New Linux/Windows Malware Allows Arbitrary Execution of Shell Commands (bleepingcomputer.com)

50

Posted by EditorDavid on Saturday November 23, 2019 @01:34PM from the through-the-backdoor dept.

"Researchers have discovered a new multi-platform backdoor that infects Windows and Linux systems allowing the attackers to run malicious code and binaries on the compromised machines," reports Bleeping Computer:

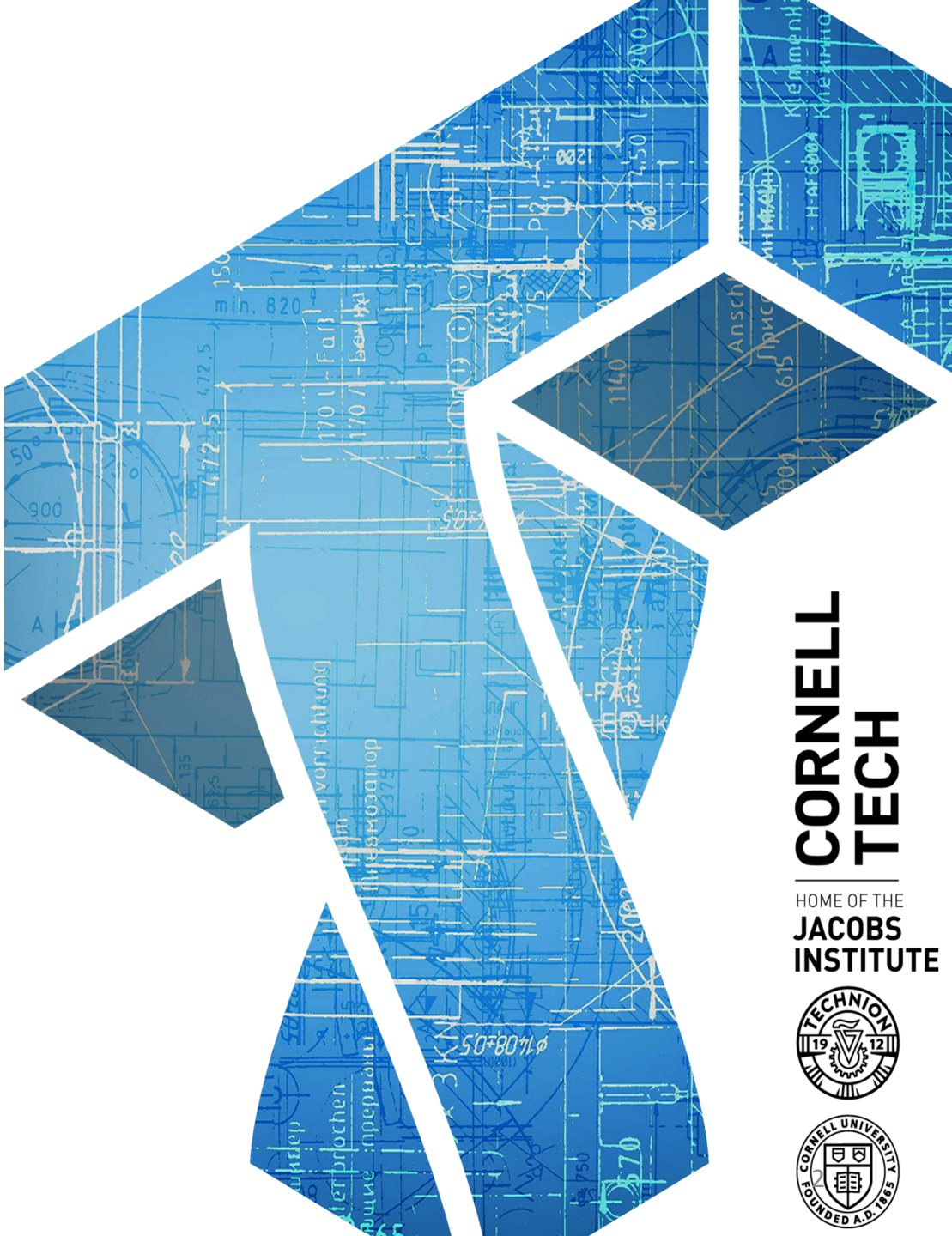
> The malware dubbed ACBackdoor is developed by a threat group with experience in developing malicious tools for the Linux platform based on the higher complexity of the Linux variant as Intezer security researcher Ignacio Sanmillan found. "ACBackdoor provides arbitrary execution of shell commands, arbitrary binary execution, persistence, and update capabilities," the Intezer researcher found.
>
> Both variants share the same command and control (C2) server but the infection vectors they use to infect their victims are different: the Windows version is being pushed through malvertising with the help of the Fallout Exploit Kit while the Linux payload is dropped via a yet unknown delivery system... Besides infecting victims via an unknown vector, the Linux malicious binary is detected by only one of the anti-malware scanning engines on VirusTotal at the time this article was published, while the Windows one is detected by 37 out of 70 engines. The Linux binary is also more complex and has extra malicious capabilities, although it shares a similar control flow and logic with the Windows version...
>
> ACBackdoor can receive the info, run, execute, and update commands from the C2 server, allowing its operators to run shell commands, to execute a binary, and to update the malware on the infected system.

The article warns that the Linux version will disguise itself as the Ubuntu UpdateNotifier utility, renaming its process as the Linux kernel thread *[kworker/u8:7-ev]*.

CS 5435: Virtualization & Side-channels

Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2019



CORNELL TECH

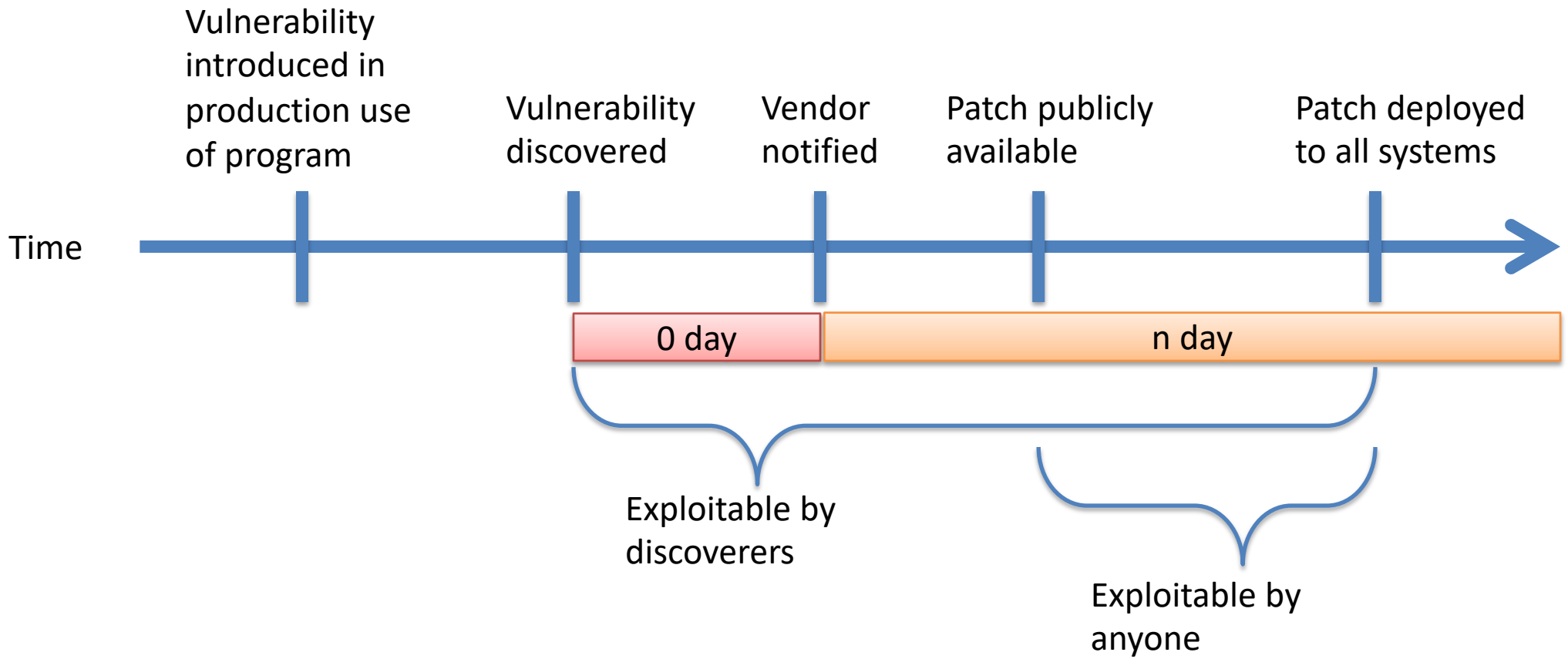
HOME OF THE JACOBS INSTITUTE

Vulnerabilities happen…

- Countermeasures to prevent exploitation
  - ASLR, canaries, W^X
- Discovering and patching vulnerabilities
  - Manual analysis, static analysis, fuzzing,

Patching and vulnerability lifecycles

Imagine an exploitable vulnerability in some widely used software

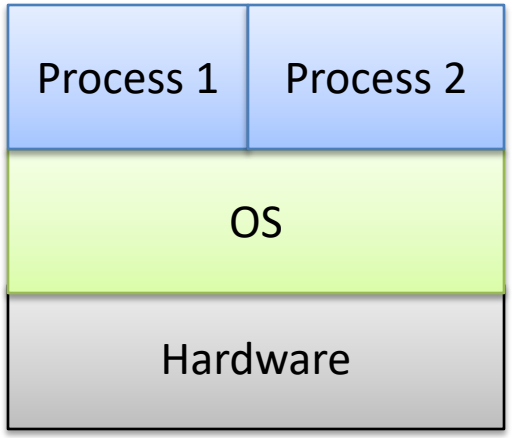


Exploitable vulnerabilities aren't going away. How to do we limit their ill-effects?

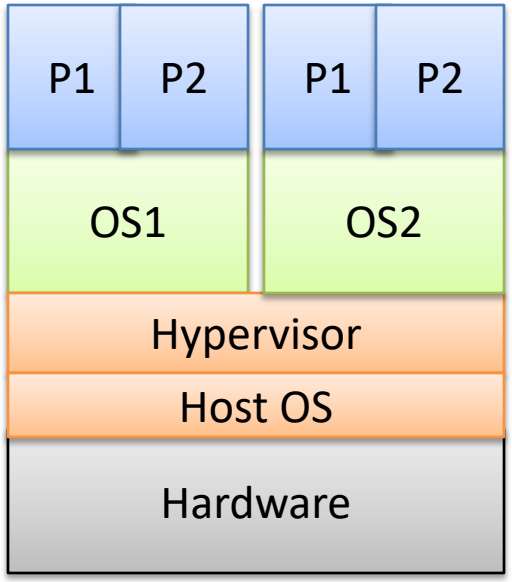
Containment and isolation

- We can try to use software/hardware mechanisms to contain exploited programs
- File system jails early example:
  - chroot: change apparent root directory for a process
  - jail (in FreeBSD): chroot + individual hostname/IP/root
- Containers
  - OS isolation of different process families
- Virtual machines
  - Different guest operating systems and programs isolated by hypervisor

