Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds

T. Ristenpart, H. Shacham, S. Savage *UC San Diego*
E. Tromer *MIT*

CPCS 722: Advanced Systems Seminar
Ewa Syta

Get Off of My Cloud

"Hey! You! Get off of my cloud
Don't hang around 'cause two's a crowd "

The Rolling Stones

Goals of the Presentation

- Define the problem and the motivation
- Define the threat model
- Outline the attack
- Discuss the feasibility and consequences of the attack
- Discuss the countermeasures

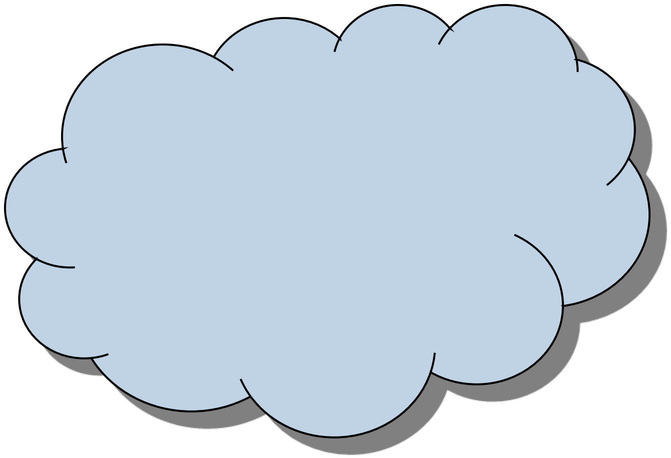
Not a Goal

- Discuss the tiniest details of the attack

… that's what the paper is for ☺

The Cloud

- What is the cloud?
  - The next infrastructure for hosting data and deploying software and services
  - Software/Platform/Infrastructure as a service*
  - Hot topic! Everyone moves to the cloud
- (some) Benefits
  - Cost savings
  - Scalability
  - Flexibility
- (new) Risks
  - Trust & Dependence
  - Security (multi-tenancy)



**The NIST Definition of Cloud Computing* http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Multi-Tenancy

- Multiplexing VMs of disjoint customers upon the same physical hardware
  - Your instance is placed on the same server with other customers
  - The problem: you wouldn't invite an attacker into your own physical server, would you?
- New risks
  - Side channels exploitation
  - Vulnerable VM isolation mechanisms
  - Lack of control who you're sharing server space with

The Attack

- Motivation
  - Explore the threats of multi-tenancy in cloud computing
- Real cloud service provider used (Amazon EC2)
- Two main steps
  1. *Placement* – place a malicious VM on the same physical machine as that of the victim
  2. *Extraction* – extract confidential information via a side channel attack

Threat Model

- Provider and infrastructure are trusted
- Focus on new, cloud-related capabilities of the attacker and expanding the attack surface
  - Do not consider attacks that rely on subverting administrator functions
  - Do not exploit vulnerabilities of the virtual machine monitor and/or other software
- Attacker
  - Not affiliated with the provider
  - Can run many instances
  - His instances might be placed on the same physical hardware as potential victims
  - Might manipulate shared physical resources

What Can Be Achieved?

1. Can one determine where in the cloud infrastructure an instance is located?
2. Can one easily determine if two instances are co-resident on the same physical machine?
3. Can an adversary launch instances that will be co-resident with other user's instances?
4. Can an adversary exploit cross-VM information leakage once co-resident?

Amazon Elastic Computer Cloud*

- Scalable, pay-as-you-go compute capacity in the cloud
- Customers can run different operating systems within a virtual machine
- Different computing options (instances) available
  - Standard, Micro, High-Memory, High-CPU, Cluster Compute & Cluster GPU
- Different regions available
  - US, EU, Asia
- Regions split into availability zones
  - In US: East (Virginia), West (Oregon), West (Northern California)

*Amazon EC2 website http://aws.amazon.com/ec2/

EC2 Instances

- m1.small m1.large m1.xlarge c1.medium c1.xlarge*

Small Instance – default*

1.7 GB memory
1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)
160 GB instance storage
32-bit platform
I/O Performance: Moderate
API name: m1.small

