VM and Cloud Security

Yiying Zhang

Outline

- VM security concerns and possible solutions
- Cloud security concerns and possible solutions
- Other security issues

Traditional vs VM-Based Computing

- Traditional server
  - Fixed location, management, lifetime, etc.
  - Homogeneous OS, single owner, etc.
- VMs
  - More flexibility in creating, migrating, pausing, shutting down VMs
    - Implications: harder to enforce security tasks in a more dynamic environment
  -

A Potential VM Security Solution

- Proposal: a ubiquitous virtualization layer for security functions
  - Hypervisor + distributed
- Benefits
  - Centralizes security management by moving from guest OS to this layer
  - Can perform security functions when VMs are offline
  - Independent of where VMs run or migrate to

Scaling

- Traditional computing: IT manually provisions hardware, sets it up
  - More controlled and limited (by manual set up time and budget)
- VMs: easy to create and duplicate
  - As easy as copying a file, can be done over the network

Scale - Security Problems

- Need to apply security patches and other configurations to all VMs
- But administrative work is not fully automated
- Large number of VMs can significantly increase the overhead of these tasks

Scale – Solution

- The ubiquitous security layer allows admin to manage VMs without an account in the OS
- Allows management tasks to be automated
- Can limit the VMs coming in and out of the infrastructure
- Can scan, patch, purge, update and manage offline, online & archived VMs

Transience - Security Problems

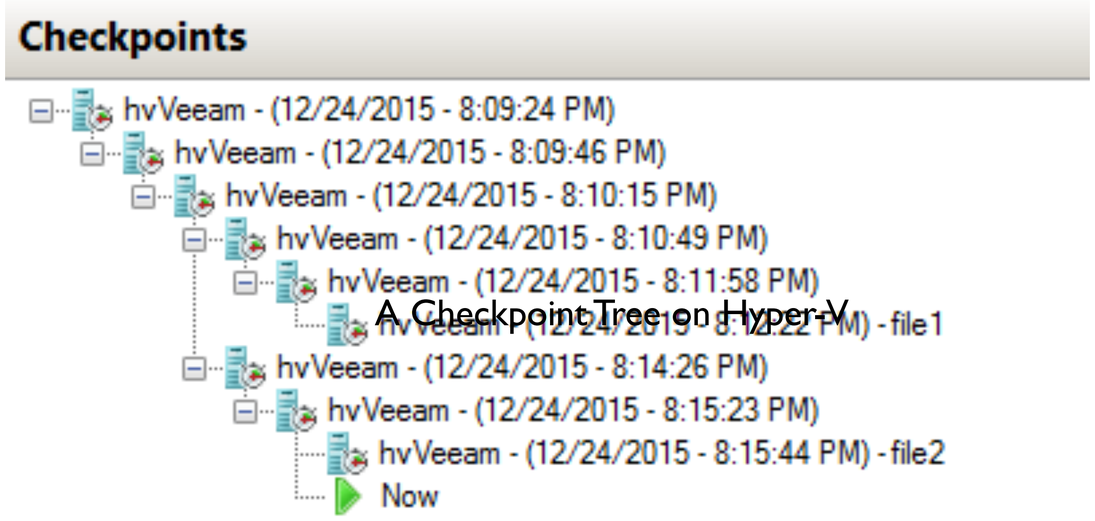
- Traditional computing: servers are online most of the time
  - Easy to detect attacks, as many machines can be infected and stay infected
- VMs: can appear and disappear sporadically
  - Infected VM can infect other VMs and then go offline => harder to detect

Transience – Solution

- Using the ubiquitous layer to
  - Control VMs in and out
  - Prevent VMs from being placed on certain hosts
  - Track what is on the network

Software Lifecycle

- Traditionally a machine's lifecycle is a straight line
  - Execution, config changes, patches integrated, updates, etc.
- VM lifecycle is like a tree: a state can be branched off into multiple forks
  - Can role back to any checkpoint



A Checkpoint Tree on Hyper-V

Image from: https://www.veeam.com/blog/what-are-hyper-v-snapshots-12-things-to-know.html

Software Lifecycle - Security Problems

- Rollbacks can re-expose patched vulnerabilities
- Re-enable previous accounts/passwords
- Bring back previously retired encryption keys
- Reintroduce worms/viruses/malware
- Random number generation might not be random
- Same key could be used to encrypt 2 different plain texts – leaking information
- Repeated TCP SYN/ACK numbers risk protocols that rely on freshness
- and many more …

Software Lifecycle – Solutions

- Move security relevant state out of the guest OS to the ubiquitous layer
  - e.g., a dedicated distribute storage location that cannot be rolled back
- Obtain randomness from VMM
- Replace lifecycle dependent algorithms with lifecycle independent ones
- Send notification from VMM to guest OS when guest OS restarts

Diversity - Security Problems

- Traditionally, easier to tackle security by enforcing on homogeneous machines and OSs
- VMs have more diversity
  - Harder to enforce security on heterogenous OSs and hardware

Diversity – Solutions

- Allows enforcing of policy specific to the vulnerability from outside
  - Not relying on (guest) OS patches
- Moving management to ubiquitous layer avoids version dependencies
  - Regardless of what versions of software/OS VMs run

Mobility - Security Problems

- VMs can migrate across servers
  - A compromised server can infect all VMs that ever live on it
  - A VM’s safety implies all servers it ever lives on need to be safe
  - VM migration means it’s easier to steal data

Mobility - Solution

- Enforcing security at the ubiquitous layer
  - Independent of how VMs move

Identity – problems

- Physical servers: ownership is defined (via MAC add, hostname, employee, etc.)
- VMs: dynamic in nature
  - Random MAC address allocation
  - Hostnames are configurable
  - VMs can be migrated
  - VMs can be passed from one owner to another

Identity - solution

- Assign virtual identity to the VM
  - Hostname, MAC addresses, IP addresses, keys
- Reintroduce a notion of ownership, responsibility, and machine history

Data Lifetime – problems

- General security goal: limit the time sensitive data remains in the system
- Traditional computing: sensitive not persisted
- VMs: log execution states to enable rollback
  - Sensitive data can be leaked
  - and spread across machines

Name	Size	Type	Modified
vSphere Management Assistant...	2.41 KB	Virtual Machine	7/30/2012 3:56:05 PM
vSphere Management Assistant...	0.28 KB	File	6/8/2012 12:17:46 PM
vSphere Management Assistant...	0.00 KB	File	8/25/2011 7:56:35 AM
vSphere Management Assistant...	5,242,880.00 KB	Virtual Disk	6/14/2012 10:08:21 AM
vSphere Management Assistant...	8.48 KB	Non-volatile memory file	7/30/2012 3:56:05 PM
vmware-7.log	42.73 KB	Virtual Machine log file	6/14/2012 9:32:47 AM
vmware-2.log	38.15 KB	Virtual Machine log file	10/19/2011 6:30:10 AM
vmware-3.log	41.24 KB	Virtual Machine log file	6/8/2012 8:19:57 AM
vmware-4.log	36.97 KB	Virtual Machine log file	6/8/2012 8:21:04 AM
vmware-5.log	36.97 KB	Virtual Machine log file	6/8/2012 8:21:46 AM
vmware-6.log	82.70 KB	Virtual Machine log file	6/8/2012 11:57:25 AM
vmware.log	39.82 KB	Virtual Machine log file	7/30/2012 3:56:05 PM

Image from: https://www.vmwarearena.com/configuring-virtual-machine-log-rotation/

Data Lifetime – Solutions

- Limit access to VMs containing sensitive data
- Data-at-rest encryption: encrypting data at the hypervisor