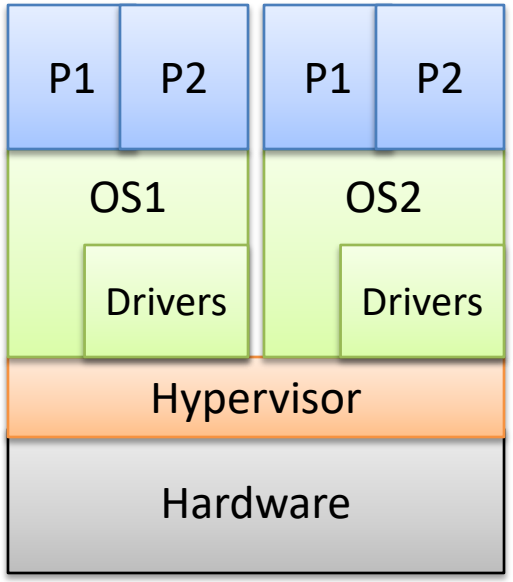
Virtualization



No virtualization



Full virtualization + Type 2



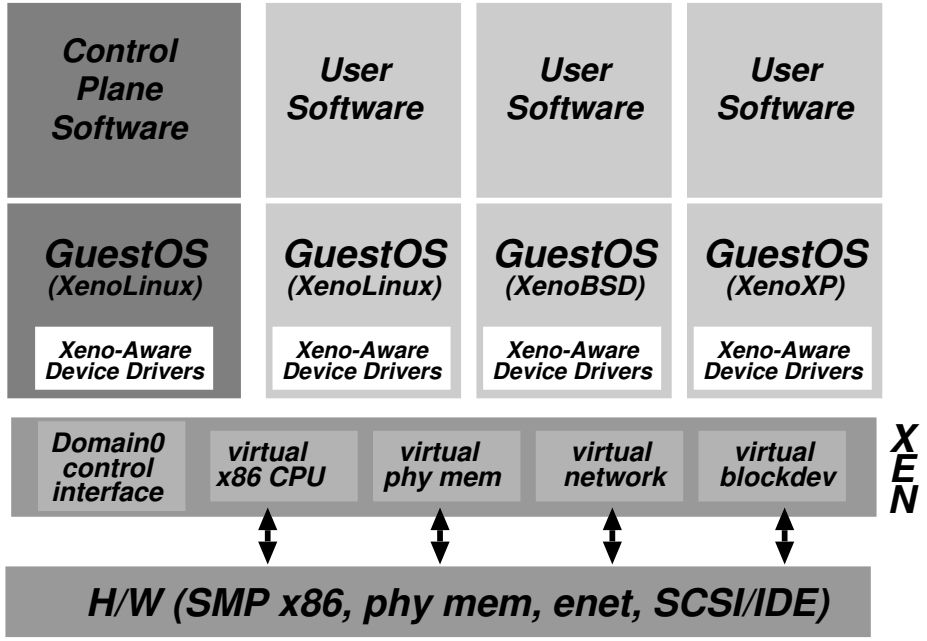
Paravirtualization + Type 1

Type-1: Hypervisor runs directly on hardware
Type-2: Hypervisor runs on host OS

Full virtualization: unmodified guest OS
Paravirtualization: modified guest OS

Xen

- 2003: academic paper
  - "Xen and the Art of Virtualization"
- Paravirtualization
  - Modified guest OS
  - Hypercalls (like system calls but for guest OS) to hypervisor
  - Each guest given 1 or more VCPUs
- Why paravirtualization?
  - Performance improvements
- VMWare, VirtualBox, HyperV

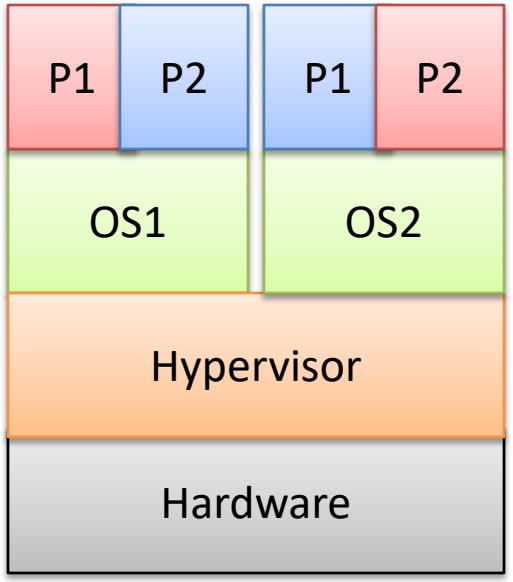


Example VM Use Cases

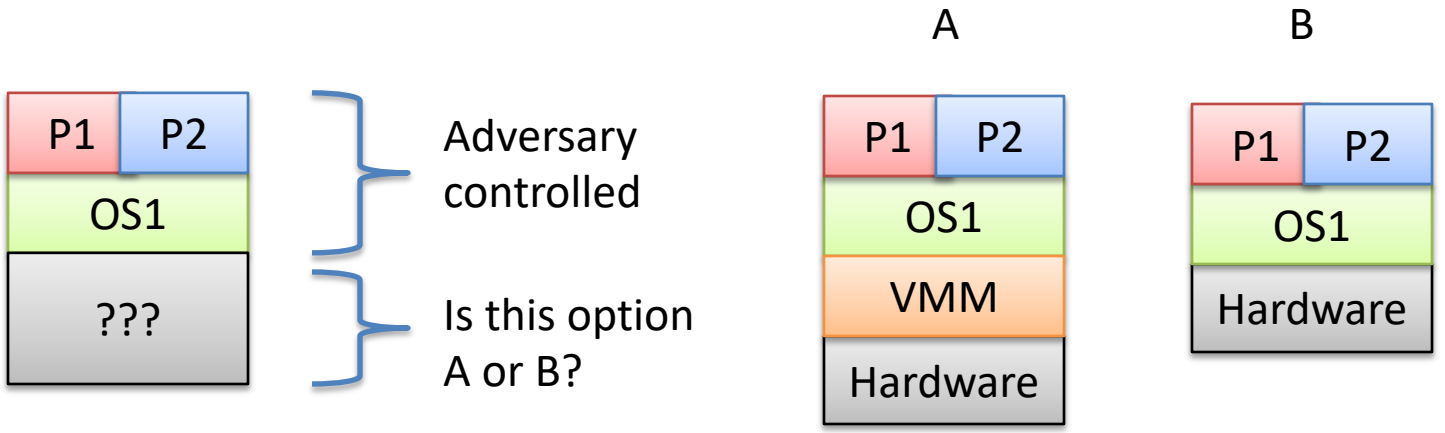
- Legacy support (e.g., IBM VM/370 from 1970s)
- Development
- Server consolidation
- Sandboxing / containment
- Cloud computing Infrastructure-as-a-Service

Study of malware

- Researchers use VMs to study malware
- Example of VM sandboxing
  - Hypervisor must contain malicious code
- How would you evade analysis as a malware writer?
  - split personalities



VMM Transparency

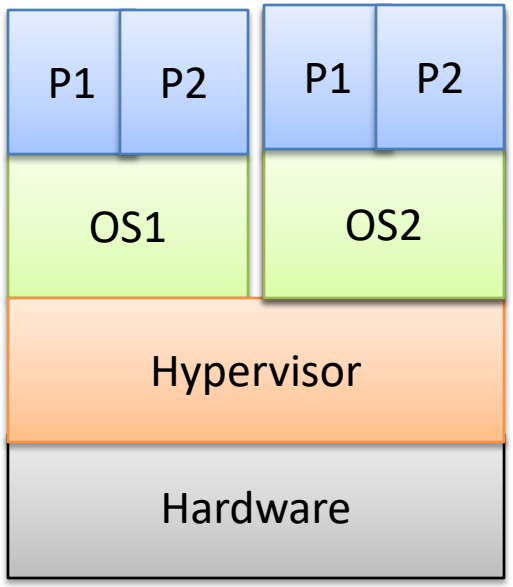


- Adversary can detect if:
  - Paravirtualization
  - Logical discrepancies
    - Expected CPU behavior vs virtualized
    - Red pill (Store Interrupt Descriptor Table instr)
  - Timing discrepancies
    - Slower use of some resources

Garfinkel et al. "Compatibility is not transparency: VMM Detection Myths and Reality"

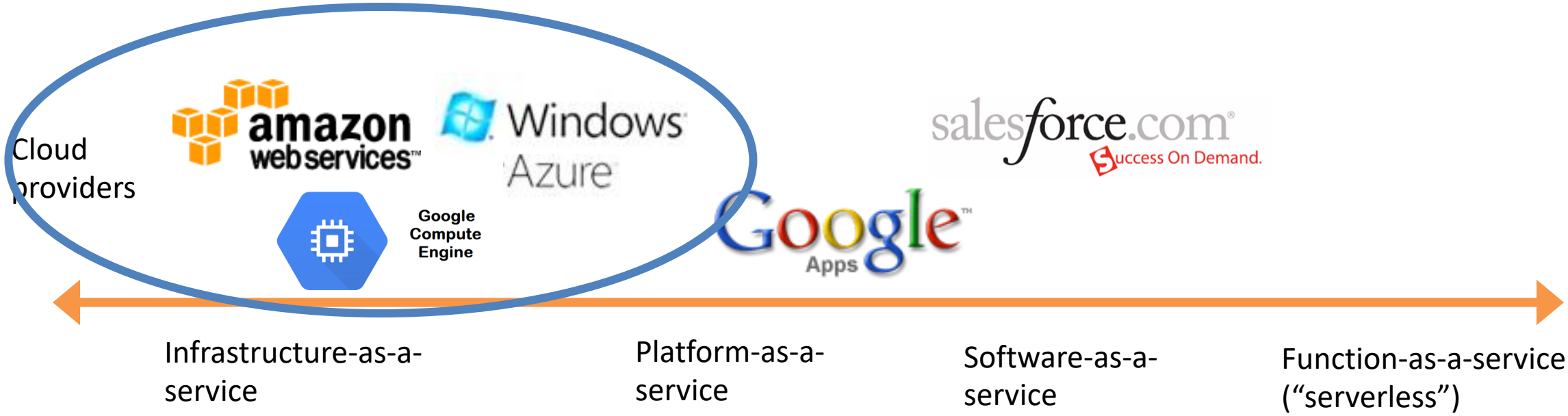
Server consolidation

- Consolidation
  - Use VMs to optimize use of hardware
  - Pack as many VMs onto each server as possible
  - Turn off other servers
- Threat model?
  - Containment
  - Isolation
  - Assume guests are/can be compromised



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 IaaS cloud computing

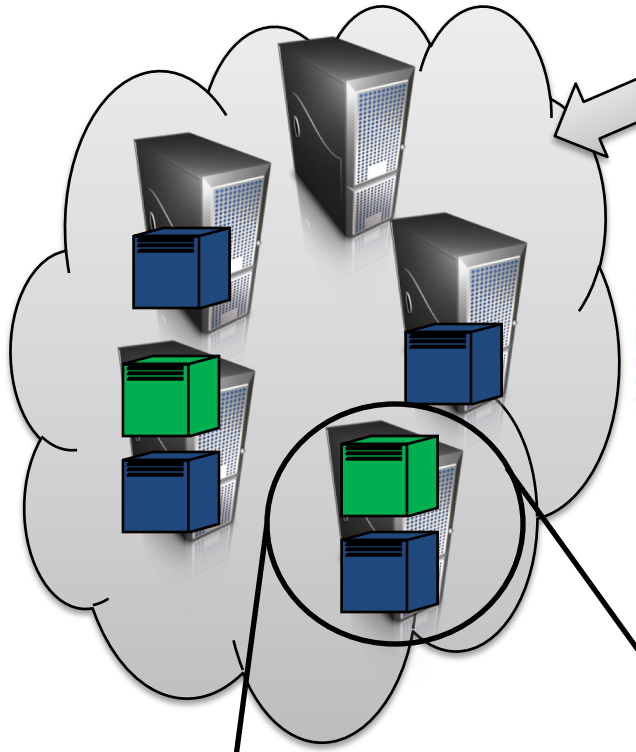
Users run Virtual Machines (VMs) on cloud provider's infrastructure