Large Instance

7.5 GB memory
4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)
850 GB instance storage
64-bit platform
I/O Performance: High
API name: m1.large

Extra Large Instance

15 GB memory
8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Unit
1,690 GB instance storage
64-bit platform
I/O Performance: High
API name: m1.xlarge

High-CPU Extra Large Instance

7 GB of memory
20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each)
1690 GB of instance storage
64-bit platform
I/O Performance: High
API name: c1.xlarge

High-CPU Medium Instance

1.7 GB of memory
5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)
350 GB of instance storage
32-bit platform
I/O Performance: Moderate
API name: c1.medium

*More instances available now. We will focus on the ones available at the time of the attack.

EC2 Highlights

- Xen (Virtual Machine Monitor)
- Domain0 (Dom0) privileged virtual machine configured to route packets for its guest images and reports itself as a hop in traceroutes
- Customers randomly assigned to physical machines based on their instance and region choices
- When an instance is launched, it is assigned to a single physical machine for its lifetime
- Each instance assigned internal and external IP address and domain name
- Amazon balances load across machines
- Instances are assigned to servers in a predictable manner (strong placement locality)

Q1: Cloud Cartography

- Instance placing is not disclosed by Amazon but needed in order to carry out the attack
  - Map the EC2 service to understand where instances are placed
- Hypothesis
  - Different availability zones and instance types correspond to different IP address ranges
- Evaluation
  - Survey public servers on EC2map internal addresses to public addresses
  - Review addresses assigned to a large number of launched instances

Network Probing

- Identify public services hosted on EC2 and verify co-residence
- Tools (free and readily available) used to probe ports (80 and 443)
  - *nmap* – perform *TCP connect* probes (attempt to complete a 3-way hand-shake between a source and target)
  - *hping* – perform *TCP SYN* tracerouts, which iteratively sends *TCP SYN* packets with increasing *TTL*s
  - *wget* – used to retrieve web pages

Survey Public Servers on EC2

- Create a set of public EC2-based web services
- Use *WHOIS* to identify distinct IP address prefixes associated with EC2 (/17, /18, /19)
- Use external probes to find responsive IPs
  - *TCP connect* probe on 80 (11315 responsive IPs)
  - Follow up with *wget* on 80 (9558 responsive IPs)
  - *TCP scan* on 443 (8375 responsive IPs)
- Translate all responsive public IPs into internal IPs using the EC2's internal DNS
  - 14054 unique internal IPs

Instance Placement Parameters

- EC2's internal address space is cleanly partitioned between availability zones
  - 20 instances launched for each of the 15 availability zone/instance type pairs (Fig 1)
- How about the instances? (Fig 2)
  - 10 instances launched (20 of each type)
  - Two accounts used (A & B)
  - B's instances launched 39 hours after terminating A's

Results

Fig 1. A plot of 300 internal IP addresses assigned to instances launched during the initial mapping experiment

The figure shows that different availability zones correspond to different IP address ranges