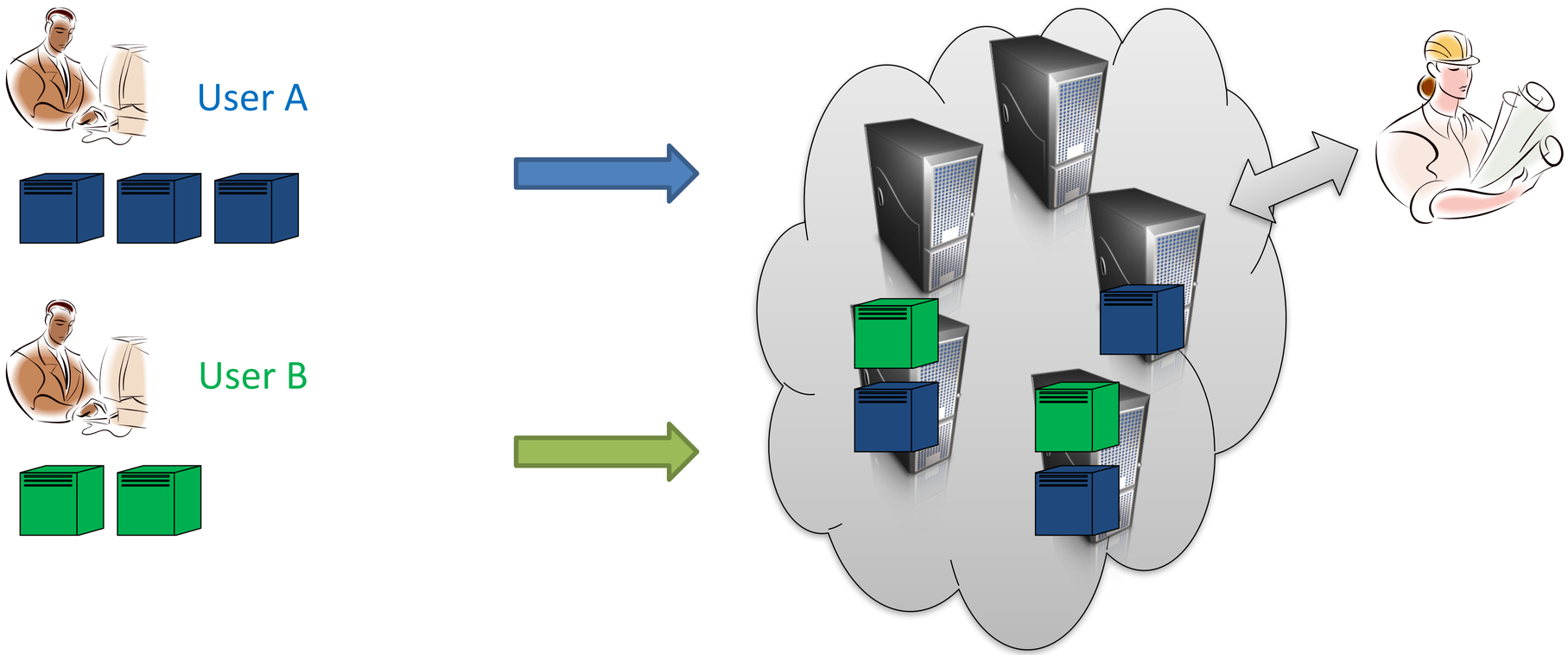
Discussion

- Do these problems apply to containers? What about the solutions?
- Any cons of having the security and manageability layer in the hypervisor?

Slides coming from: Cloud security

Tom Ristenpart
CS 6431

Trust models in public cloud computing



Security threats in public cloud computing?

