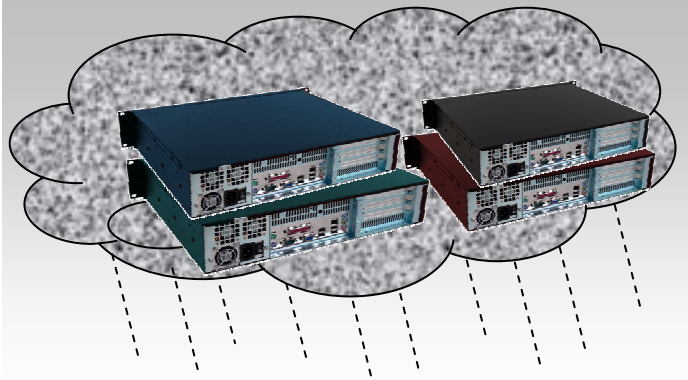


Cross-VM Vulnerabilities in Cloud Computing

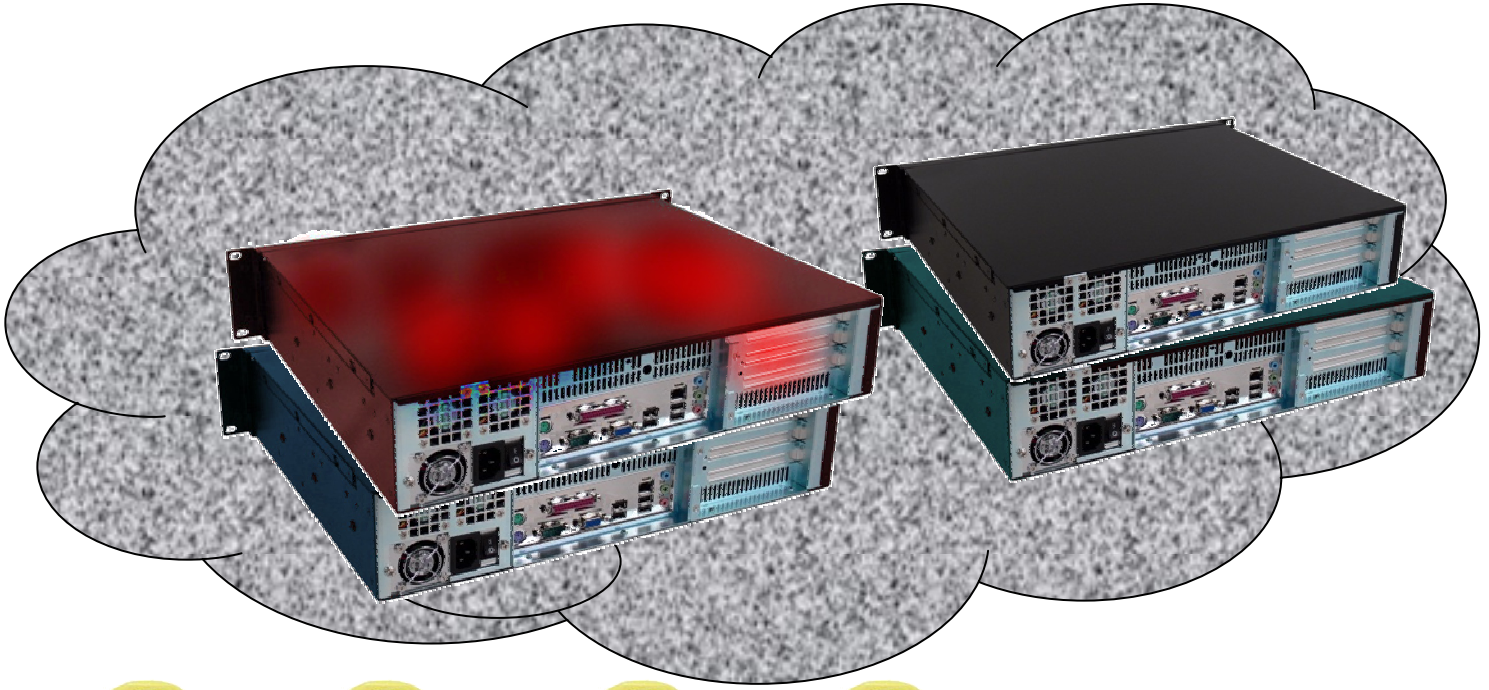
Thomas Ristenpart	UCSD
Eran Tromer	MIT
Stefan Savage	UCSD
Hovav Shacham	UCSD