User A

virtual machines (VMs)

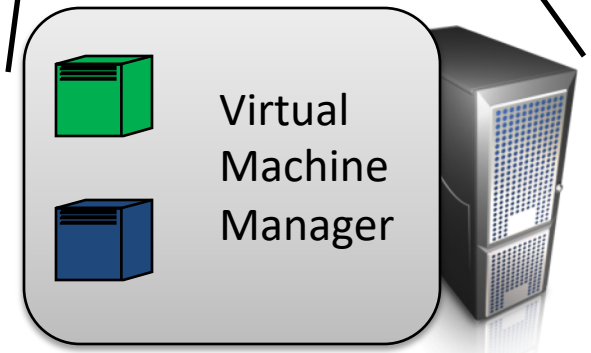
User B

virtual machines (VMs)



Owned/operated by cloud provider

Google Compute Engine

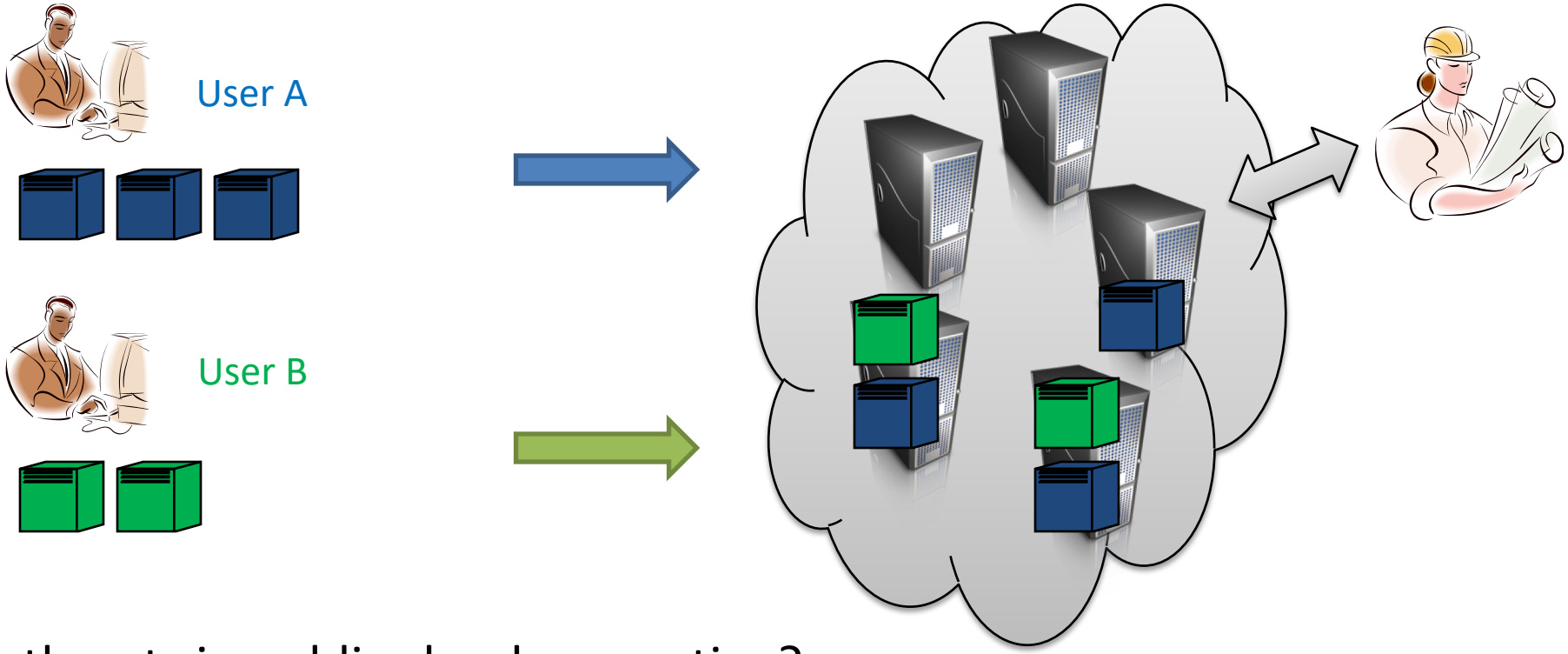


Multitenancy (users share physical resources)

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

To the VM should look like dedicated server

Trust models in public cloud computing



Security threats in public cloud computing?

Provider spying on running VMs / data

External attacks against infrastructure

Cross-user attacks

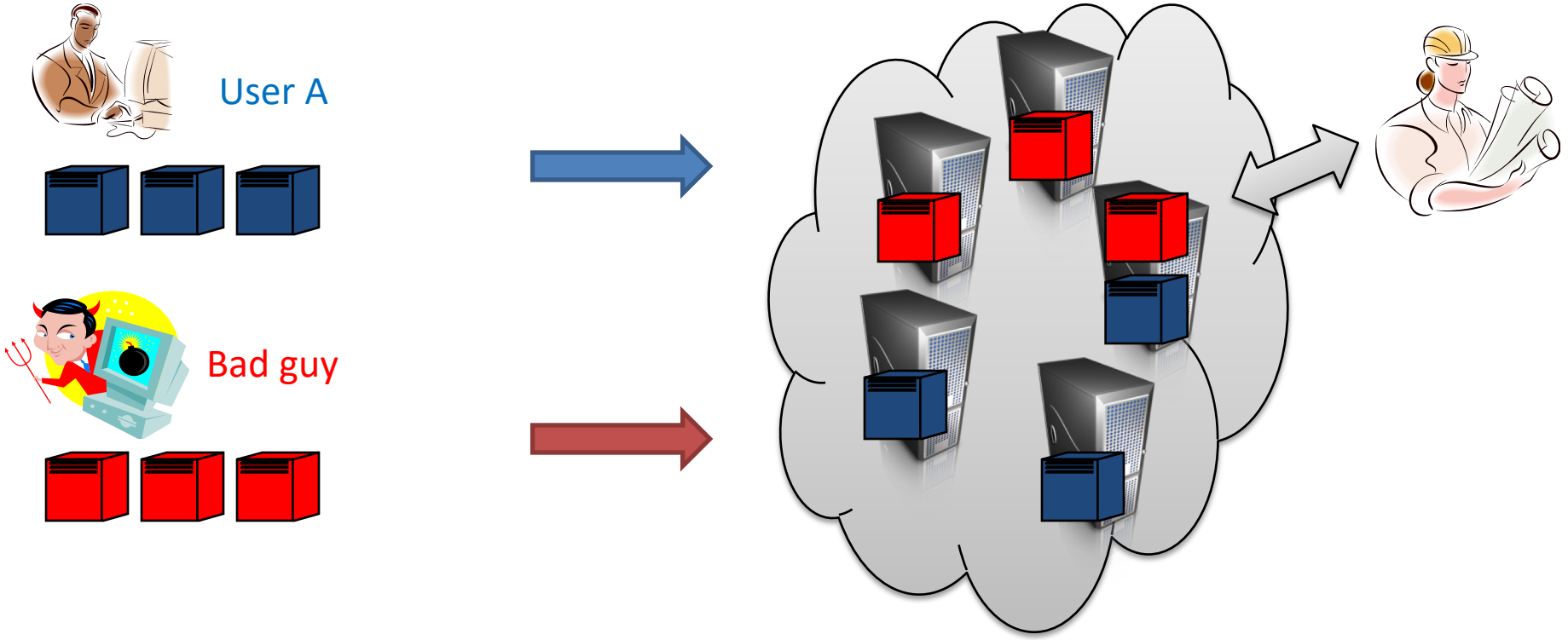
Escape-from-VM

Side-channels attacks

Degradation of service attacks

Resource-stealing attacks

Trust models in public cloud computing



Attacker identifies one or more victims VMs in cloud

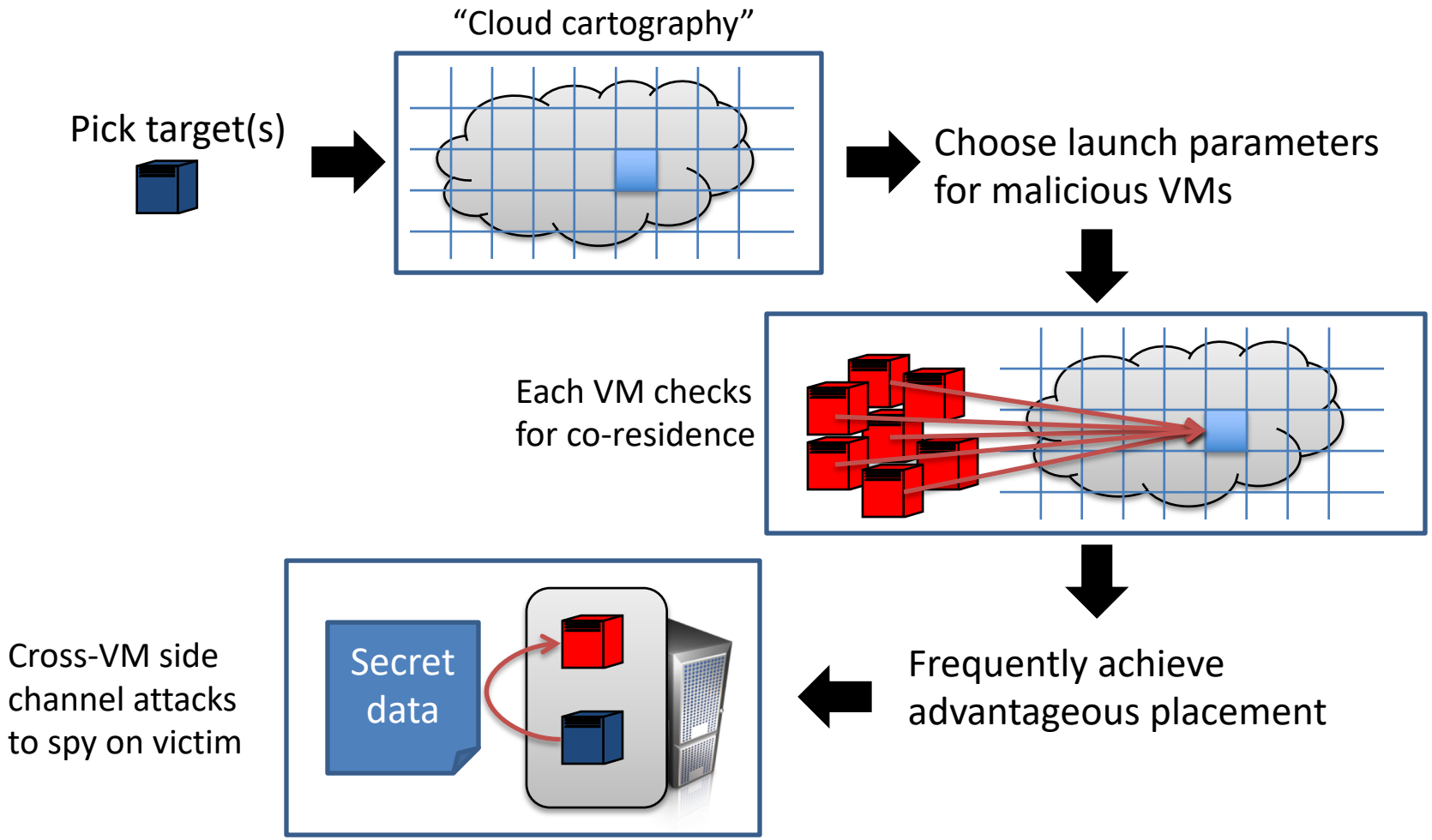
1) Achieve advantageous placement via launching of VM instances