External attacks against infrastructure

Adversarial provider spying on running VMs / data

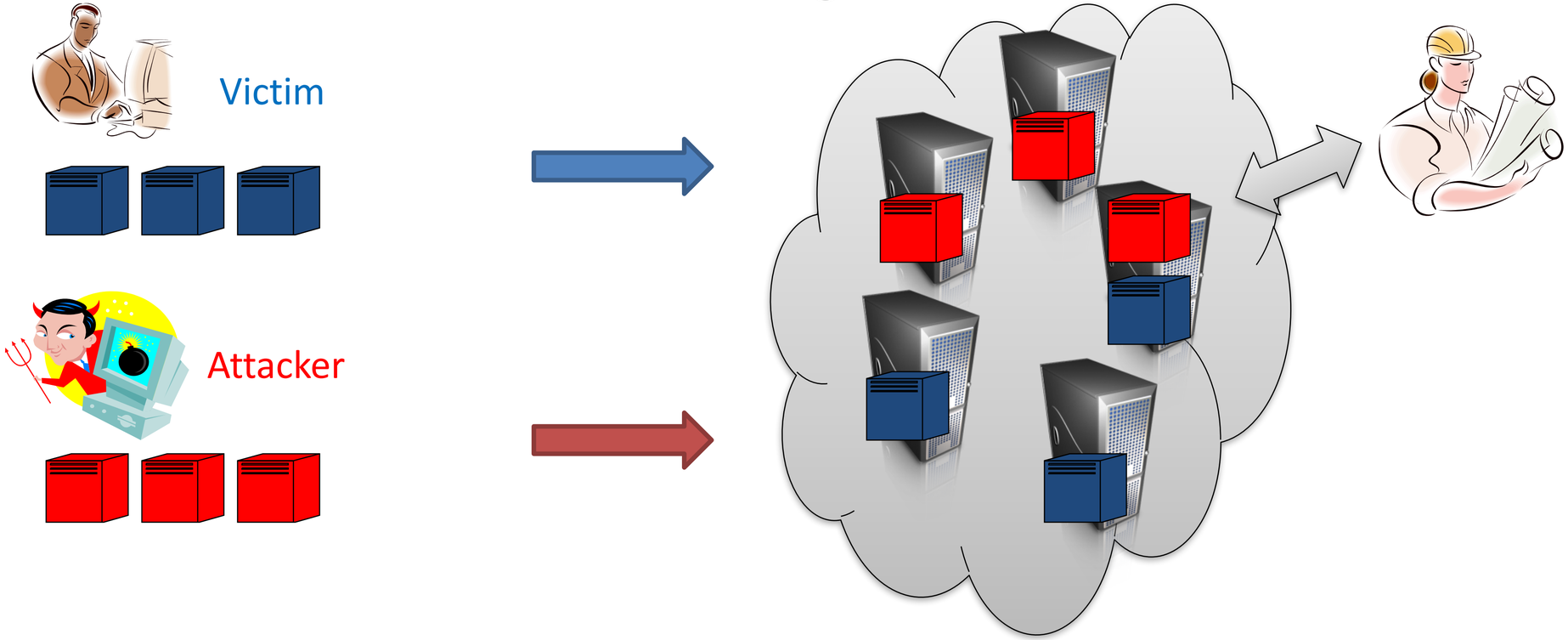
Cross-user attacks

- VM image attacks
- Side-channels attacks

Cloud provider attacking users

- Steal user data
- Infer user access patterns
- Not running the right user code

Threats due to resource sharing



Attacker identifies one or more victims VMs in cloud

1) Achieve advantageous placement via launching of VM instances

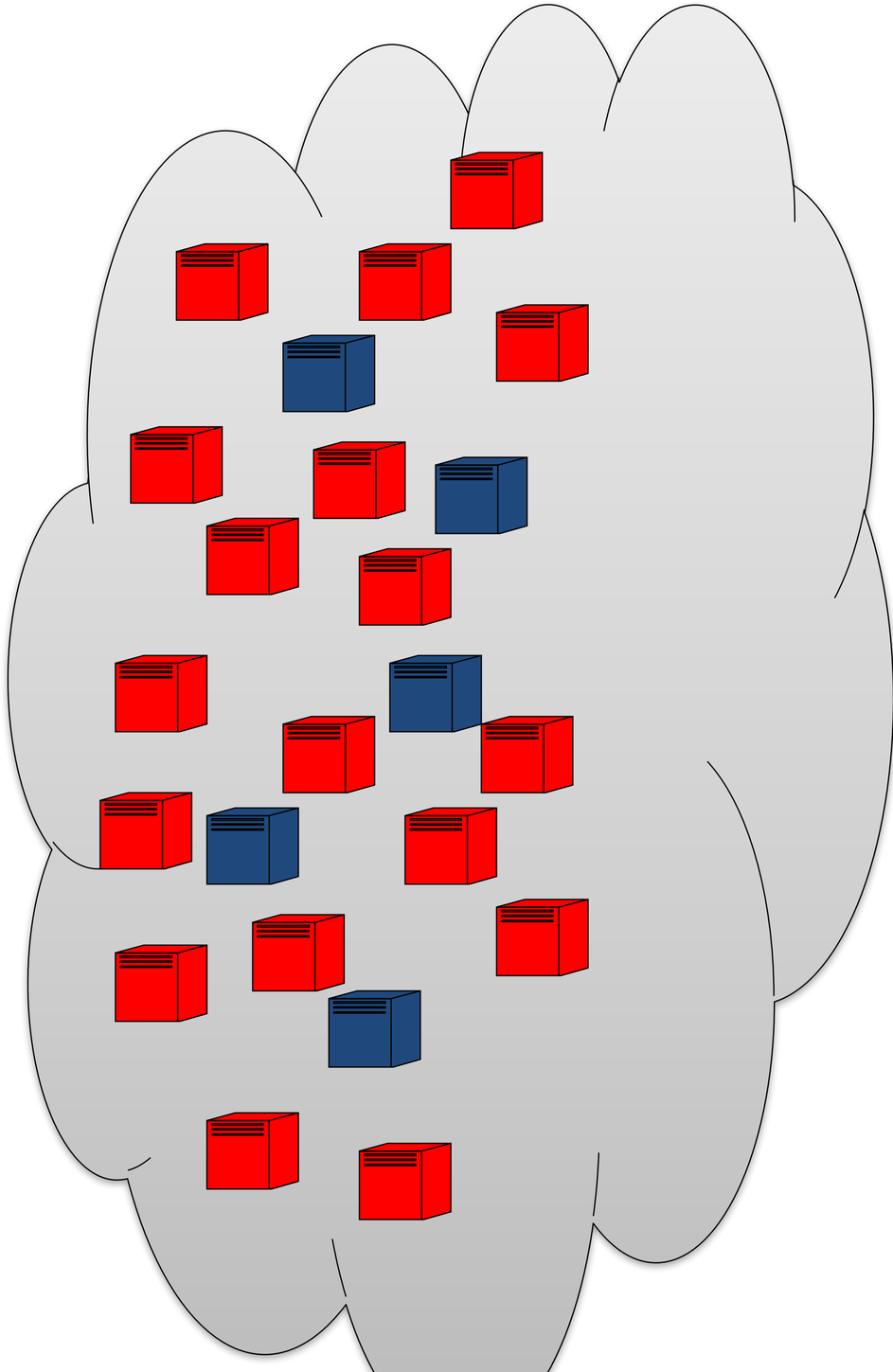
2) Launch attacks using physical proximity

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

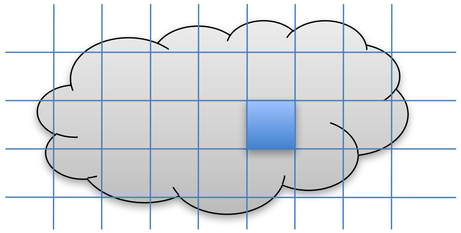
Can attackers do better?



Outline of a more damaging approach:

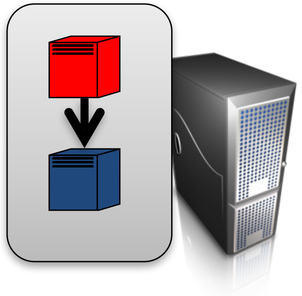
1) Cloud cartography

map internal infrastructure of cloud
map used to locate targets in cloud



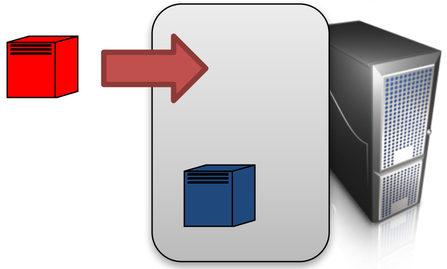
2) Checking for co-residence

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

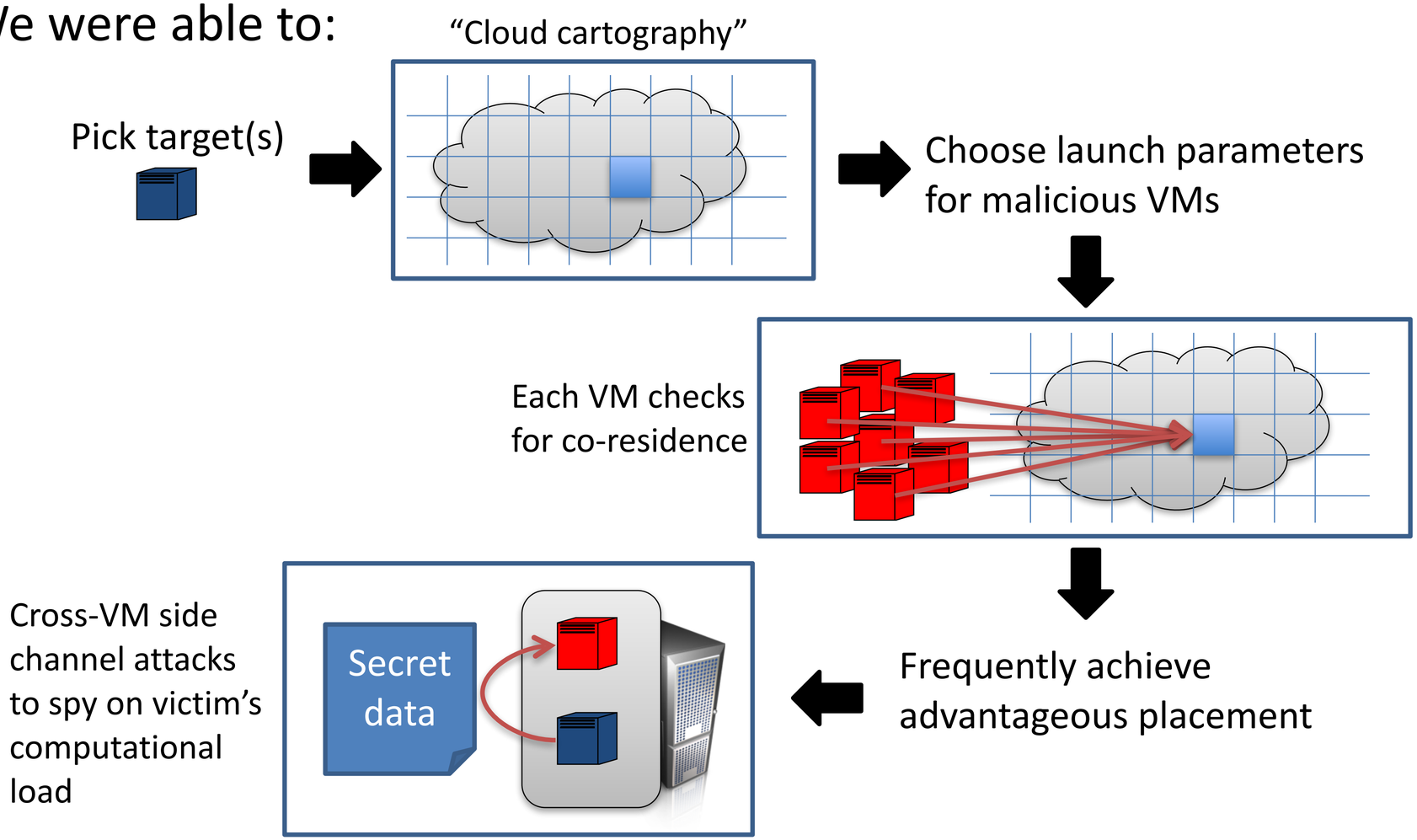
Case study with Amazon's EC2

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:



Some info about EC2 service (at time of study)