Infrastructure as a Service

Instant virtual machines

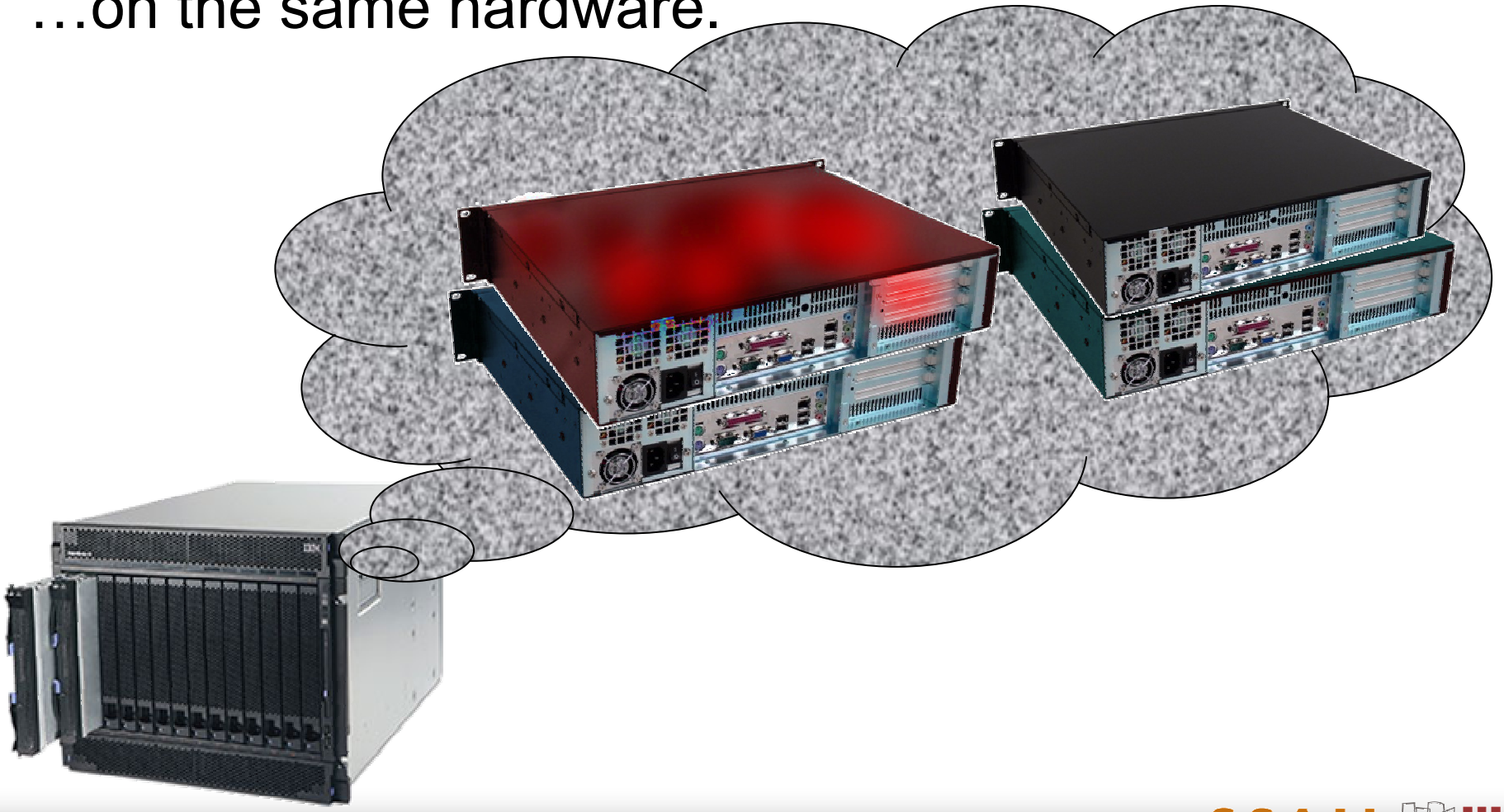
Public Clouds
(Amazon EC2, Microsoft Azure, Rackspace Mosso)