2) Launch attacks using physical proximity

Exploit VMM vulnerability DoS Side-channel attack

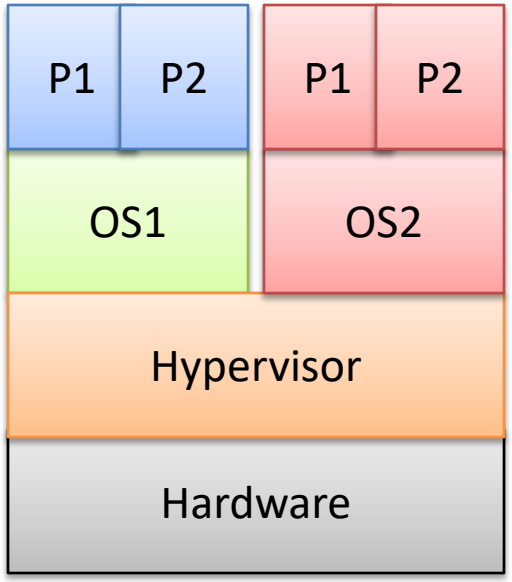
2009 case study with Amazon's EC2

Adversary able to:



Violating containment

- Escape-from-VM
  - Vulnerability in VMM or host OS (e.g., Dom0)
- Memory management flaws in VMM



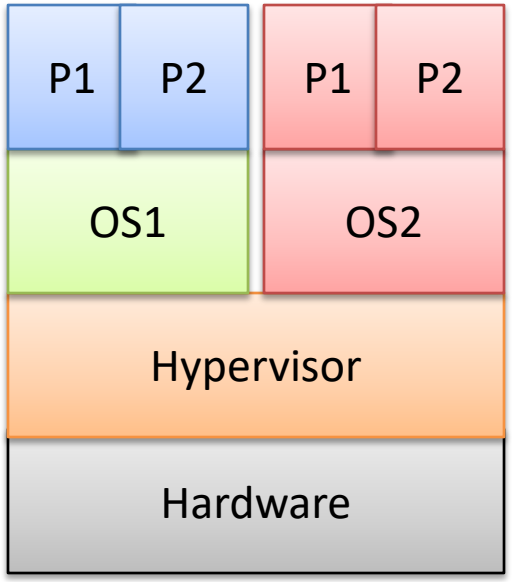
Zero-Day Exploit Published for VM Escape Flaw in VirtualBox

by Lucian Constantin on November 8, 2018

A security researcher disclosed a yet unpatched zero-day vulnerability in the popular VirtualBox virtualization software that can be exploited from a guest operating system to break out of the virtual machine and gain access to the host OS.

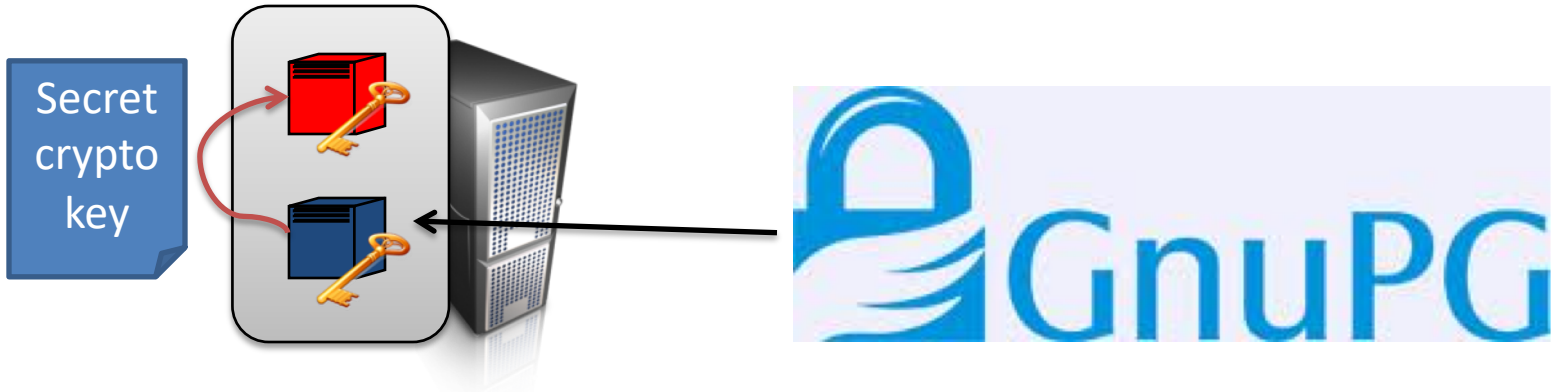
Violating isolation

- Covert channels between VMs circumvent access controls
  - Bugs in VMM
  - Side-effects of resource usage
- Degradation-of-Service attacks
  - Guests might maliciously contend for resources
  - Xen scheduler vulnerability
- Side channels
  - Spy on other guest via shared resources



Cross-VM cryptographic side-channel attacks

[Zhang, Juels, Reiter, R. – CCS `12]



Target is 4096-bit ElGamal secret key e

Modular Exponentiation (x, e, N):

let $e_n \ldots e_1$ be the bits of e

$y \leftarrow 1$

for e_i in $\{e_n \ldots e_1\}$

- $y \leftarrow$ **Square**(y) **(S)**
- $y \leftarrow$ **Reduce**(y, N) **(R)**
- if $e_i = 1$ then
  - $y \leftarrow$ **Multi**(y, x) **(M)**
  - $y \leftarrow$ **Reduce**(y, N) **(R)**

return y // $y = x^e \bmod N$

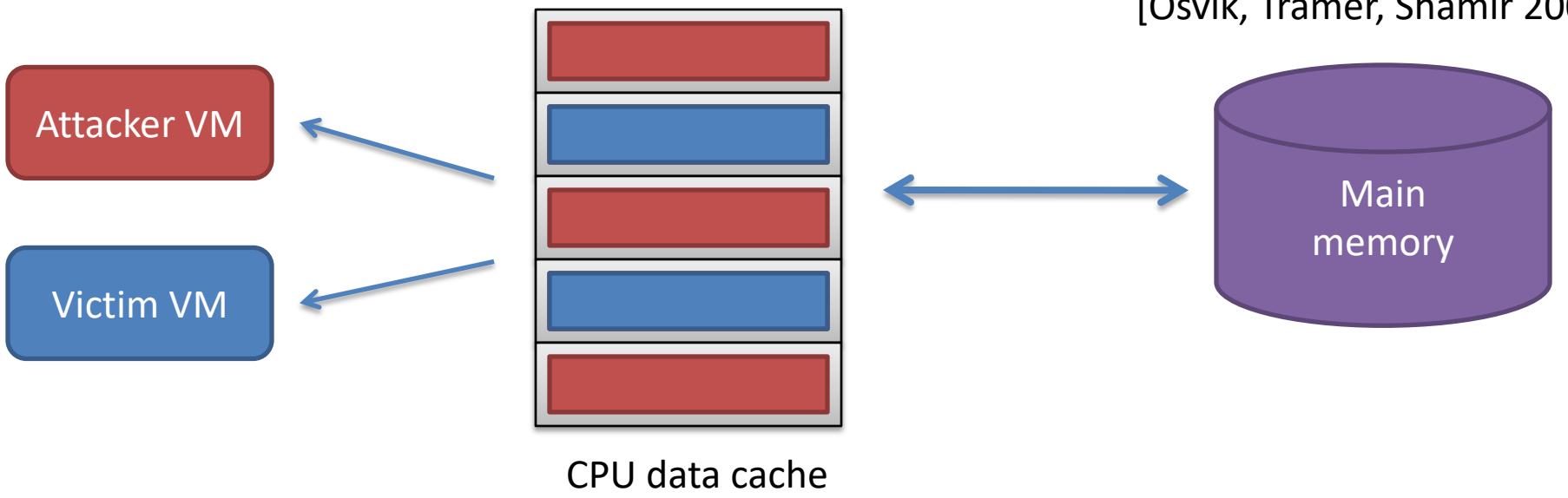
$e_i = 1 \rightarrow$ "SRMR"

$e_i = 0 \rightarrow$ "SR"

Sequence of function calls reveals secret key

Cross-VM side channels using CPU cache contention

Introduced in cross-process setting by [Osvik, Tramer, Shamir 2006]



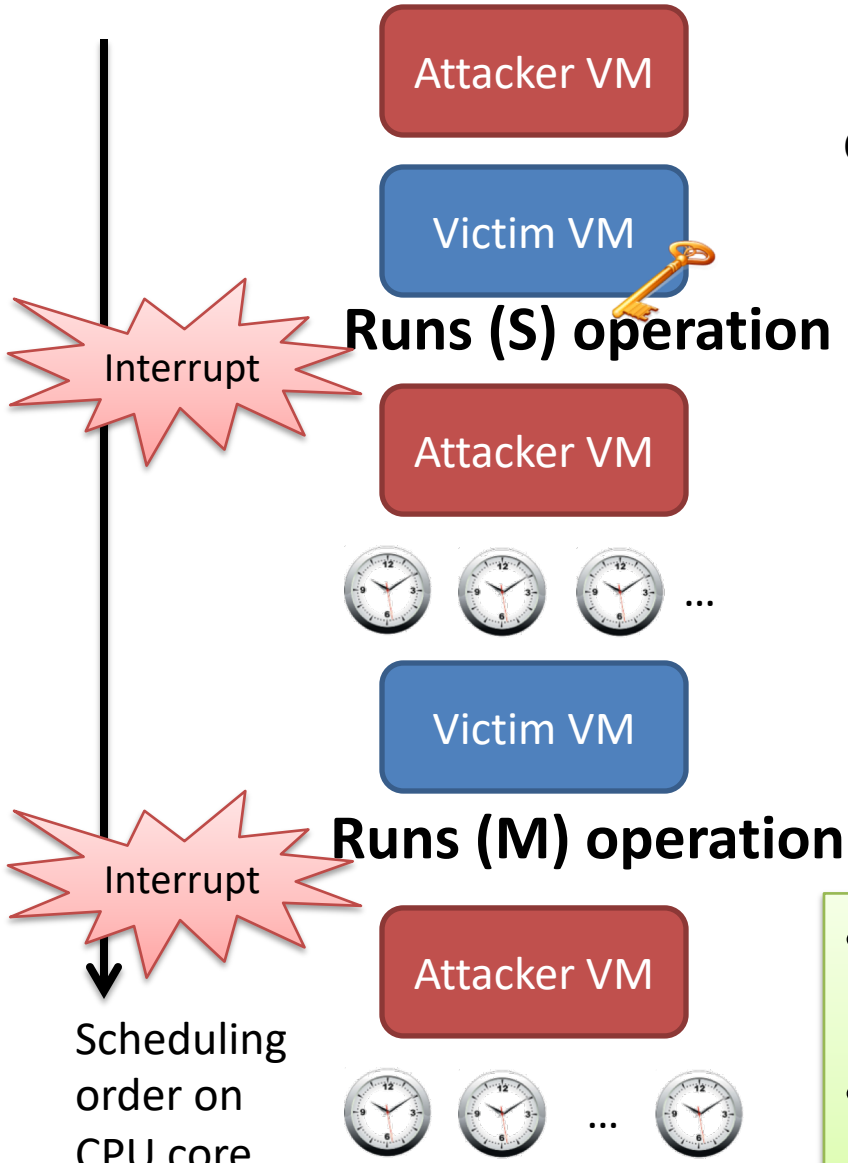
1) Read in a large array (fill CPU cache with attacker data)