Linux-based VMs available
Uses Xen-based VM manager

launch parameters

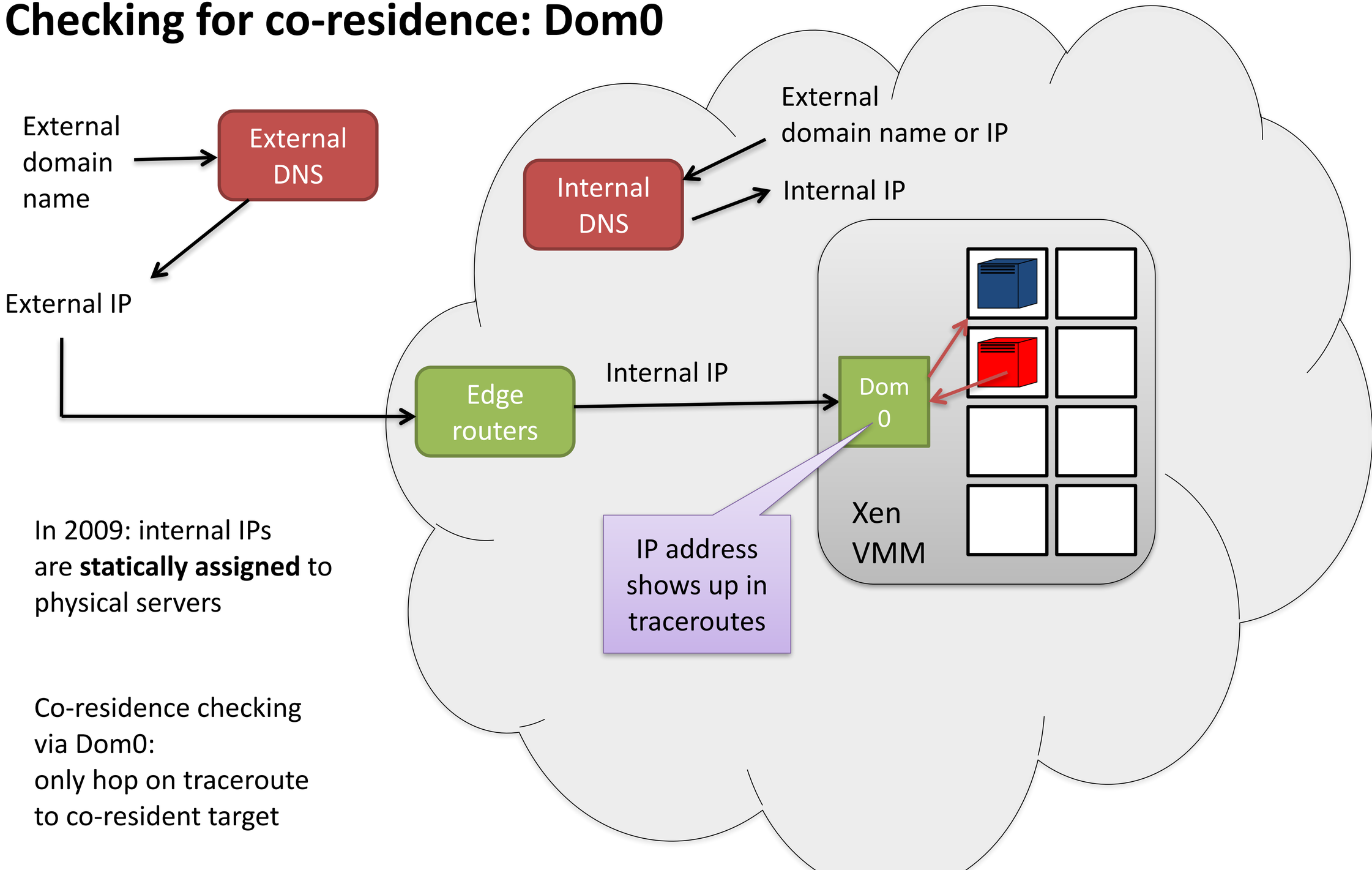
- User account
- 3 “availability zones” (Zone 1, Zone 2, Zone 3)
- 5 instance types (various combinations of virtualized resources)

Type	gigs of RAM	EC2 Compute Units (ECU)
m1.small (default)	1.7	1
m1.large	7.5	4
m1.xlarge	15	8
c1.medium	1.7	5
c1.xlarge	7	20

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.

Checking for co-residence: Dom0



In 2009: internal IPs are **statically assigned** to physical servers

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

Checking for co-residence: Dom0 and RTT effectiveness

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

Dom0 check works

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

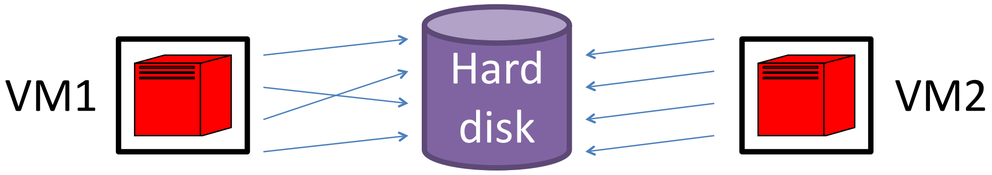
RTT times also indicate co-residence

	Median RTT (ms)
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

Checking for co-residence: Covert channels

Covert channels allow cooperating sender and receiver to send a message to each other while skirting information flow controls

Lampson. A Note on the Confinement Problem. 1973



Sender transmits ‘1’ by franticly reading random locations

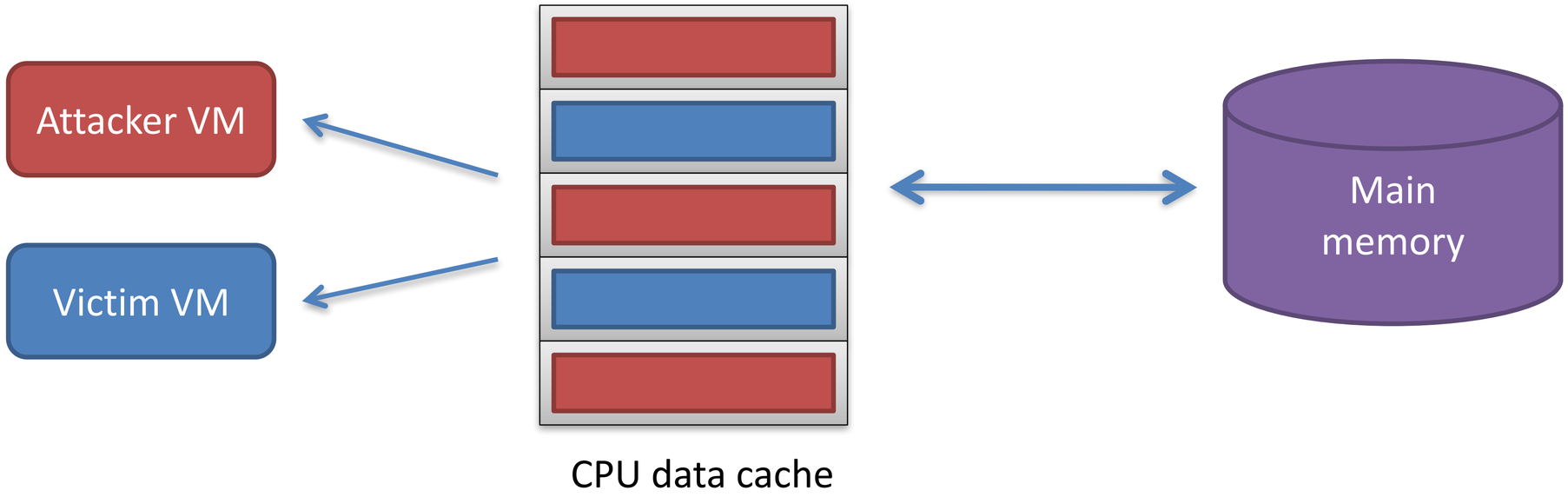
Sender transmits ‘0’ by doing nothing

Receiver times reading of a fixed location

Covert channels require control of both VMs:
Not directly useful to determine co-residency with target victim

Checking for co-residence: Side channels

A side-channel is an unintentional leakage of information about confidential process