Instant virtual machines
... for anyone



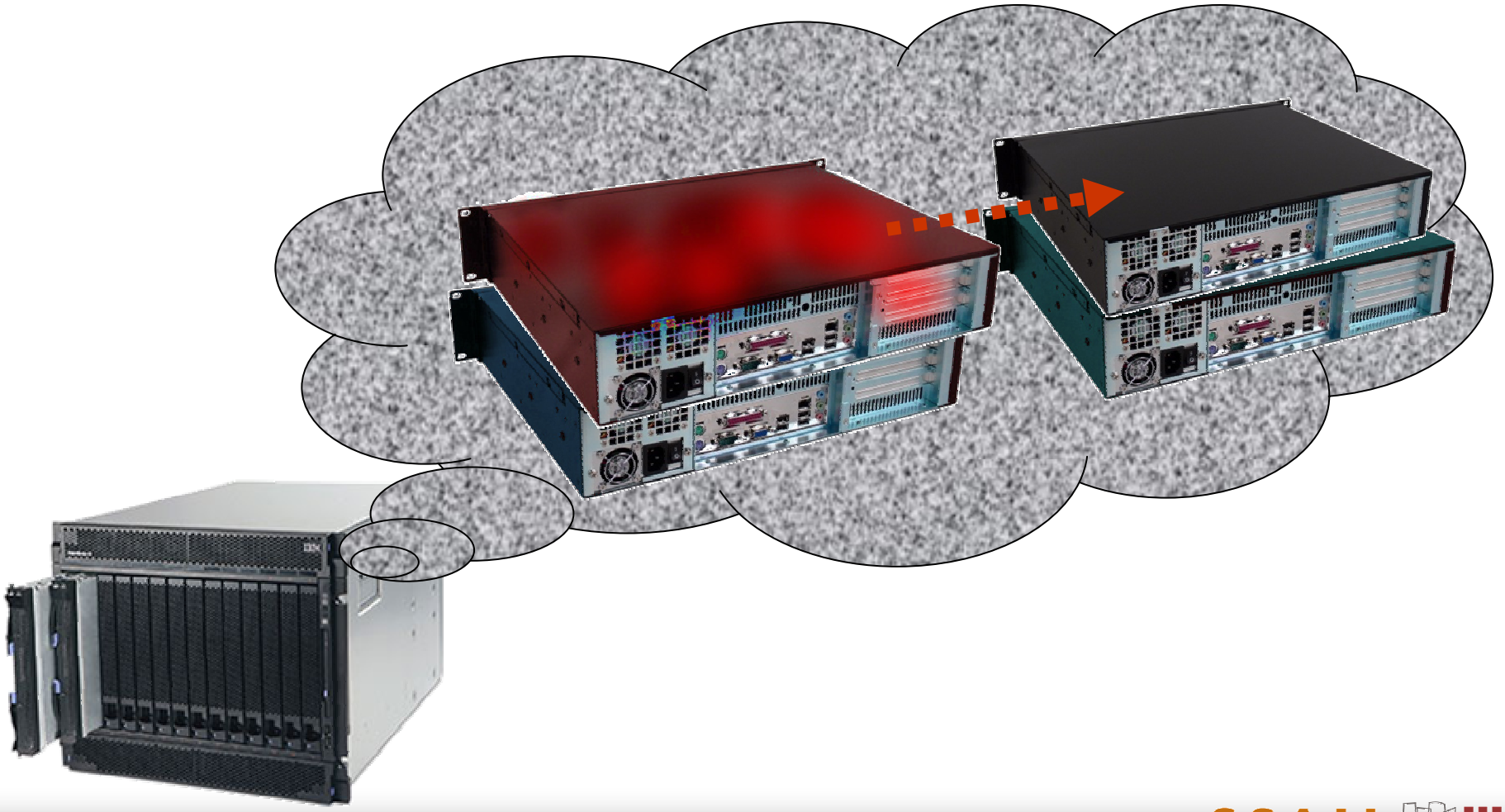
Virtualization

Instant virtual machines
... for anyone
…on the same hardware.



Virtualization

What if someone running on that hardware is malicious?

