Hey, You, Get Off of My Cloud:

Exploring Information Leakage in Third-Party Compute Clouds

Presenter: Ramya Pradhan

Course: CAP 6135, Spring 2012

Author information

- Dept. of Computer Science and Engineering, UCSD
  - Thomas Ristenpart, Hovav Shacham, Stefan Savage
- Computer Science and Artificial Intelligence Laboratory, MIT
  - Eran Tromer
- Presented at 2009 ACM Conference on Computer Security, Chicago, Illinois.

Problem (Opportunity)

- Maximize profit
- Customer seeking cloud's infrastructure as service
  - Low cost of operability
  - High scalability
  - Dynamic provisioning
- Cloud service provider
  - Multiplex existing resources

Problem (Opportunity)

- Trust relationship
  - Third-party infrastructure
- Threats from other customers
  - Physical resource sharing between virtual machines

Problem (Opportunity)

- Threats from other customers
  - Customer and adversary co-tenancy
  - Cross-VM attacks
  - Is it PRACTICAL?

Research questions

- Can my adversary know where I am?
- Can my adversary knowingly be my co-tenant?
- Can my adversary knowingly access shared resources when I access them?
- Can my adversary, being my co-tenant, steal my confidential information via cross-VM information leakage?

Testing platform

- Amazon's Elastic Compute Cloud (EC2)
  - Linux, FreeBSD, OpenSolaris, Windows
  - VM provided by a Zen hypervisor
    - Domain0 or Dom0
      - Privileged VM
      - Manages guest images, physical resource provisioning, access control rights
      - Routes guest images' packets via being a hop in traceroute

Testing platform

- Amazon's Elastic Compute Cloud (EC2)
  - Terminology
    - Image: user with valid account creates one or more of these
    - Instance:
      - Running image
      - One per physical machine
      - 20 concurrently running instances

Testing platform

- Amazon's Elastic Compute Cloud (EC2)
  - Degrees of freedom
    - Regions: US and Europe
    - Availability zones: infrastructure type
    - Instance type:
      - 32-bit architectures: m1.small, c1.medium
      - 64-bit architectures: m1.large, m1.xlarge, c1.xlarge

Testing platform

- Amazon's Elastic Compute Cloud (EC2)
  - Addressing
    - External IPv4 address and domain name
    - Internal RFC 1918 private address and domain name
    - Within cloud: domain names resolve to internal address
    - Outside cloud: external name maps to external address

Information collection tools

- nmap
  - TCP connect probes
    - 3-way handshake between source and target
- hping
  - TCP SYN traceroutes
    - Iteratively send packets until no ACK is received
- wget
  - Retrieve 1024 bytes from web pages

Information collection tools

- Evaluation
  - External probing: outside EC2 to instance in EC2
  - Internal probing: between two EC2 instances

Where is my target: Cloud cartography

- Hypothesis:

  Different availability zones likely to correspond to different internal IP address ranges. Similarly, different availability zones may correspond to different instance types.

Where is my target: Cloud cartography

- Facilitating service
  - EC2's DNS maps public IP to private IP
    - Infer instance type and availability zone

Where is my target: Cloud cartography

- Evaluation
  - External probing:
    - Enumerate public EC2-based web servers
    - Translate responsive public IPs to internal IPs using DNS queries within cloud

Where is my target: Cloud cartography

- Evaluation
  - Internal probing:
    - Launch EC2 instances of varying types
    - Survey resulting IP address assignment

Where is my target: Cloud cartography

- External probing
  - WHOIS query
  - Distinct IP address prefixes: /17, /18, /19
  - 57344 IP addresses found
  - 11315 responded to TCP connect probe on port 80
  - 8375 responded to TCP port 443 scan
  - ~14000 unique internal IPs

Where is my target: Cloud cartography

- Facilitating features of EC2
  - Internal IP address space cleanly partitioned
  - Instance types within partitions show regularity
  - Different accounts exhibit similar placement

Where is my target: Cloud cartography

- Evaluation results
  - Static assignment of IP addresses to physical machines
  - Availability zones use separate physical infrastructure
  - IP addresses repeated for instances from disjoint accounts only

Hide me: prevent cloud cartography

- Dynamic IP addressing
- Isolation of account's view of internal IP address space

Know thy neighbor: Determining co-residence

- Co-resident: instances running on same machine
- Conditions: any one of
  - Matching Dom0 IP address
  - Small packet round-trip times
  - Numerically close internal IP addresses

Know thy neighbor: Determining co-residence

- Matching Dom0 IP address
  - Dom0 always on traceroute
  - Instance owner's first hop
  - TCP SYN traceroute to target
  - Target's last hop

Know thy neighbor:

Determining co-residence

- Packet round-trip times
  - 10 RTTs
  - 1st always slow
  - Use last 9

Know thy neighbor:

Determining co-residence

- Internal IP addresses
  - Contiguous sequence of IP addresses share same Dom0 IP
    - 8 m1.small instances can be co-resident by design

Know thy neighbor:

Determining co-residence

- How to check
  - Communication between two instances
    - Possible: co-resident
    - Impossible: not co-resident
- Low false positives using three checks for matching Dom0 means two instances co-resident

NO thy neighbor:
Obfuscating co-residence

- Network measurement obfuscation techniques
  - Unresponsive Dom0 to traceroutes
  - Random internal IP generation at instance launch
  - Virtual LANS to isolate accounts
- Network-less based techniques for "know thy neighbor"? Is it possible?