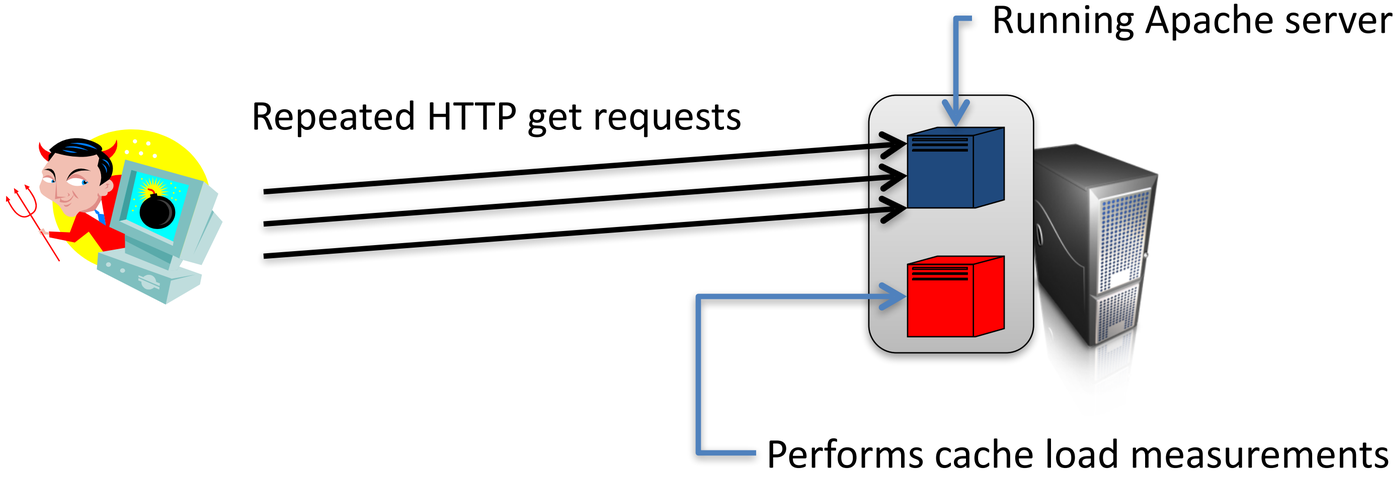


1) Read in a large array (fill CPU cache with attacker data)

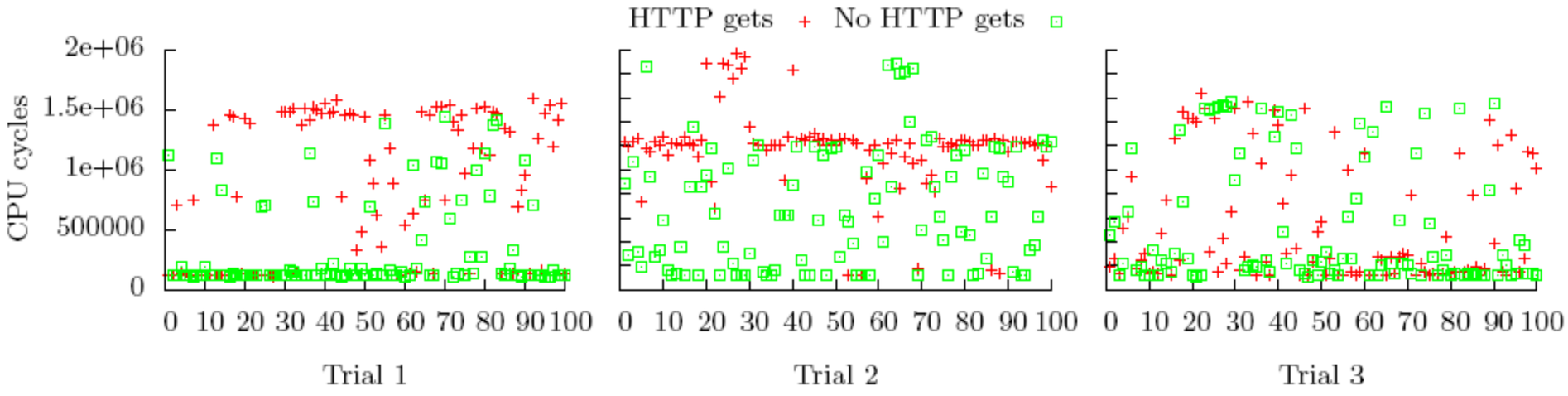
2) Busy loop (allow victim to run)

3) Measure time to read large array (the *load measurement*)

Checking for co-residence: Side channels effectiveness



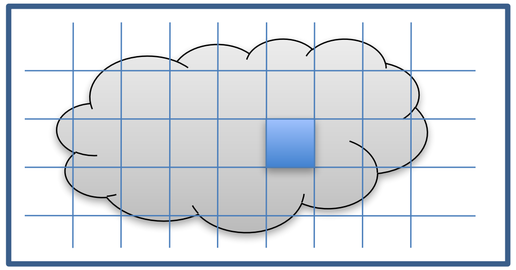
3 pairs of instances, 2 pairs co-resident and 1 not
100 cache load measurements during **HTTP gets** (1024 byte page) and with **no HTTP gets**



Instances co-resident | Instances co-resident | Instances NOT co-resident

Effective Attacker Launch: Cloud cartography

Pick target(s)



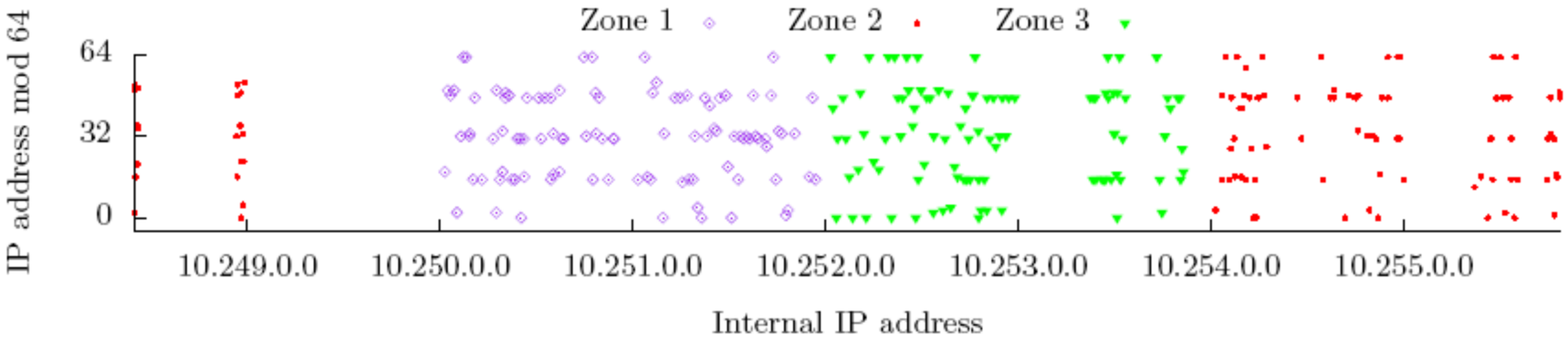
Choose launch parameters for malicious VMs

launch parameters

3 "availability zones"
(Zone 1, Zone 2, Zone 3)

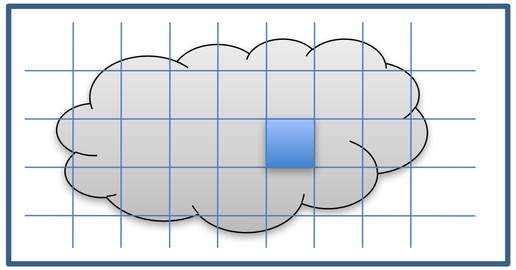
5 instance types
(m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)