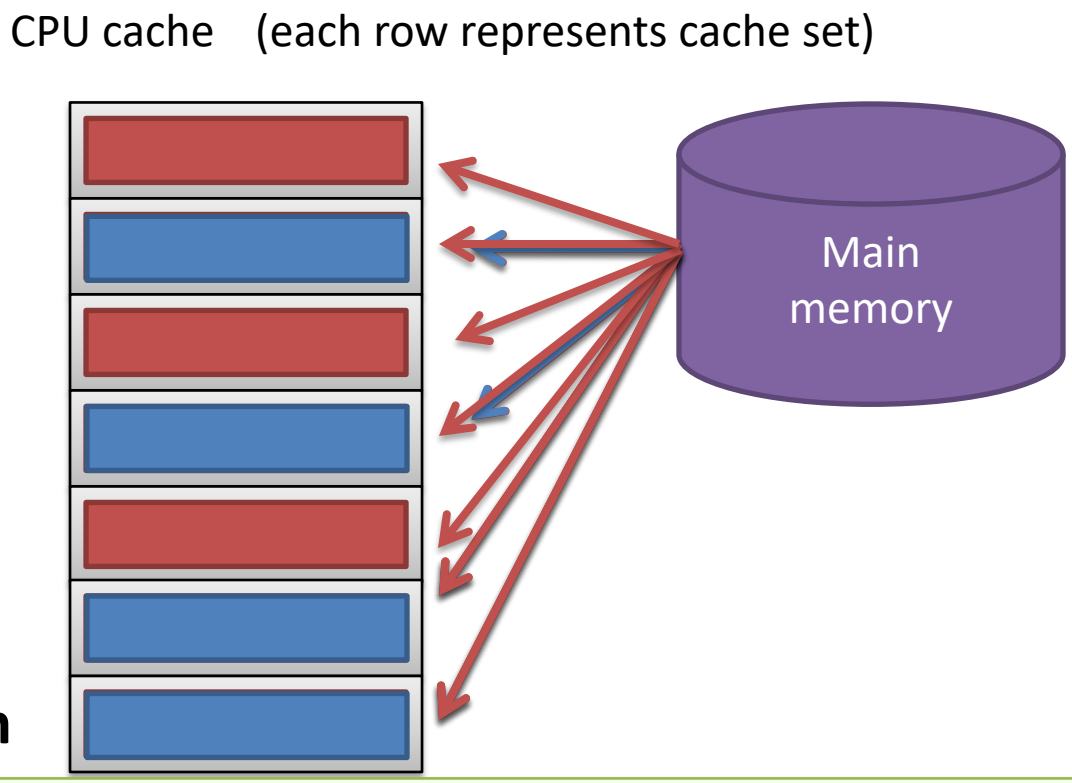
2) Busy loop (allow victim to run)

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

Locations in cache occupied by victim will take longer to load → Information about victim's use of cache revealed to attacker

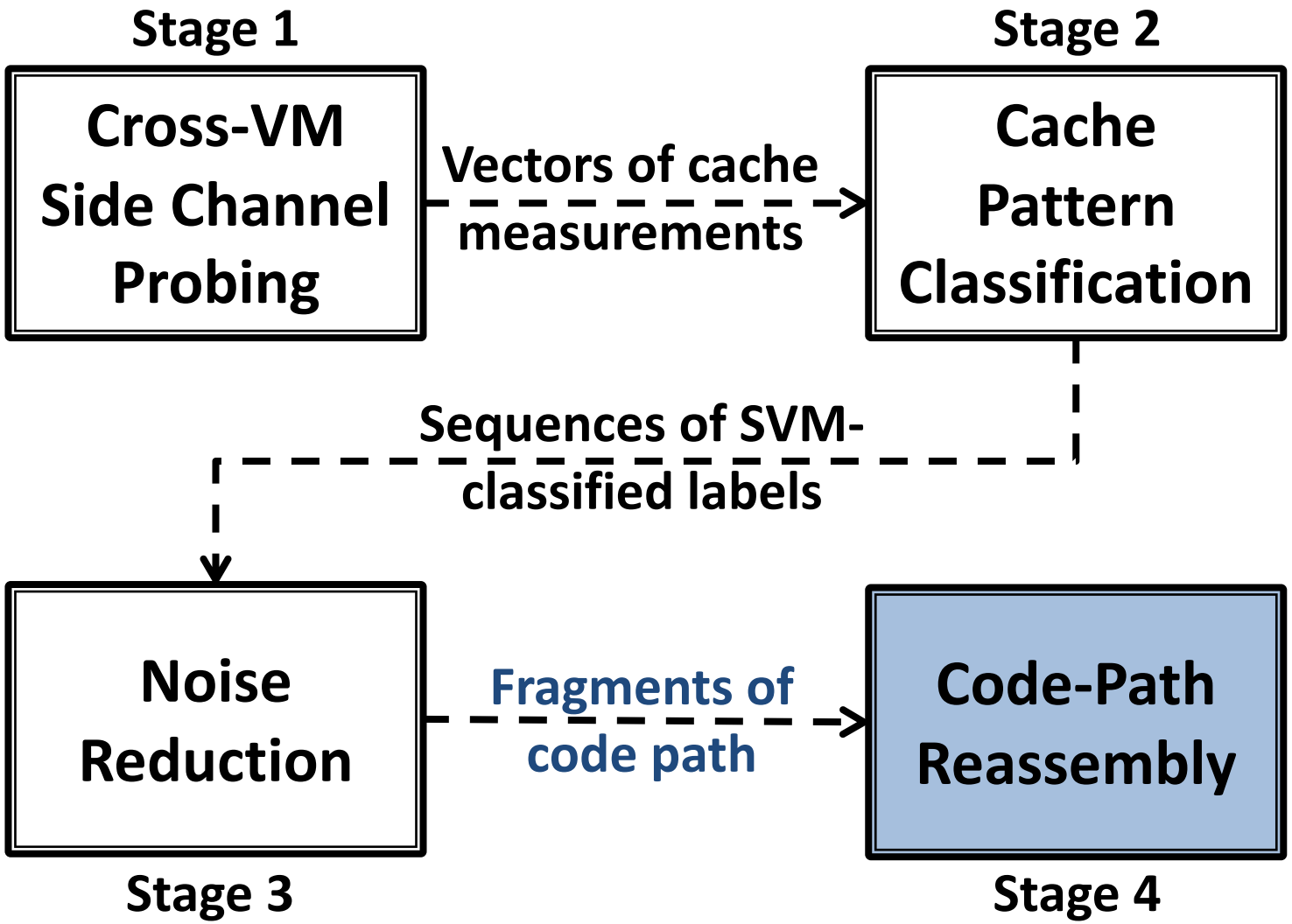
Prime+Probe protocol





- Timings correlated to (distinct) cache usage patterns of S, M operations
- Can spy frequently (every ~16 µs) by exploiting scheduling

Stages of the prototype attack



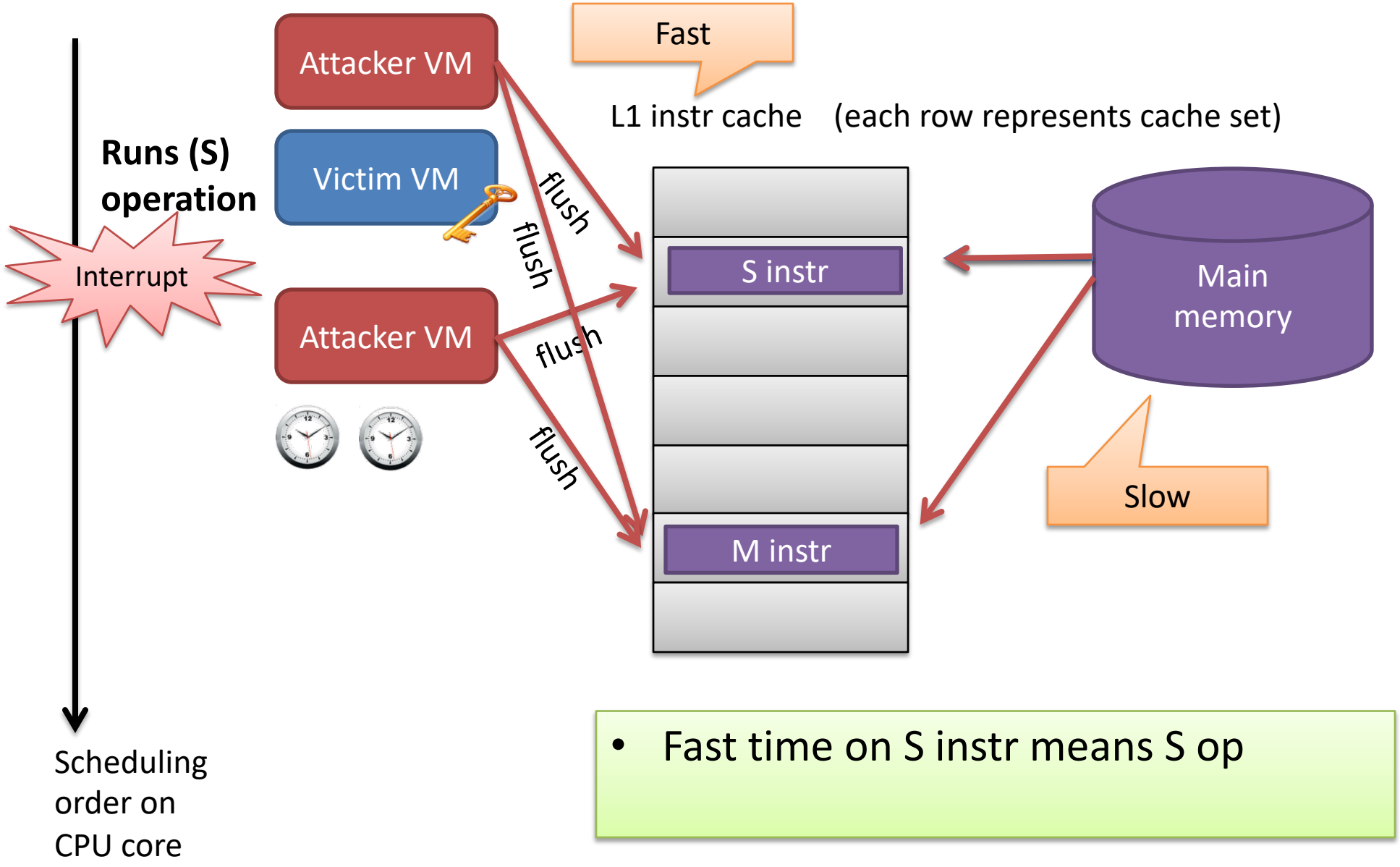
Results

- Setup for in-lab experimentation:
  - Intel Yorkfield processor (4 cores, 32KB L1 instruction cache)
  - Xen + linux + GnuPG + libgcrypt
- **Best result:**
  - 300,000,000 prime-probe results (6 hours)
  - Over 300 key fragments
  - Brute force the secret key in ~9800 guesses
- Not practical to run the attack in deployment settings

