, **hping** and **wget** to determine liveness of EC2 instances:
  • **nmap** for TCP connect probes on ports 80 and 443
  • **hping** for TCP SYN traceroutes on ports 80 and 443
  • **wget** to retrieve limited amounts of web page data
• Internal probing is subject to Amazon’s acceptable use policy!
• Circumvented by using DNS resolution queries to discover the external/internal IP addresses of the target instance, then probing externally.
CLOUD CARTOGRAPHY

• Hypothesis: different availability zones (and possibly instance types) are likely to correspond to different internal IP address ranges.

• Since we already know that it’s possible to infer the internal IP address of an instance associated with a public IP through the EC2’s DNS service...

• If this hypothesis holds, an adversary can use a map of EC2 to determine the instance type and availability zone of their target, dramatically reducing the number of instances needed to achieve co-residence.
MAPPING AVAILABILITY ZONE

Plot of the internal IP addresses of 300 instances launched by a single account.

- This tells us that, not surprisingly, the Amazon EC2 internal address space is cleanly partitioned between availability zones.
- This stands to reason because availability zones exist to offer varied points of failure.
MAPPING INSTANCE TYPE & ACCOUNT

Plot of the internal IP addresses of Zone 3 instances launched by “Account A” and, 39 hours later, by “Account B.”

- No IP address was ever observed to be assigned to more than one instance type!
- This tells us that, independent of the Account launching the instance, EC2 maps specific IP ranges within a zone to each instance type.

55 of the Account B IPs were repeats of those assigned to instances for Account A.
OBVIOUS SOLUTION: STOP DOING THAT?

- Providers have plenty of incentive to prevent cloud cartography.
  - Hide infrastructure and usage data.
  - Protect against exploitation of placement vulnerabilities

- Still, they can’t just take the map away.
  - Administration would become more costly.
  - This would only slow down the technique for locating an instance – other approaches are already detailed in this paper!
DETERMINING CO-RESIDENCE

Network Based Checks

*Hypothesis: Instances are likely co-resident if they have*

1. Matching Dom0 IP address
2. Small packet round-trip times
3. Numerically close internal IP addresses (within 7)

- The correctness of these checks was verified with a cross-VM, hard-disk-covert channel.
- *Result: false positive rate of 0% for the Dom0 IP co-residence check.*
SOLUTION: OBFUSCATE CO-RESIDENCE?

• Providers could easily render these checks moot. MOOT!
  • Have Dom0 not respond to traceroutes.
  • Randomly assign internal IP addresses at instance launch.
  • Use Virtual LANs to isolate accounts.

• Unlike last time, this is a very good idea.
  • Network-based checks are an absurdly efficient method of determining co-residence.
  • Still, side channels can be used to verify co-residence without the use of network-based checks.
EXPLOITING PLACEMENT IN EC2

Towards Understanding Placement: Relevant Facts

• A single account running \( n \) instances will likely have placement on \( n \) separate machines.

• Sequentially launched instances from different accounts exhibit *placement locality*.

• *Load Balancing* in placement algorithm: Correlation between a machine’s instance density and the likelihood that a new instance will be assigned to that machine.
EXPLOITING PLACEMENT IN EC2

Brute Force Method

- Run numerous instances over a (relatively) long period of time and see how many targets one can achieve co-residence with.
- This naïve approach enjoyed 8.4% coverage against a target set of 1,686.
- *Sequential Locality* is what kept the rate from being higher. 90% of the hits occurred in the first third of the probe launches.
EXPLOITING PLACEMENT IN EC2

Abusing Placement Locality

• If an attacker can launch instances relatively soon after the launch of a victim’s, they can engage in instance flooding in the appropriate zone/type.
• Is this a reasonable scenario?
• Yes! The point of cloud computing is to only run servers when needed.
  • Network probe the server’s state until it disappears, then instance flood as soon as it reappears.
  • Trigger the launch of a new instance in an auto-scaling system.
EXPLOITING PLACEMENT IN EC2

The effects of zone, account, and time of day

<table>
<thead>
<tr>
<th>Zone</th>
<th># victims v</th>
<th># probes p</th>
<th>coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone 1</td>
<td>1</td>
<td>20</td>
<td>1/1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>5/10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>7/20</td>
</tr>
<tr>
<td>Zone 2</td>
<td>1</td>
<td>20</td>
<td>0/1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>18</td>
<td>3/10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>19</td>
<td>8/20</td>
</tr>
<tr>
<td>Zone 3</td>
<td>1</td>
<td>20</td>
<td>1/1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>2/10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>8/20</td>
</tr>
</tbody>
</table>

Coverage of victim instances. Five minutes after victim launches a number of probes, the attacker launches 20 probes.

Focus on the 1-to-20 scenario.
- Two different accounts
- Three different times of day.
- 5 attempts per account/time combo

Given a five minute delay, success rate is 40%.
EXPLOITING PLACEMENT IN EC2

The effect of increased time lag

Hourly attack rounds against 40 victim instances. The success rate of each round stays essentially the same.

Sequential Placement Locality at work – the number of new machines that probes were placed on levels off rapidly.