Results

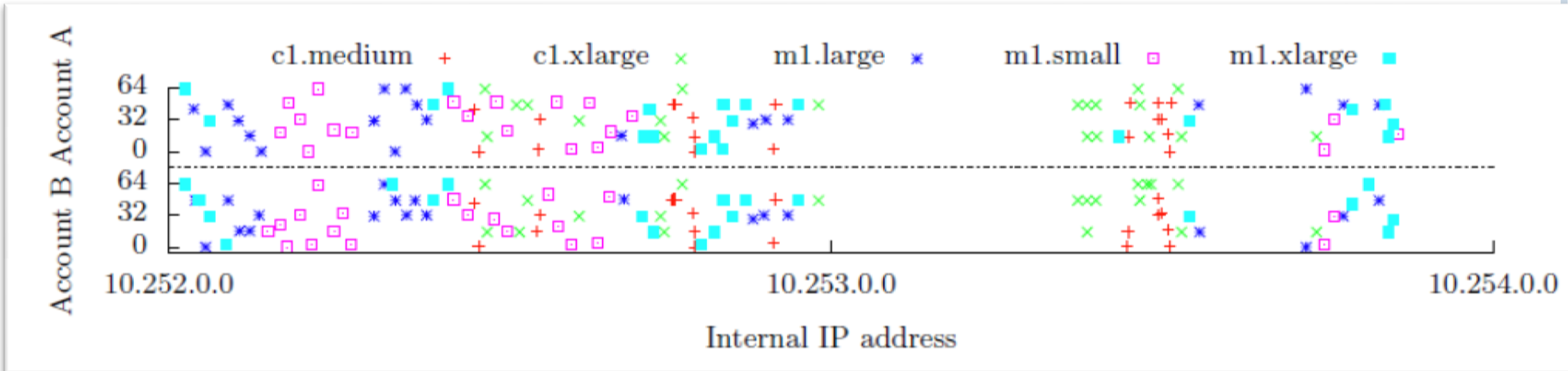


Fig 2. A plot of the internal IP addresses of instances launched in Zone 3 by Account A and, 39 hours later, by Account B

55 of the Account B IPs were repeats of those assigned to instances for Account A

Some instance types correspond to same IP address ranges

Q2: Determining Co-Residence

- Use the set of identified targets
- Check for co-residence: likely to be co-resident if
  1. Matching Dom0 IP address
  2. Small packet round-trip times, or
  3. Numerically close internal IP address (within 7)
- Verify co-residence
  - Hard disk based covert channel*
    - To send a 1, sender reads from random locations on a shared volume, to send a 0 sender does nothing
    - Receiver times reading from a fixed location on the disk, longer read times mean a 1 is set, shorter a 0
  - Perform the above verification for the 3 co-residency tests

*Unauthorized communication channel

Experiment

- 3 m1.small EC2 accounts: control, victim, probe
  - 2 control instances in 3 availability zones, 20 victim and 20 probe instances in Zone 3
- Determine Dom0 address of each instance.
- For each ordered pair (A,B) of 40 instances perform co-residence checks
- After 3 independent trials 31 (potentially) co-resident pairs have been identified
- 5-bit message from A to B over the covert channel was successfully sent for 60 out of 62 ordered pairs

Effective Co-Residence Check

- The previous approach works and will be used for checking co-residence with target instances
  - Compare internal IP addresses to see if they are close
  - If yes, perform a *TCP SYN* traceroute to an open port on the target and see if there is only a single hop (Dom0 IP)
- Very "quiet" check (little communication with the victim)

Q3: Causing Co-Residence

- Can we place an attacker's instance on the same physical machine as a particular victim?
- Two strategies to achieve "good" coverage (co-residence with a good fraction of the target set)
  - Brute-force placement (launch many instances over a relatively long period of time)
    - Of 1686 target victims co-residence achieved with 141 victim servers (8.4% coverage)
  - Target recently launched instances (take advantage of the tendency for EC2 to assign fresh instances to the same small set of machines)

Leverage Placement Locality

- Launch lots of instances right after the launch of victim's instance
  - Attacker may trigger a new instance launch by overloading the victim with requests (auto-scaling)
- Experiment
  - Single victim instance is launched
  - Attacker launches 20 instances within 5 minutes
  - Perform co-residence check
- 40% of the time the attacker launching just 20 probes achieves co-residence against a specific target instance

Targeting Commercial Instances

- Accounts created with RightScale and rPath
  - Both run on EC2
- Apply the attack strategy
  - Map the fresh instance and flood
- RightScale
  - Two rounds of 20 instances launched using one account before achieving co-residence
  - Two rounds of 38 instances launched using two accounts achieved three-way co-residency
- rPath
  - Co-residence achieved using 40 instances
  - Another attempt of flooding failed
    - rPath instance might have been placed on a full machine