Microarchitectural side-channels

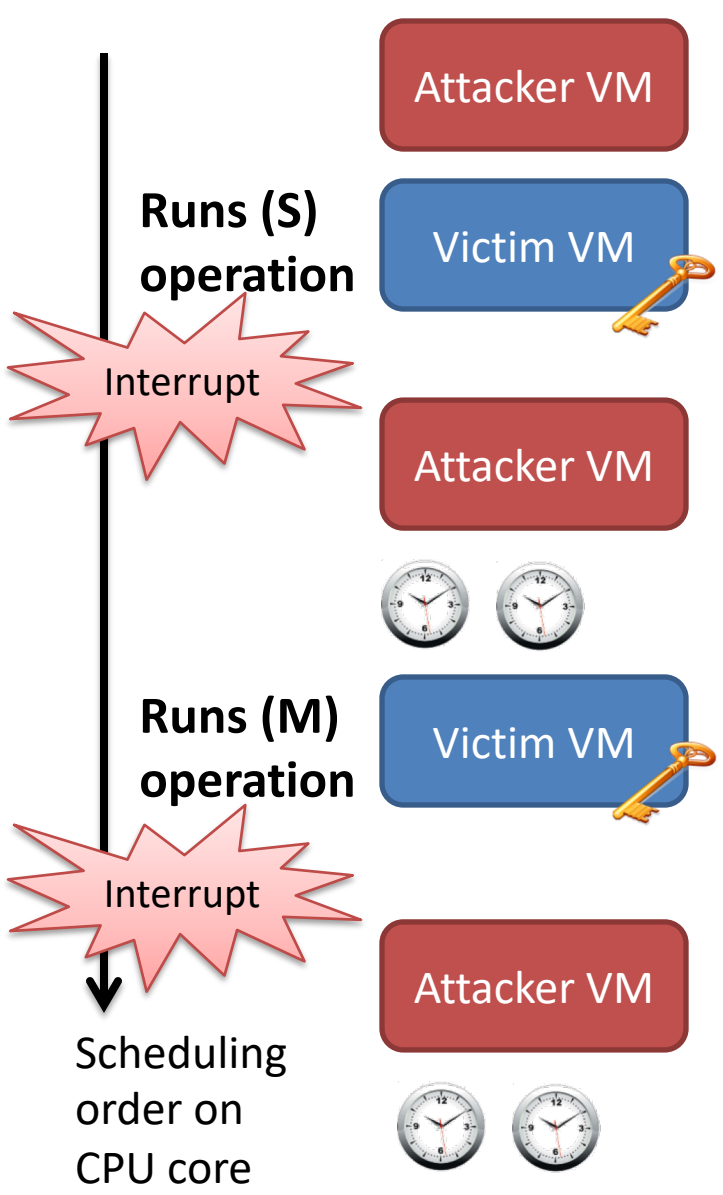
- Lots of research on exploiting microarchitectural side-channels
- State-of-the-art Prime+Probe attacks:
  - Sinan Inci et al. 2016 “Cache Attacks Enable Bulk Key Recovery on the Cloud”
- Flush+Reload (F+R) more robust side-channel in shared memory settings [Yarom, Falkner 2013]
  - Spy process flushes memory shared with victim from caches
  - Idles
  - Times how long it takes to read shared memory

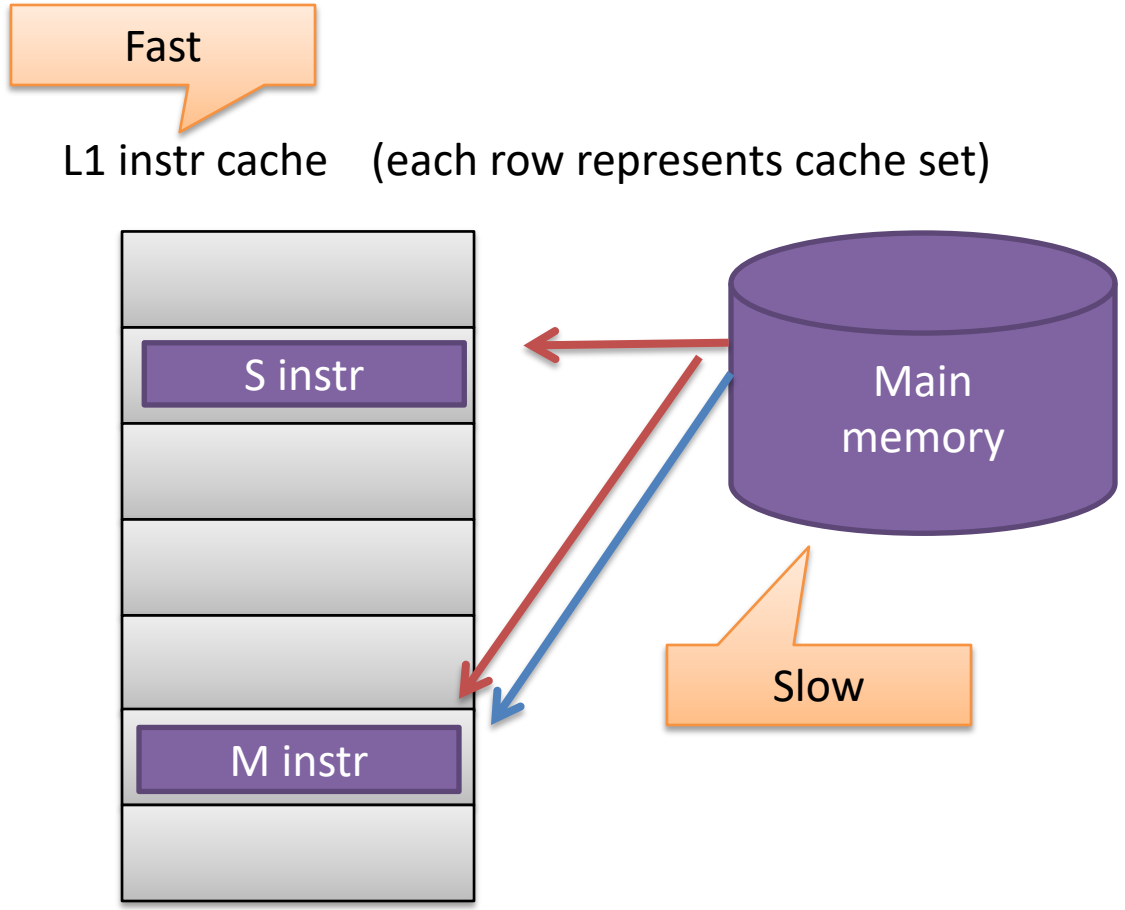
Flush+Reload protocol



- Fast time on S instr means S op

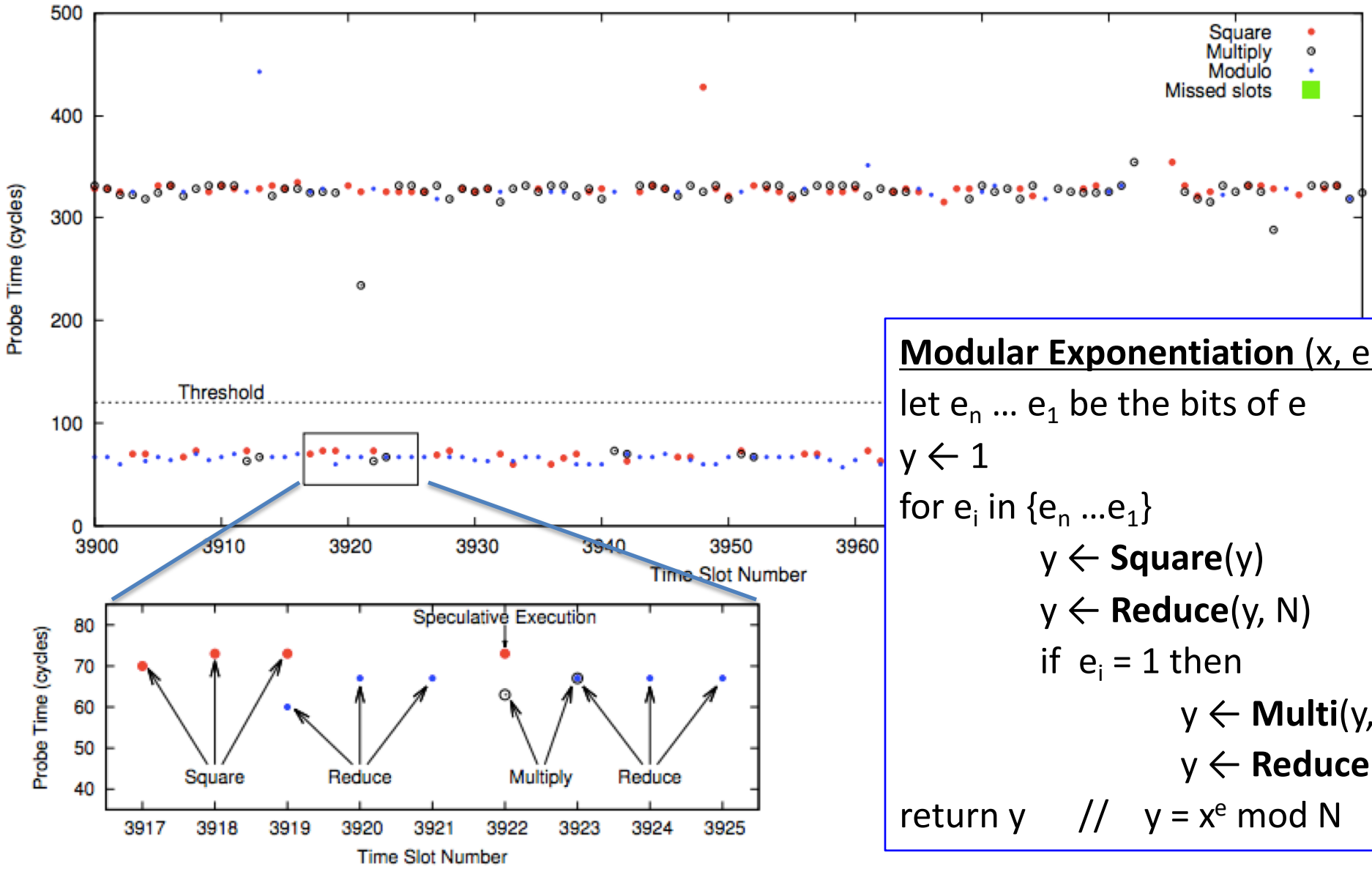
Flush+Reload protocol





- Fast time on S instr means S op
- Fast time on M instr means M op

Attacking Square and Multiply



Modular Exponentiation (x, e, N):

let $e_n \ldots e_1$ be the bits of e

$y \leftarrow 1$

for e_i in $\{e_n \ldots e_1\}$

- $y \leftarrow$ **Square**(y) **(S)**
- $y \leftarrow$ **Reduce**(y, N) **(R)**
- if $e_i = 1$ then
  - $y \leftarrow$ **Multi**(y, x) **(M)**
  - $y \leftarrow$ **Reduce**(y, N) **(R)**

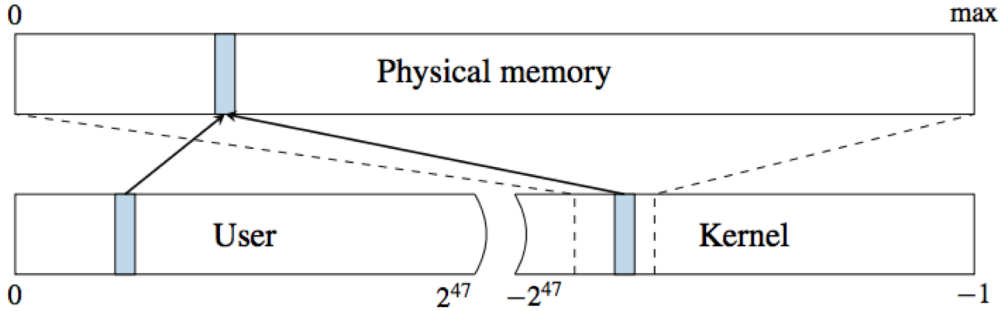
return y // $y = x^e \bmod N$

Flush+Reload widely applicable side-channel

- Useful anywhere code is shared across processes / VMs / containers
- Cross-tenant attacks in platform-as-a-service (PaaS) clouds
  - [Zhang, Juels, Reiter, R. 2014]
- Used as building block for Meltdown, Spectre vulnerabilities

Meltdown

- Speculative execution bugs in Intel x86, ARM, IBM processors + cache-based side-channels (I
  - Allows reading kernel (or hypervisor, other VM) memory