The window of opportunity is quite large.
EXPLOITING PLACEMENT IN EC2

Targeting Commercial Instances

The writers attempt to perform targeted instance flooding against instances run by other user’s accounts.

• RightScale: co-residence achieved in two rounds of 20 instance probes from one account.
• RightScale: three-way co-residency achieved in two rounds of 38 instance probes from two different accounts.
• rPath: Failure! Target was likely placed on a full or dense machine and was therefore unassailable.
SOLUTION: OFFLOAD CHOICE TO USERS

Patching Placement Vulnerabilities

• Inhibiting cartography or co-residence checks will only slow down a dedicated attacker.
• As demonstrated with the rPath scenario, an instance is unassailable if it is on a protected machine.
• Let users request placement of their VMs on machines that can only be populated by VMs from their (or other trusted) accounts.
• In exchange, users can pay the opportunity cost of under-utilization.
CROSS-VM INFORMATION LEAKAGE

On Stealing Cryptographic Keys

Theoretically possible, but somewhat difficult in the cloud/EC2 environment.

• Core migration
• Coarse scheduling algorithms
• Double indirection of memory addresses
• Unknown load from other instances
• No hyperthreading
CROSS-VM INFORMATION LEAKAGE

Other channels; denial of service

• Any physical machine resources multiplexed between the attacker and the target forms a potentially useful channel.

• These resources can also be used to mount cross-VM performance degradation.

• These resources can also be used to mount cross-VM denial-of-service attacks.

• We’ve learned previously that such attacks, while costly and annoying, are inherently less effective in a scalable environment.
CROSS-VM INFORMATION LEAKAGE

Measuring Cache Usage

- An attacking instance can measure CPU utilization caches on its physical volume.
- These measurements can be used to estimate current load of the machine.
- A high load indicates activity on co-resident instances.
- The writers utilize a “Prime+Trigger+Probe” technique to measure cache activity in a time-shared virtual machine environment.
CROSS-VM INFORMATION LEAKAGE

Prime+Trigger+Probe

1. Prime: Read from a contiguous buffer of b bytes at s-byte offsets in order to ensure the buffer is cached.
2. Busy-loop until the CPU’s cycle counter jumps by a large value, meaning that the probing VM was preempted by Xen in favor of another VM.
3. Measure the time it takes to again read from the buffer at s-byte offsets.

The time of (3)’s read, the load sample, will be strongly correlated with the use of the cache by other instances running on the machine.
CROSS-VM INFORMATION LEAKAGE

Cache-based covert channel

• This technique can be used to create a very effective covert channel!

• This can be relevant in situations where cross-VM communication over the network is supposedly forbidden.

• Partition the cache into even sets and odd sets.

• Read even for 0, odd for 1.

• Signal is more noise resistant because of this use of differential encoding – the signal is carried in the difference between the two sets.
CROSS-VM INFORMATION LEAKAGE

Load-based Co-Residence Detection

Three trials of 100 Prime+Trigger+Probe cache timing measurements. Instances were co-residences in Trial 1 and Trial 2, not in Trial 3.

• Same technique, different application.
• Confirm co-residency by measuring load while executing GET requests on an instance running a web server.
Mean cache load measurements while different rates of web requests were issued to an instance running a public server.

- Same technique, different application.
- Load measurement provides a method for estimating the number of visitors to a co-resident web server.
- This information may not be public and could be valuable to a corporate competitor.
CROSS-VM INFORMATION LEAKAGE

Keystroke Timing Attack

• Could a very patient attacker use load measurement to detect keystrokes and perform password recover?

• A few of the challenges:
  • Requires an otherwise idle machine
  • Live EC2 performs core migration, attacking VM will only share a core about 25% of the time.

• For what it is worth, the writers ARE able to detect keystrokes with a 95% success rate in a controlled, local testbed running a Xen hypervisor.

... yeah, ok.
SOLUTION: INHIBIT SIDE-CHANNEL ATTACKS?

• Can we apply blinding techniques to minimize the information that can be leaked?
  • Previous work on cache side channel countermeasures are typically impractical or insufficient for fully mitigating risk.
  • Requires that *all* possible side-channels have been anticipated and disabled
CONCLUSIONS

1. Consider obfuscating internal structure and placement policy, if only to get on adversaries’ nerves.

2. Apply blinding techniques on discovered side-channel vulnerabilities.

3. The only complete solution is to expose the risk and placement decisions directly to users.
   - A user might insist on using physical machines only with other trusted VMs.
   - That user would bear the opportunity cost of under-utilization.
   - Large users would only incur minor penalties as a fraction of their total cost.
LEGAL, ETHICAL & CONTRACTUAL OBLIGATIONS

In Brief

- Network Probing = Controversial
- Computer Fraud and Abuse Act (CFAA) says that computer system “access” must be “authorized.”
- The writers only probe *publicly facing* ports.
- AWS Web Services Agreement:
  - Probing for *vulnerabilities* is prohibited.
  - Connecting to web servers is implicitly authorized.
- The writers were using their own accounts anyway.
Figure 7: A plot of the internal IP addresses of public EC2 servers.
THE END?

THANKS FOR YOUR TIME.