User account



Effective Attacker Launch: Cloud cartography

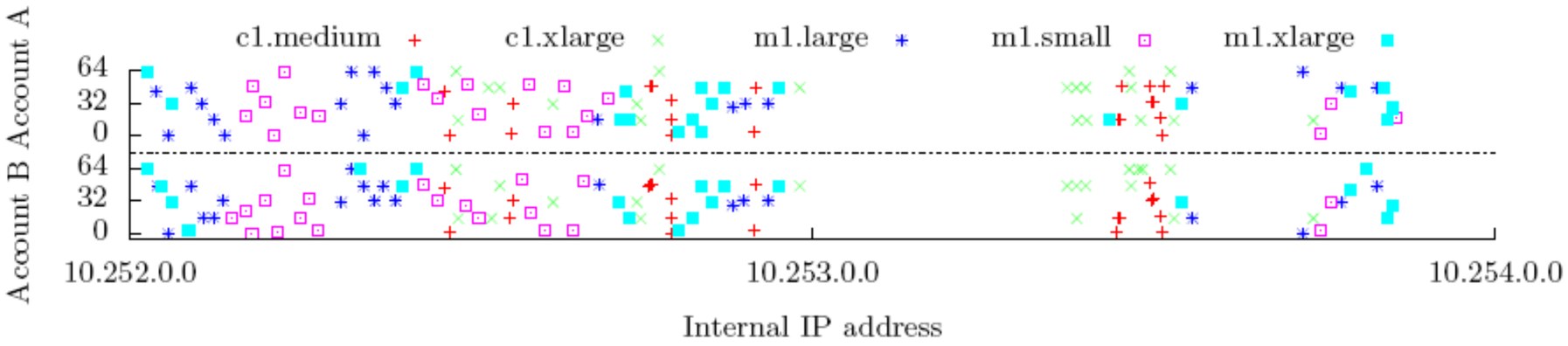
Pick target(s)



Choose launch parameters for malicious VMs

launch parameters

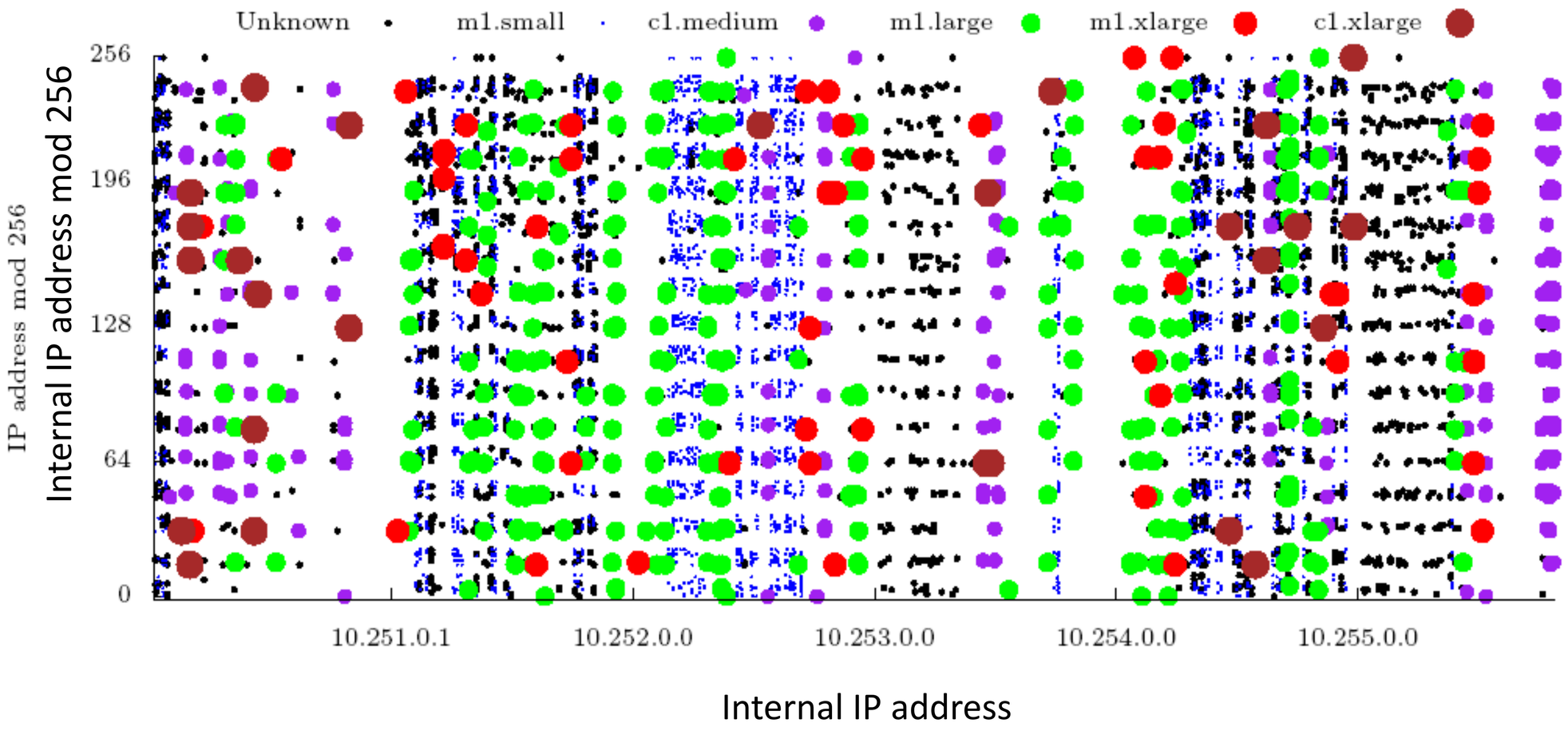
- 3 "availability zones"
  (Zone 1, Zone 2, Zone 3)
- 5 instance types
  (m1.small, c1.medium, m1.large, m1.xlarge, c1.xlarge)
- User account



Associate to each /24 an estimate of Availability zone and Instance Type

External IP $\xrightarrow{\text{DNS}}$ Internal IP $\xrightarrow{/24}$ Availability zone, Instance Type

Mapping 6,577 public HTTP servers running on EC2 (Fall 2008)



Effective Attacker Launch: 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

<u>Experiment:</u>

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

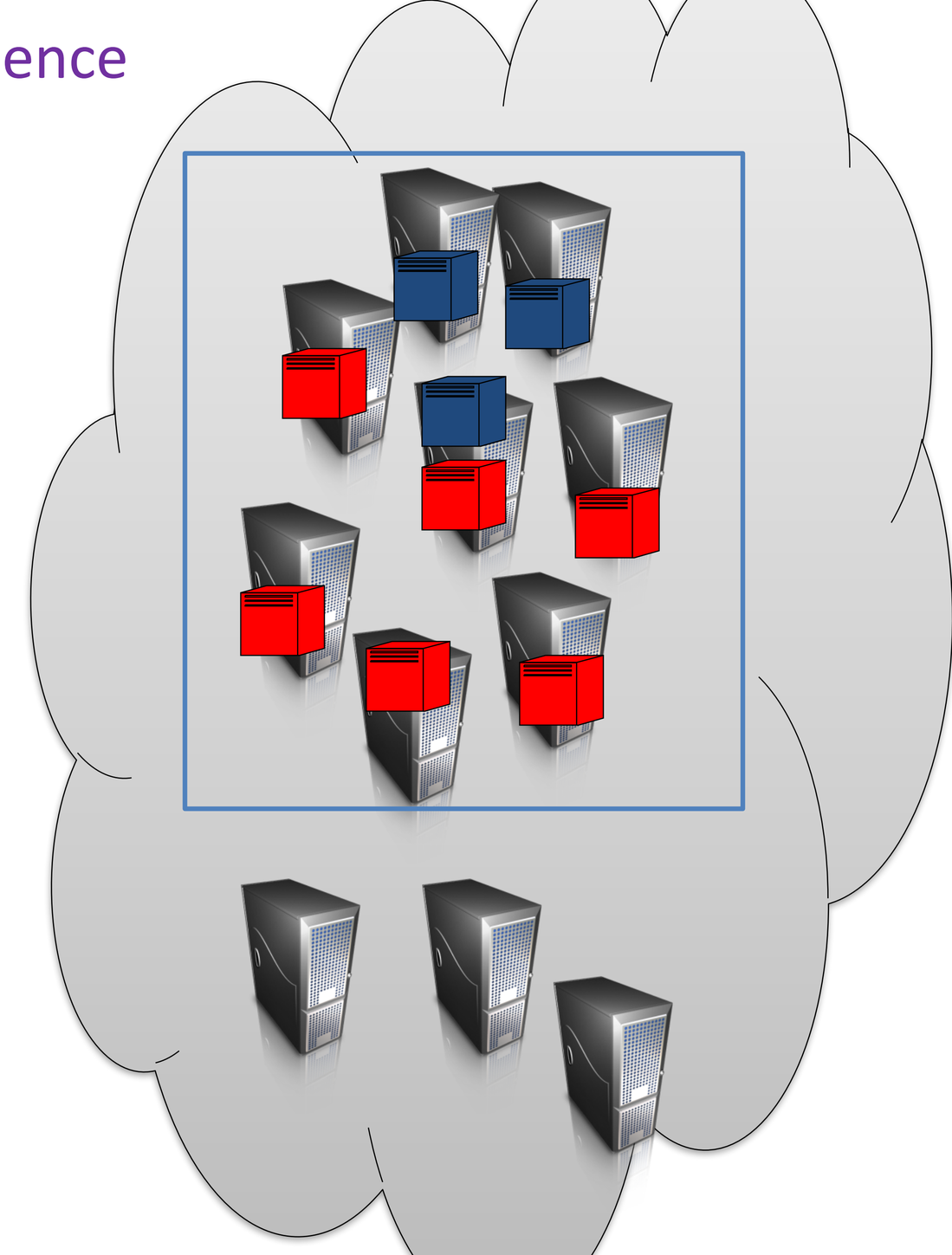
1,785 "attacker" instances launched over 18 days