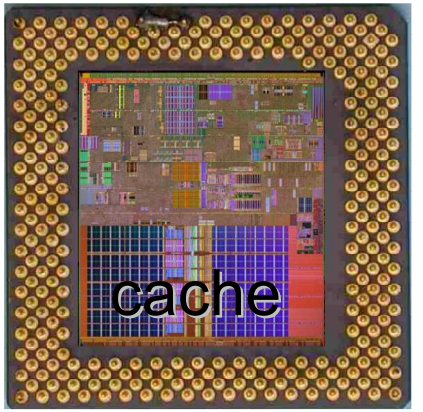


Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.

Attacker app

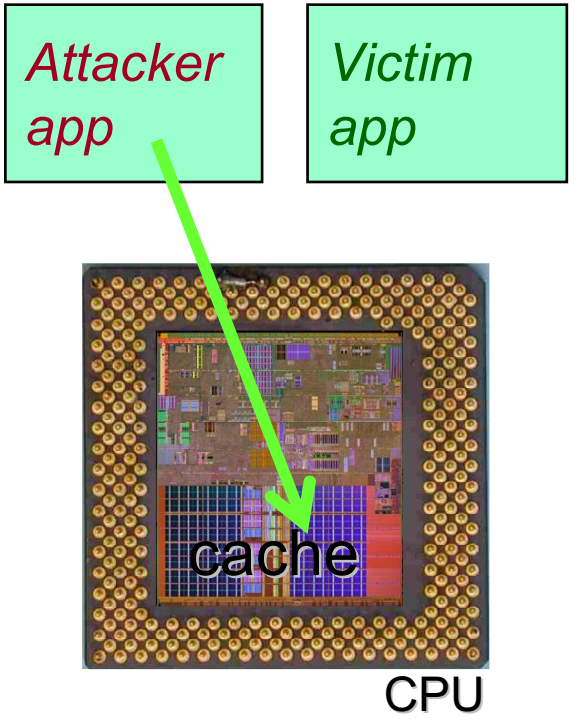
Victim app



CPU

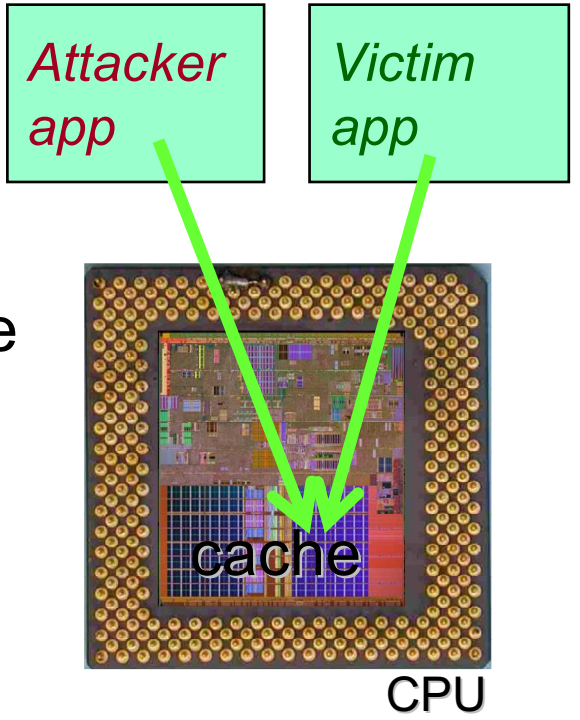
Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.