Q4: Exploiting Co-Residence

- Cross-VM attacks can allow for information leakage
- How can we exploit the shared infrastructure?
  - Gain information about the resource usage of other instances
  - Create and use covert channels to intentionally leak information from one instance to another
    - Crypto keys?
- Extraction of crypto keys
  - Potentially possible but still very difficult
  - The attacks shown are coarse
  - Very serious consequences if achieved

Exploiting Co-Residence

- Measuring cache usage
  - Time-shared cache allows an attacker to measure when other instances are experiencing computational load
  - Cache-based covert channel:
    - Sender idles to transmit a 0 and frantically accesses memory to transmit a 1
    - Improved *Prime+Trigger+Probe* technique yields a bandwidth of approximately 0.2 bps
- Load-based co-residence check
  - Co-residence check can be done without network-based technique
  - Use a priori knowledge about load variation
  - Induce computational load (lots of HTTP requests) and observe

Results

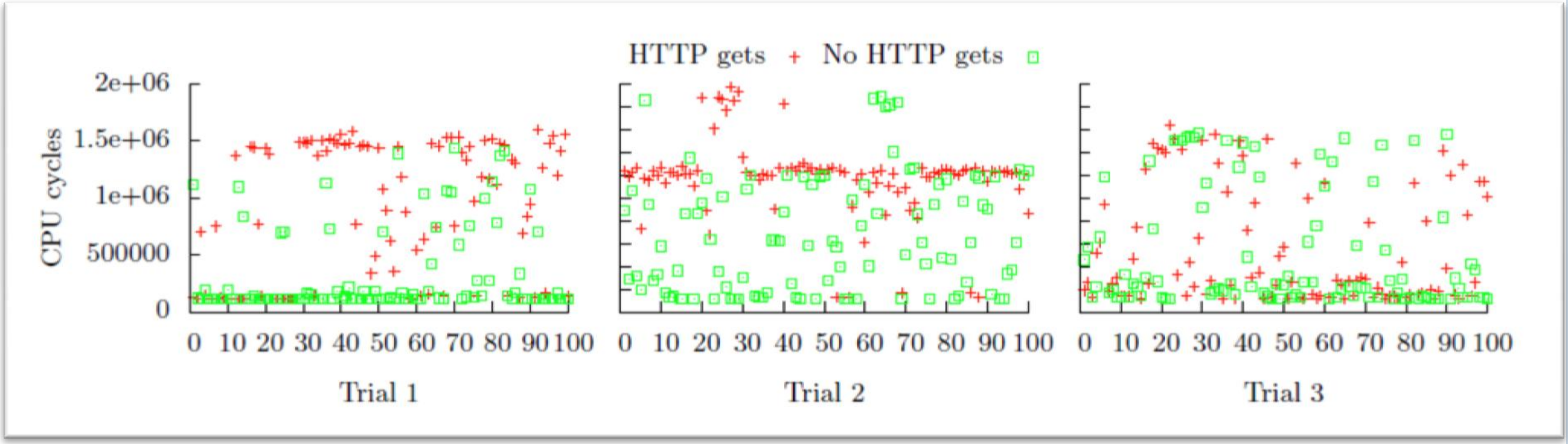


Fig 4. Results of executing 100 *Prime+Trigger+Probe* cache timing measurements for three pairs of m1.small instances, both when concurrently making HTTP get requests and when not

Instances in Trial 1 and Trial 2 were co-resident on distinct physical machines; instances in Trial 3 were not co-resident

Exploiting Co-Residence

- Estimating traffic rates
  - Load measurement might provide a method for estimating the number of visitors to a co-resident web server
  - It might not be a public information and could be damaging
  - Perform 1000 cache load measurements in which
    - (1) no HTTP requests are sent
    - (2) HTTP requests sent at a rate of 50 per minute
    - (3) 100 per minute
    - (4) 200 per minute