Intel didn't warn US government about CPU security flaws until they were public

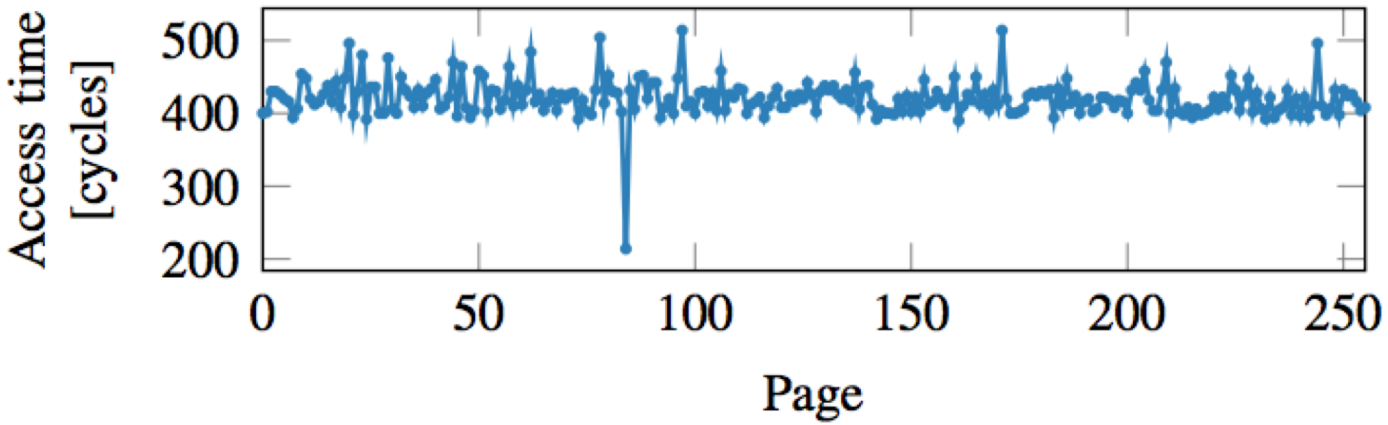
Meltdown and Spectre were kept secret

Researchers find malware samples that exploit Meltdown and Spectre

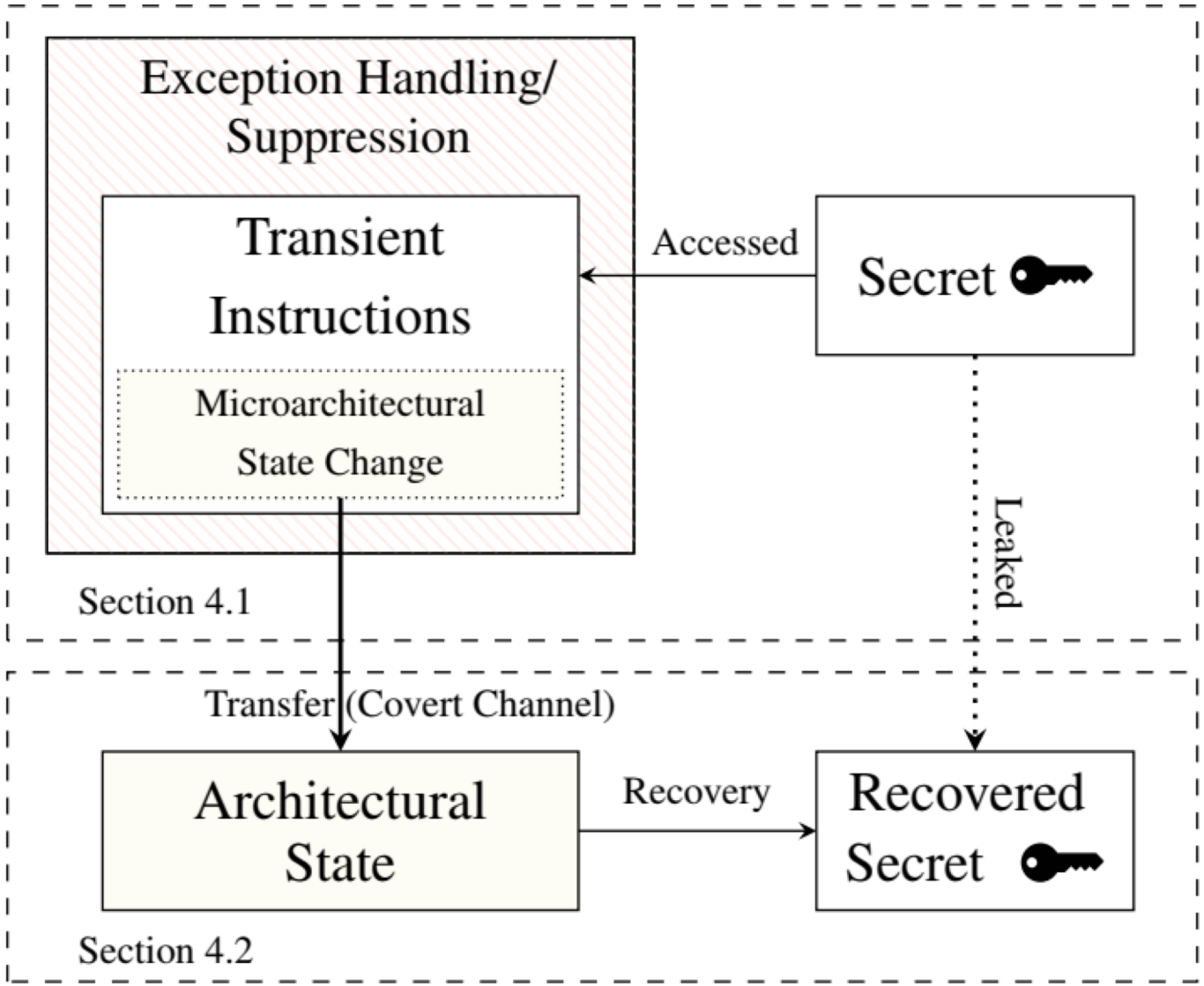
As of Feb. 1, antivirus testing firm AV-TEST had found 139 malware samples that exploit Meltdown and Spectre. Most are not very functional, but that could change.

Meltdown: intuition

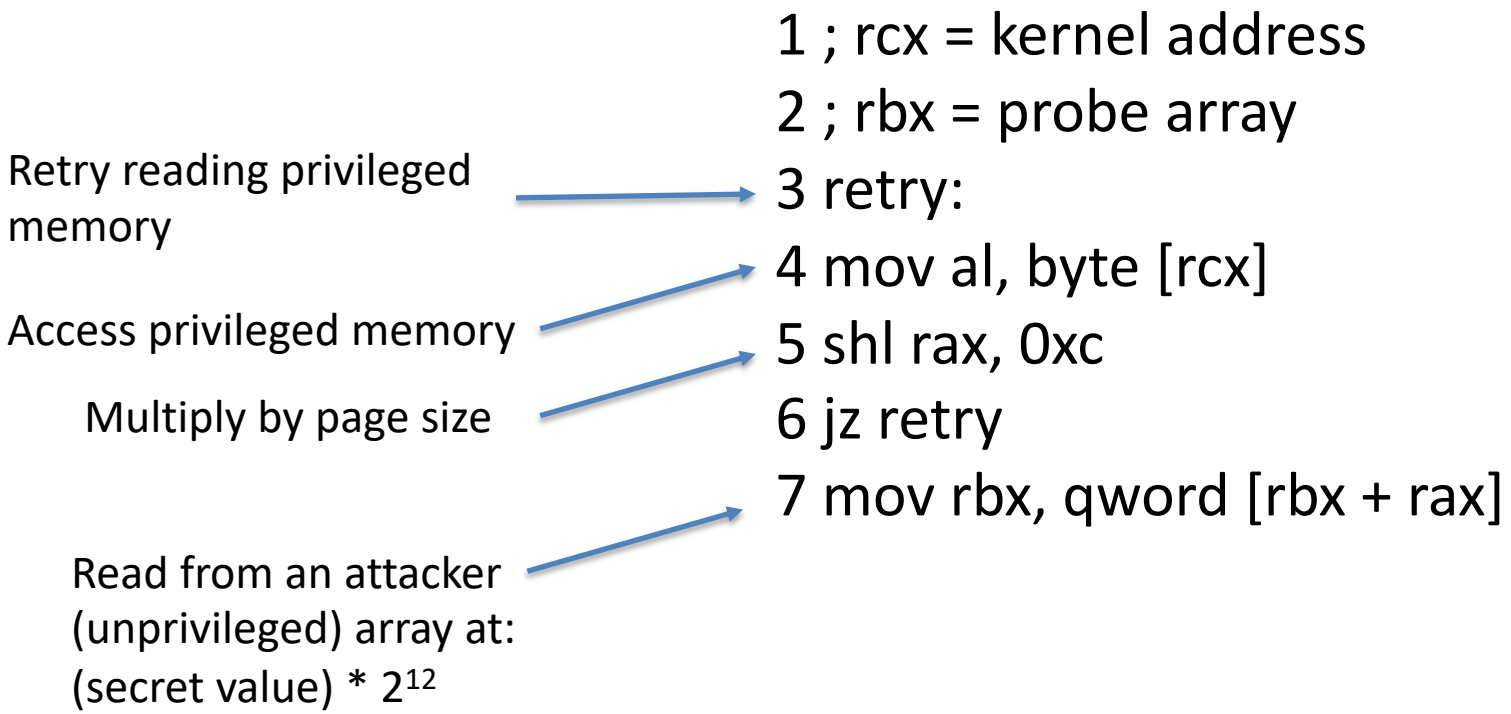
```
1  raise_exception();
2  // the line below is never reached
3  access(probe_array[data * 4096]);
```



Meltdown: design



Meltdown: core spy code



Attacker times accessing [rbx + rax] for different values of rax

When finds one that loads fast, learns sensitive byte

Lessons

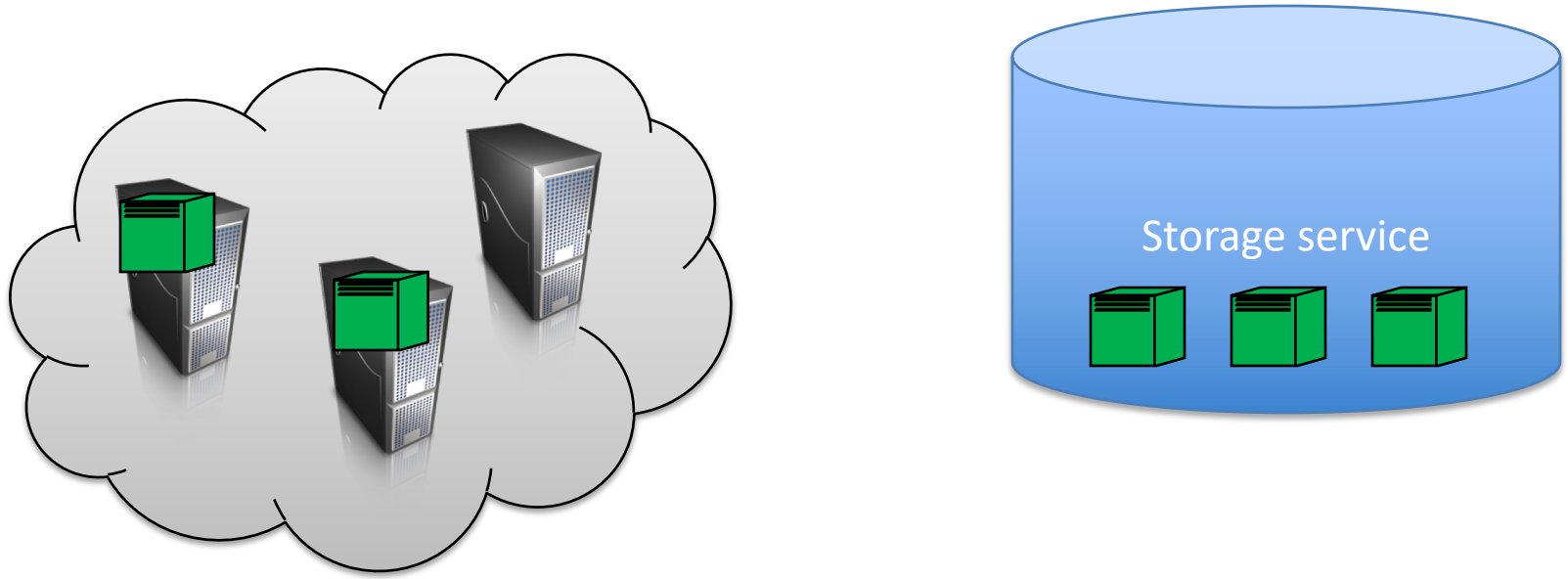
- Isolation/containment useful for defense-in-depth
- Don’t rely *solely* on VMs for:
  - VMM transparency
  - Containment
  - Strong isolation (side channels exist)
  - Securing guest OS and host OS needed for defense-in-depth
- Side-channels are perennial problem
- Microarchitectural vulnerabilities like Meltdown new frontier
  - Spectre, ForeShadow, Fallout, Zombieload, …