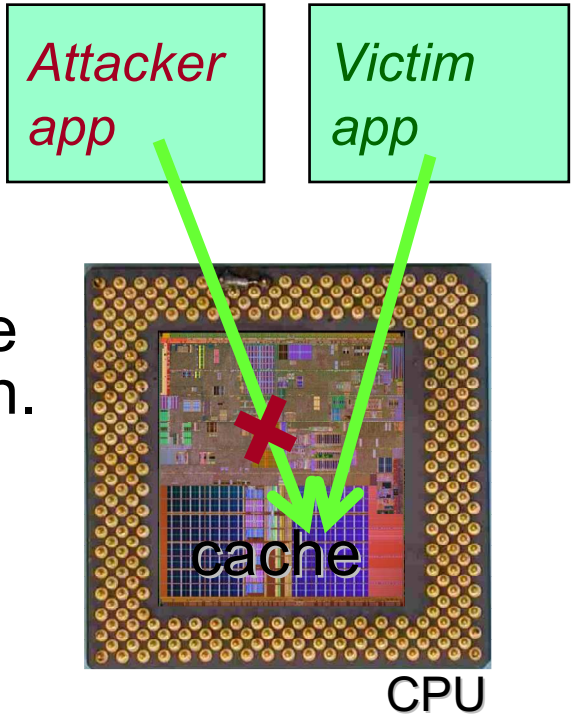
Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.
- If *Victim* also accesses memory, the cache fills up…



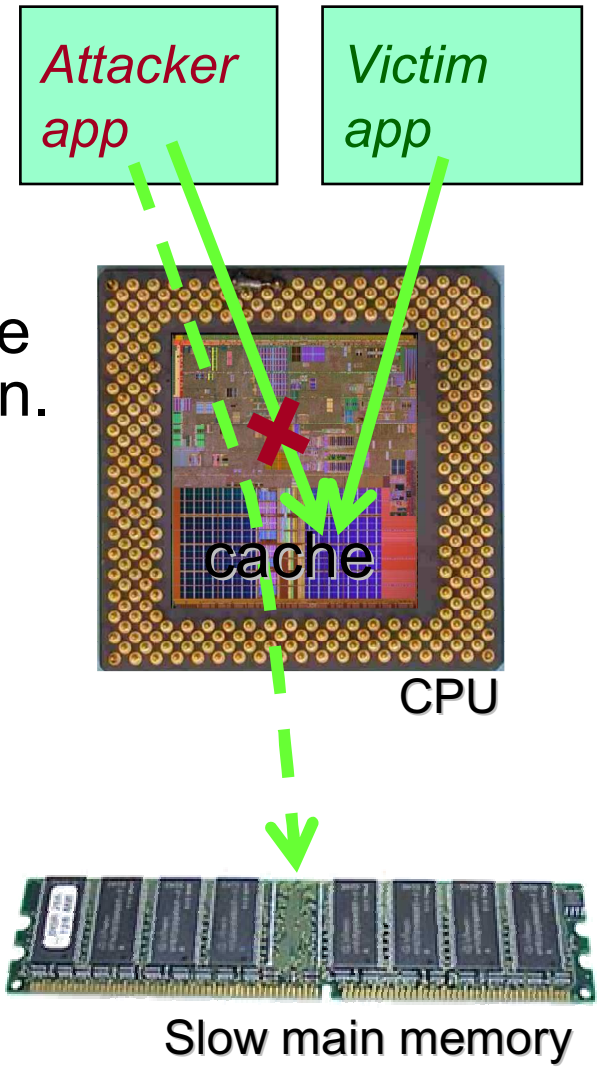
Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.
- If *Victim* also accesses memory, the cache fills up… and *Attacker* notices a slow-down.