Results

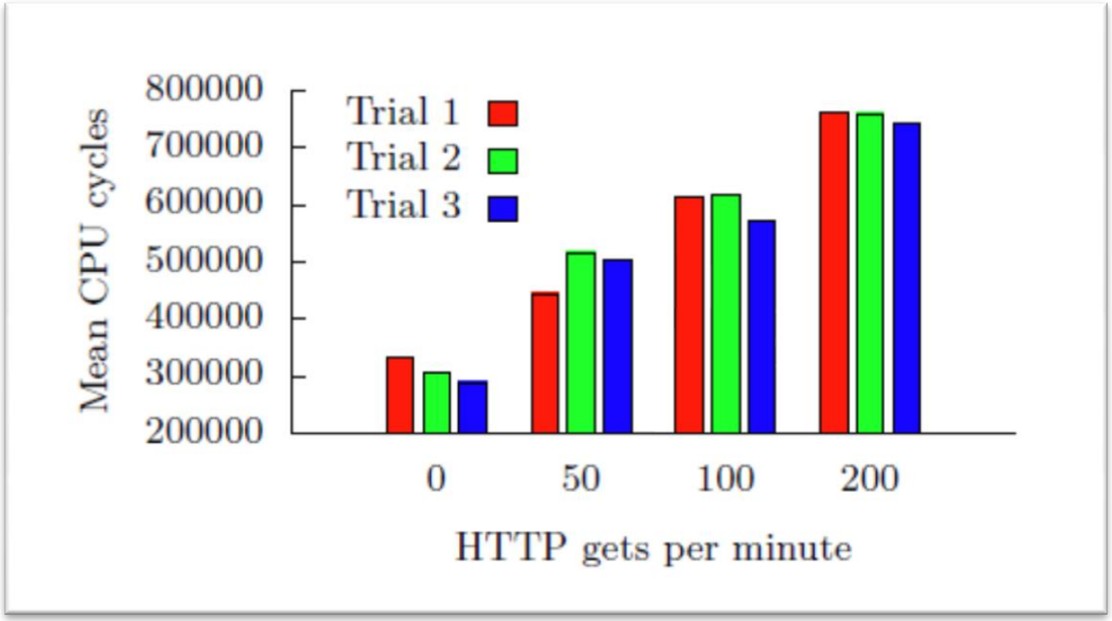


Fig 5. Mean cache load measurement timings (over 1 000 samples) taken while differing rates of web requests were made to a 3 megabyte text file hosted by a co-resident web server

There is a clear correlation between traffic rate and load sample

Exploiting Co-Residence

- Keystroke timing attack
  - Cache load measurements used to mount a key stroke attack
  - The goal is to measure the time between keystrokes made by a victim typing a password
  - Inter-keystroke times if properly measures can be used to perform recovery of the password
  - Experimental setup on a local testbed
    - Machine is completely idle; involved VMs are known $\rightarrow$ difficult to achieve in EC2 but a patient attacker might get lucky
    - Use *Prime+Trigger+Probe* to detect momentary activity spikes in a otherwise idle machine
    - Frequently perform load measurements
    - Report a keystroke when the probing measurement is between 3.1 μs and 9 μs (upper threshold filters out unrelated activity)

Exploiting Co-Residence

- Attacks are possible, however, they are not as sophisticated as one would hope
- The goal of the paper was to show the ability to cause co-residence and exploit in some way
- But, this is just the beginning

What Can Be Achieved? (Revisited)

1. Can one determine where in the cloud infrastructure an instance is located? **YES!**
2. Can one easily determine if two instances are co-resident on the same physical machine? **YES!**
3. Can an adversary launch instances that will be co-resident with other user's instances? **YES!**
4. Can an adversary exploit cross-VM information leakage once co-resident? **SORT OF**

Let's Discuss Countermeasures…

- Q1: Preventing cloud cartography
  - Dynamic IP allocation?
- Q2: Preventing co-residence checks
  - Prevent identification of Dom0?
- Q3: Preventing placement abuse
  - Allow users to decide?
- Q4: Preventing side-channel attacks
  - Good luck with this one!

Summary

- New risks from cloud computing exposed
- Shared physical infrastructure may and most likely will cause problems
  - Think of exploiting software vulnerabilities not addressed here
- Practical attack performed
- Some countermeasures proposed

- We will see much more of it!

Q&A!

Thank you!