Each checked co-residence against all targets using Dom0 IP

Results:

78 unique Dom0 IPs

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



Effective Attacker Launch: Achieving co-residence

Instance flooding near target launch abuses parallel placement locality

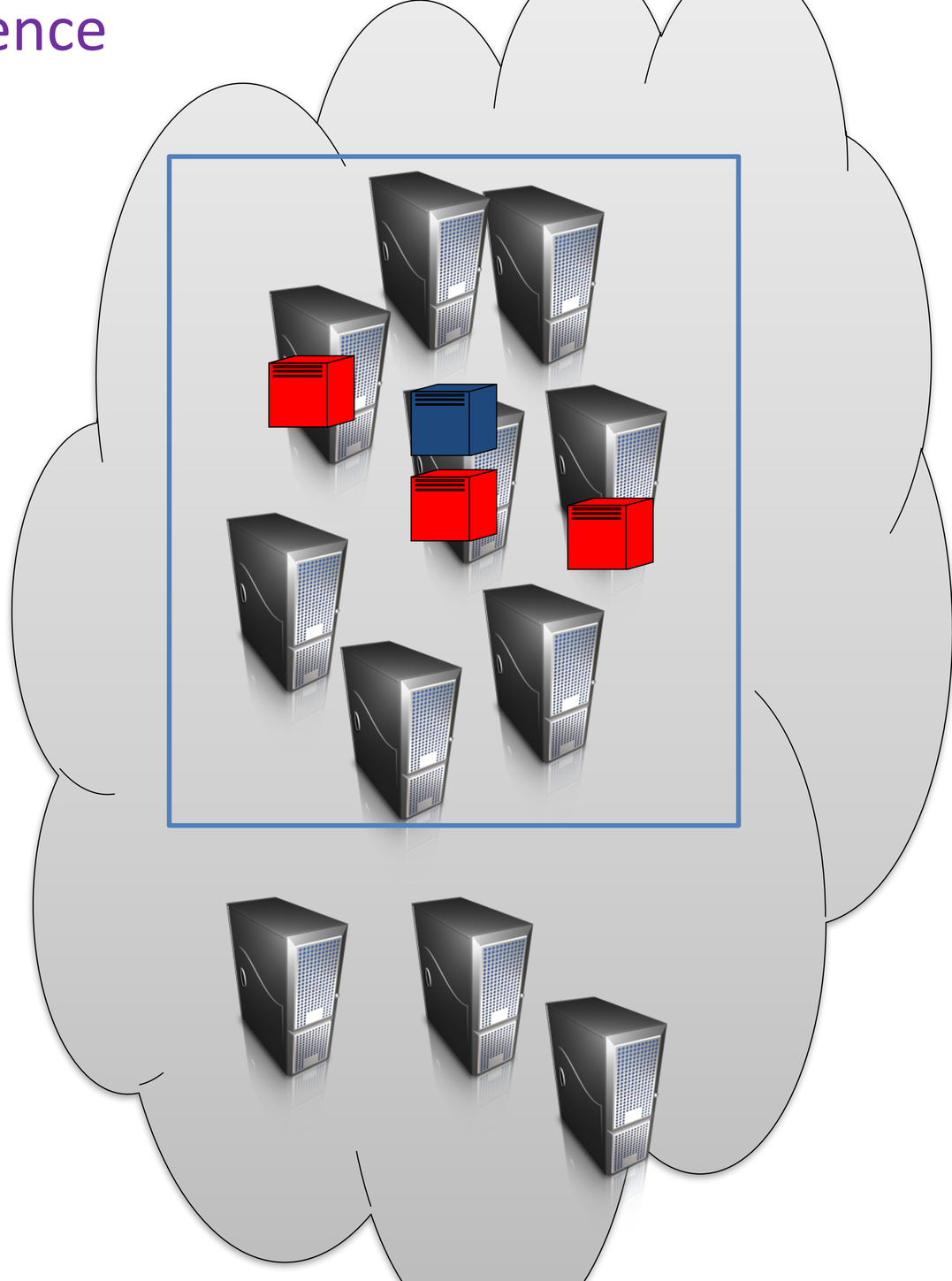
Launch many instances in parallel into known zone/type near time of target launch

<u>Experiment:</u>

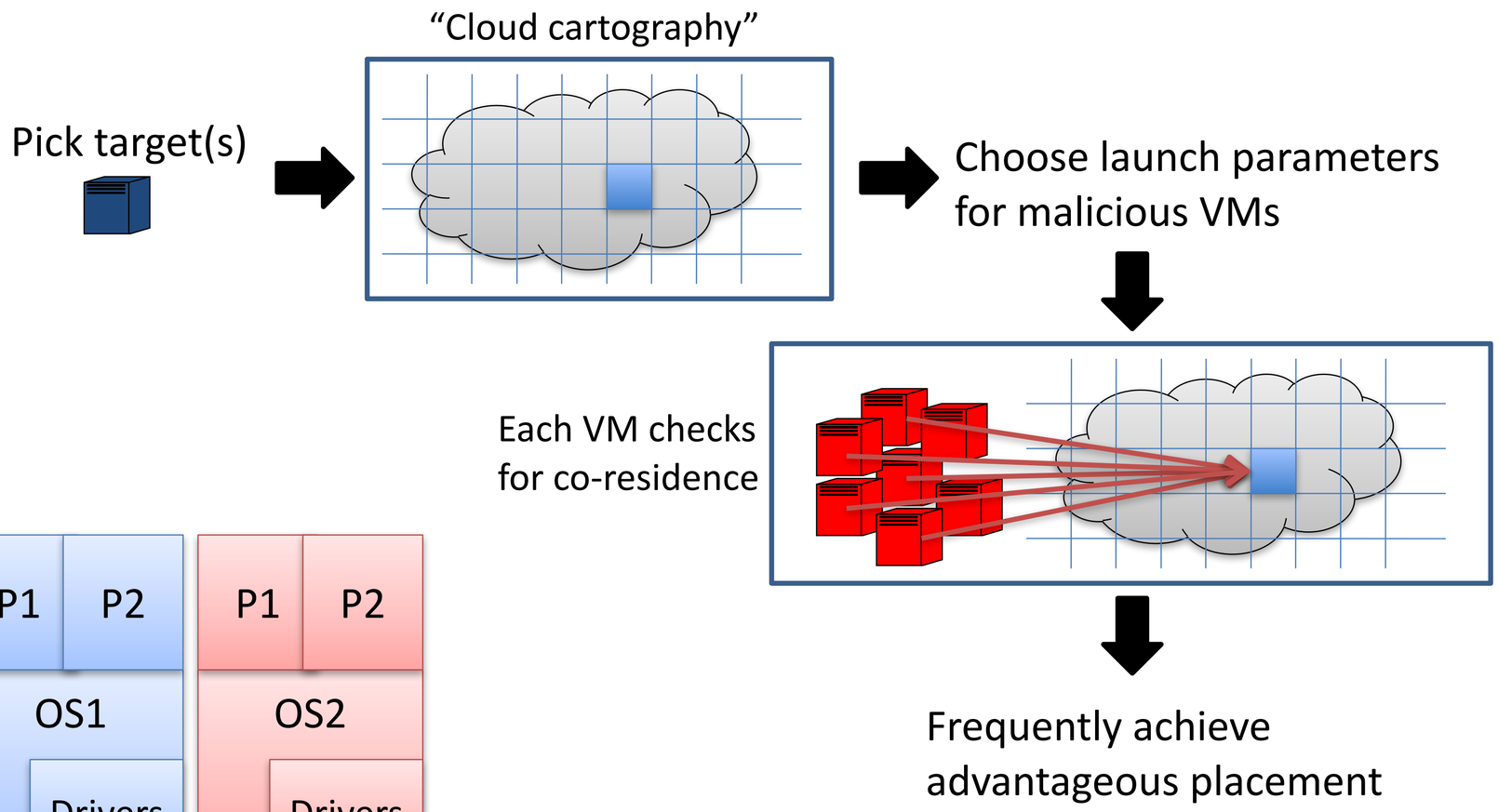
Repeat for 10 trials:

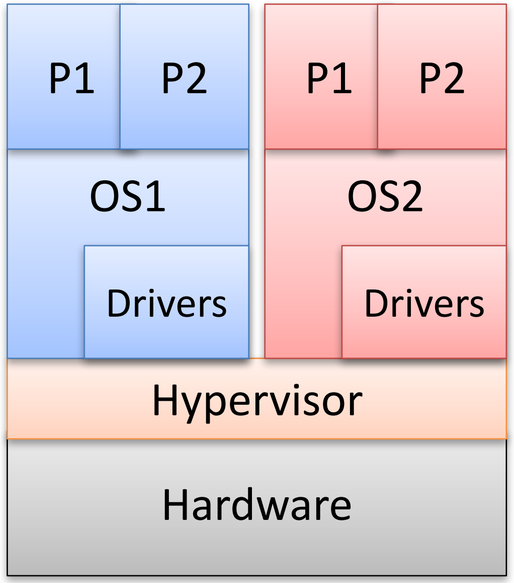
1) Launch 1 target VM (Account A)
2) 5 minutes later, launch 20 “attack” VMs (alternate using Account B or C)
3) Determine if any co-resident with target using Dom0 IP

4 / 10 trials succeeded



So far we were able to:

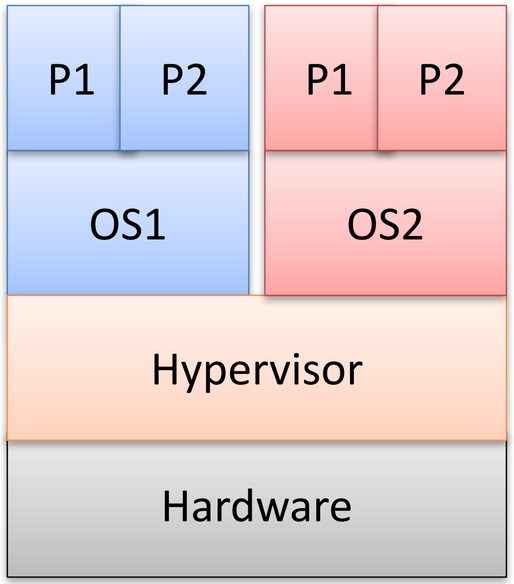




This shouldn't matter if VMM provides good isolation!

Actual Attack: Violating isolation

- Cross-VM side-channel attacks
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability
- Escape-from-VM vulnerabilities



Actual Attack: Prime+Trigger+Probe

- (1) Prime: Read buffer B at s-byte offsets in order to ensure it is cached.
- (2) Trigger: Busy-loop until the CPU's cycle counter jumps by a large value. (This means attacker VM was preempted by the Xen scheduler, hopefully to run the victim VM.)
- (3) Probe: Measure the time it takes to again read B at s-byte offsets.

- Many other types of side-channel attacks (e.g., Spectre, Meltdown)

What can cloud providers do?

1) Clo

2) Che
co-re

3) Ac
co-residence

4) Si
infor

Newer instances on EC2 use virtual private networks by default

Amazon doesn't report Dom0 in traceroutes anymore

Amazon provides dedicated instances now.
They cost more.

<u>Possible counter-measures:</u>

- Random Internal IP assignment
- Isolate each user's view of internal address space

- Hide Dom0 from traceroutes

- Allow users to opt out of multitenancy

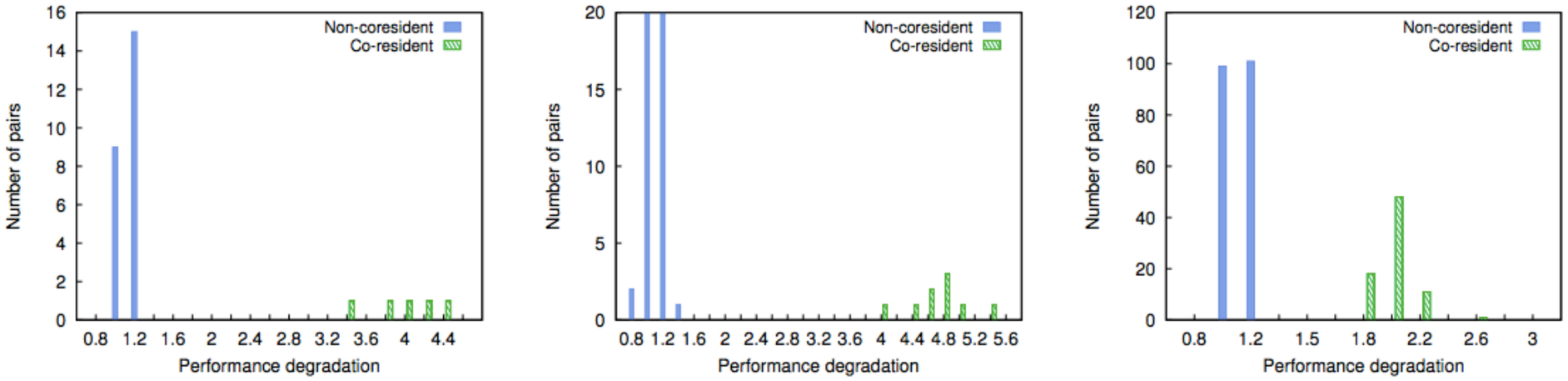
- Hardware or software countermeasures to stop leakage [Ber05,OST05,Page02,Page03, Page05,Per05]
- Improved performance isolation

Changes since 2009

- Turned off Dom0
- Virtual Private Clouds
  - Instances launched into virtual private networks
  - Logically isolated namespace
    - Prevents cloud cartography based on internal IPs
- Scale significantly larger
  - Liu 2013: ~450,000 servers for EC2

Newer placement studies

- Varadarajan et al. and Xu et al. (USENIX 2015)
- New co-residence tests, explore larger variety of settings
- Conclusions: can still achieve co-residency



(a) GCE (b) EC2 (c) Azure – Intel machines

Discussion

- Co-residence attack for container-based cloud services?

Other Security Concerns in the Cloud

- Cloud service provider being the attacker (e.g., an insider employee)
  - What if an OS/hypervisor is compromised?
  - What if someone can snoop the network link or memory bus?
  - Does the cloud faithfully execute my code?
  - Can the cloud provider steal my data?
- Real concerns that prevents sensitive workloads to migrate to the cloud

Confidential Computing

- High-level idea is to untrust the service provider, although different providers have different definitions
- One approach: establish a secure “enclave” that runs user application and plain-text user data.
  - Use a hardware root of trust which “attests” the software running on it
  - Encrypts data whenever it leaves the hardware (e.g., leaves the CPU chip)
  - Checks the integrity of data
  - Example: Intel SGX, AMD SEV (Cloud offering: Azure Confidential Computing)
  - Does not address side channels
- Another approach: partition all hardware into a safe world and an unsafe world (e.g., ARM Trustzone)