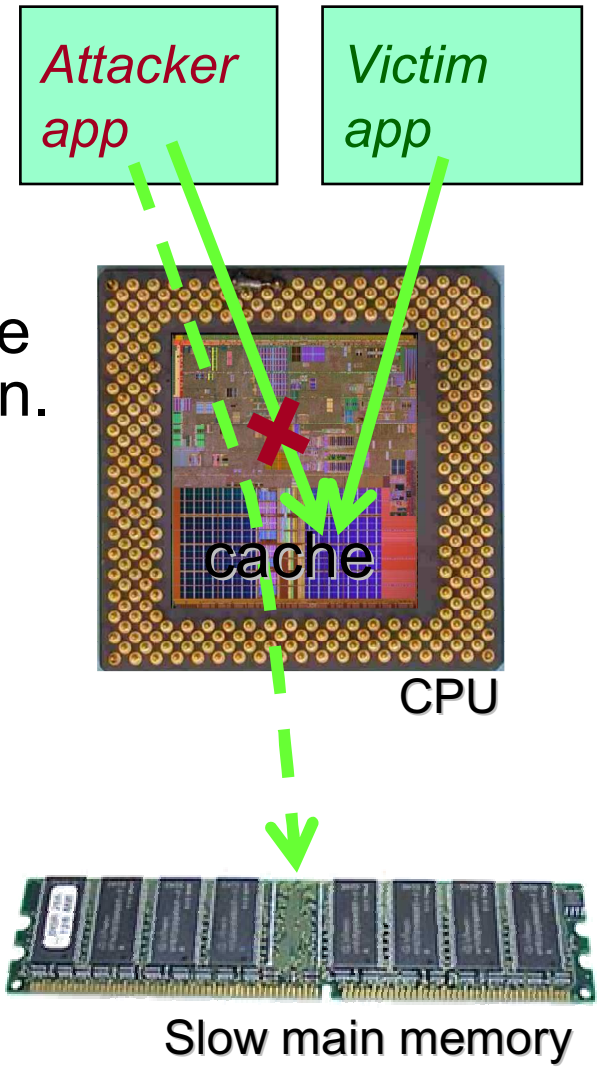
Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.
- If *Victim* also accesses memory, the cache fills up… and *Attacker* notices a slow-down.



Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.
- If *Victim* also accesses memory, the cache fills up… and *Attacker* notices a slow-down.
- From this, *Attacker* can deduce the memory access patterns of *Victim*.