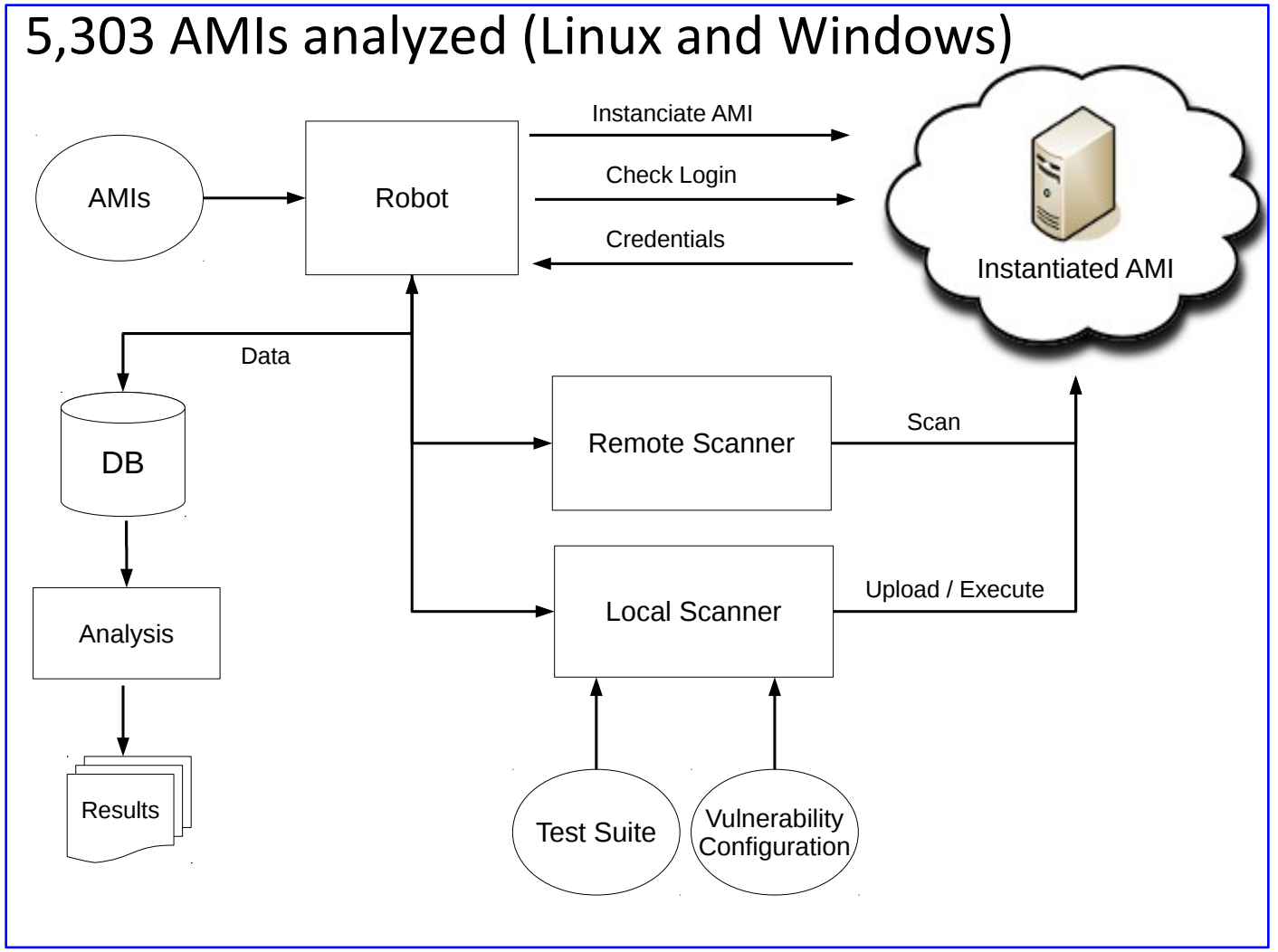
Virtual Machine Management

- Snapshots
  - Volume snapshot / checkpoint
    - persistent storage of VM
    - must boot from storage when resuming snapshot
  - Full snapshot
    - persistent storage and ephemeral storage (memory, register states, caches, etc.)
    - start/resume in between (essentially) arbitrary instructions
- VM image is a file that stores a snapshot

Amazon Machine Images (AMIs)

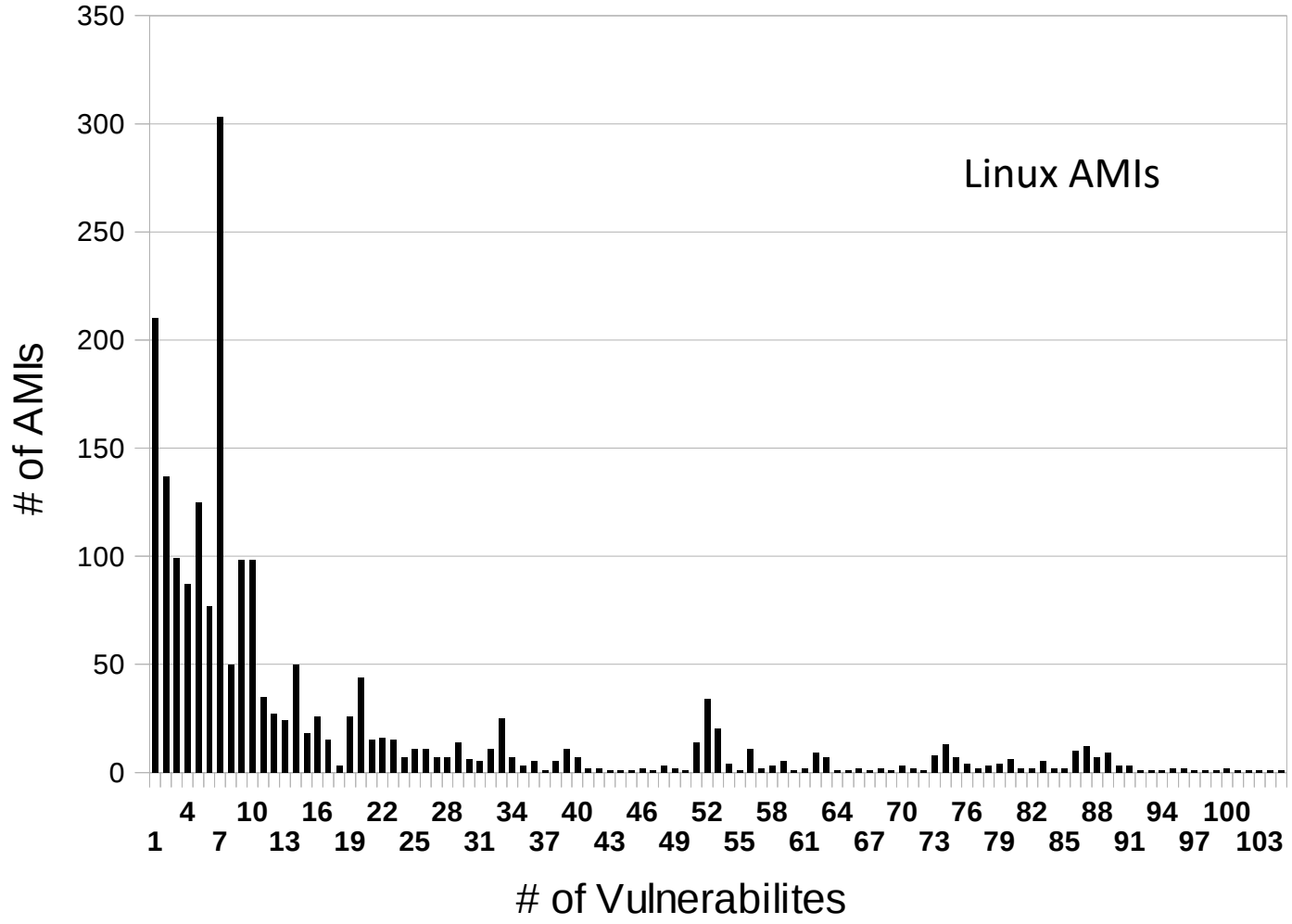
- Users set up volume snapshots / checkpoints that can then be run on the Elastic Compute Cloud (EC2)
- Can be marked as public and anyone can use your AMI





Balduzzi et al. “A Security Analysis of Amazon’s Elastic Compute Cloud Service – Long Version –”, 2011

See also Bugiel et al., “AmazonIA: When Elasticity Snaps Back”, 2011



Also: Malware found on a couple AMIs

Balduzzi et al. “A Security Analysis of Amazon’s Elastic Compute Cloud Service – Long Version –”, 2011

Balduzzi et al. analysis

- Backdoors
  - AMIs include SSH public keys within authorized_keys
  - Password-based backdoors

	East	West	EU	Asia	Total
AMIs (%)	34.8	8.4	9.8	6.3	21.8
With Passwd	67	10	22	2	101
With SSH keys	794	53	86	32	965
With Both	71	6	9	4	90
Superuser Priv.	783	57	105	26	971
User Priv.	149	12	12	12	185

Table 2: Left credentials per AMI

Balduzzi et al. analysis

- Credentials for other systems
  - AWS secret keys (to control EC2 services of an account): 67 found
  - Passwords / secret keys for other systems: 56 found

Finding	Total	Image	Remote
Amazon RDS	4	0	4
dDNS	1	0	1
SQL	7	6	1
MySql	58	45	13
WebApp	3	2	1
VNC	1	1	0
Total	74	54	20

Table 3: Credentials in history files

Balduzzi et al. analysis

- Deleted files
  - One AMI creation method does block-level copying

Type	#
Home files (`/home, /root`)	33,011
Images (min. 800x600)	1,085
Microsoft Office documents	336
Amazon AWS certificates and access keys	293
SSH private keys	232
PGP/GPG private keys	151
PDF documents	141
Password file (`/etc/shadow`)	106

Table 5: Recovered data from deleted files

Response

> “They told me it’s not their concern, they just provide computing power,” Balduzzi says. “It’s like if you upload naked pictures to Facebook. It’s not a good practice, but it’s not Facebook’s problem.”

http://www.forbes.com/sites/andygreenberg/2011/11/08/researchers-find-amazon-cloud-servers-teeming-with-backdoors-and-other-peoples-data/

- Amazon notified customers with vulnerable AMIs
- Made private AMIs of non-responsive customers
- New tutorials for bundling systems
- Working on undelete issues…