You can run, but cannot hide: Exploiting Placement in EC2

- Attacker “places” its instance on the same physical machine as target
- How to place?
  - Brute-force placement
  - Heuristic-based placement

You can run, but cannot hide: Exploiting Placement in EC2

- Brute-force placement
  - Run many instances
  - Measure how many achieve co-residency
- Hypothesis

  Brute-force placement for large target sets allow reasonable success rates.

You can run, but cannot hide: Exploiting Placement in EC2

- Brute-force placement strategy
  - List targets
  - Group them by availability zones
  - For a long period of time run probe instances
    - If co-resident, successful placement
    - Else, terminate probe instance

You can run, but cannot hide: Exploiting Placement in EC2

- Brute-force placement strategy
  - List targets
  - Group them by availability zones
  - For a long period of time run probe instances
    - If co-resident, successful placement
    - Else, terminate probe instance

You can run, but cannot hide: Exploiting Placement in EC2

- Brute-force placement strategy: Results
  - List targets: 1686 servers (authors' creation)
  - Group by availability zones: m1.small, Z3
  - Run probe instances: 1785
  - Co-residency with 141 victims (8.4%)
- Naïve techniques can cause harm!

You can run, but cannot hide: Exploiting Placement in EC2

- Heuristic-based placement strategy
  - Launch instance soon after target launches
    - Instant flooding in appropriate zone and type
    - Why this works:
      - EC2 parallel placement algorithms
      - Servers only run when required
      - Server state monitoring using network probing
      - Auto-scaling systems

You can run, but cannot hide: Exploiting Placement in EC2

- Heuristic-based placement strategy
  - Experiment:
    - Victim launches 1, 10, 20 instances
    - Adversary floods 20 instances 5 minutes after victim
  - Result:
    - 40% Co-residency achieved
    - Failed when victim instances were large

YOU can run AND can hide: Patching Placement vulnerability

- Limited effectiveness:
  - Inhibiting cloud cartography and co-residence checks
- Absolute effectiveness:
  - Let the (YO)Users decide!
    - Request placements only for their instances
    - Pay opportunity cost for under-utilized machines

Walls have ears: Cross-VM Information Leakage

- Side-channel attacks using time-shared caches
  - Co-residence detection
  - Co-resident's web traffic monitoring
  - Timing co-resident's keystroke

Walls have ears: Cross-VM Information Leakage

- Time-shared caches
  - High load implies active co-resident
  - Adversary:
    - Places some bytes at a contiguous buffer
    - Busy-loop until CPU's cycle counter jumps to a large value
    - Measure time taken to again read placed bytes

Walls have ears, PLUG them: Inhibiting side-channel attacks

- Blinding techniques
  - Cache wiping, random delay insertion, adjust machine's perception of time
- But, are these effective?
  - Usually, impractical and application specific
  - May not be possible to PLUG all side-channels
- Only way: AVOID co-residence

In conclusion:

- Problem exists
- Risk mitigation techniques do just that – mitigate.
- Only way out:
  - Acknowledge the problem
  - Creative solutions are bound to come up

Strengths

- Effectively introduces the “Elephant in the room”
  - Information leakage between co-residents on a third-party cloud is UNAVOIDABLE
- Gives detailed experimental procedures
  - Helps with replication studies

Strengths

- Explores effective ways to unmask the problem
  - Network probing, cloud cartography, determining co-residency, exploiting placement policies
- Explores solutions to these problems
  - Inhibiting used from doing the above helps to some extent
  - ONLY current solution: Let the user know.

Weakness

- This scheme does not work on a target on a full system.
- Open to interpretation "… accounts under our control"
  - Amazon acknowledged this study as "controlled experiment"

    (http://www.techworld.com.au/article/324189/amazon_downplays_report_highlighting_vulnerabilities_its_cloud_service)
  - Authors mean "accounts that they created", and not "controlled experiment".

Possible extensions

- Bring more awareness to users
  - More papers without scope for interpretation ambiguity
  - Collaborate research efforts with other universities
  - Explore similar vulnerabilities with other cloud providers
    - Authors say they exist, but proof is required
- Mathematically model the phenomenon.

Questions?

Thank You!!