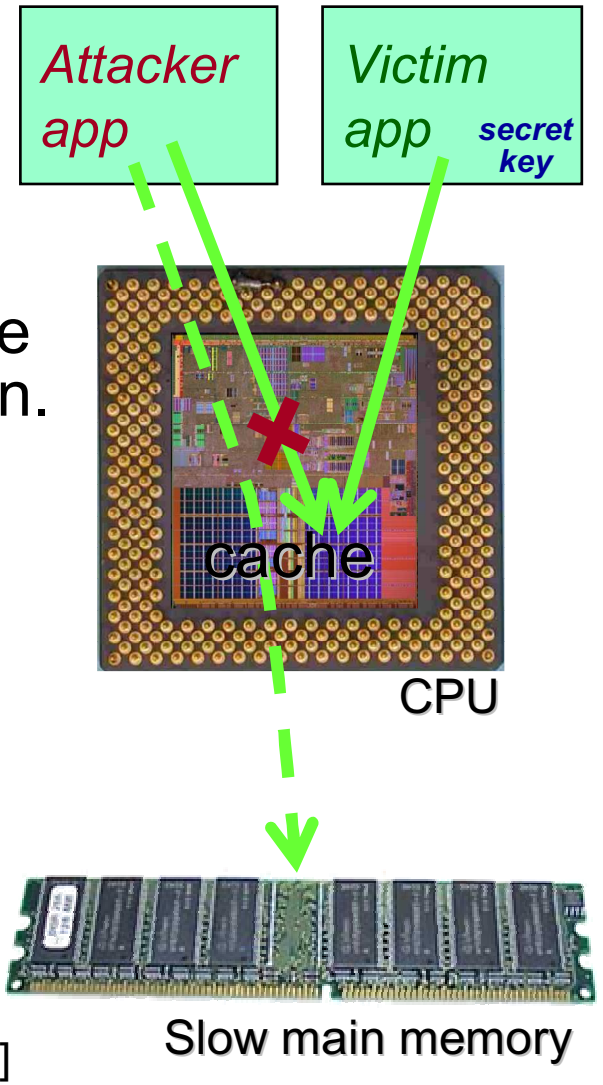


Architectural side-channel attacks
Example: cache attacks

- CPU contains small, fast memory **cache** shared by all applications.
- If only *Attacker* accesses memory, it is served from the fast cache.
- If *Victim* also accesses memory, the cache fills up… and *Attacker* notices a slow-down.
- From this, *Attacker* can deduce the memory access patterns of *Victim*.
- This is sensitive information!

  Example: if *Victim* is performing RSA or AES decryption, the access patterns are determined by the *secret key*.

→ unprivileged *Attacker* can steal AES secret in 65 milliseconds (non-cloud settings) [Tromer Osvik Shamir 09]



Difficulties in cloud computing

- How can the attacker reach a target VM?
- How to exploit it? Practical difficulties:
  - Core migration
  - Extra layer of page-table indirection
  - Coarse hypervisor scheduler
  - Load fluctuations
  - Choice of CPU
- Is the “cloud” really vulnerable?

Cross-VM vulnerabilities in cloud computing

We have demonstrate, using Amazon EC2 as a study case:

- Cloud cartography
  The structure of “cloud” network can be mapped.
- Placement vulnerabilities
  An attacker can place his VM on the same physical hardware as a target VM, for a few dollars.
- Cross-VM vulnerabilities
  Once VMs are co-resident, information can be exfiltrated across VM boundary.

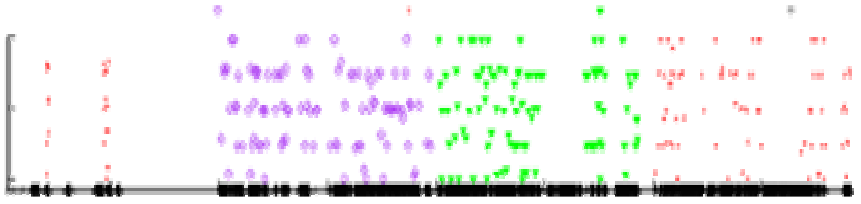
All via standard consumer capabilities, using our own VMs to simulate targets. We believe these vulnerabilities are general and apply to most vendors. No virtual machines or virtualization infrastructure were harmed during this investigation.

Cloud cartography

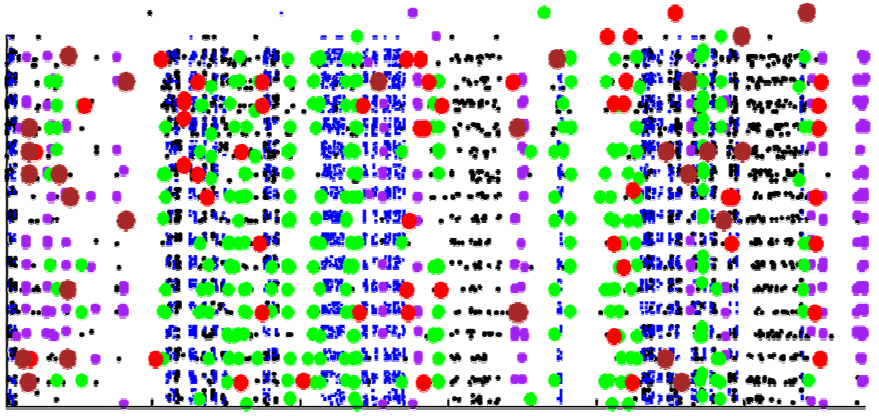
Where in the world is the target VM, and how can I get there?

- On EC2, VMs can be co-resident only if they have identical creation parameters (region, availability zone, instance type).
- The cloud-internal IP addresses assigned to VMs are strongly correlated with their creation parameters.

We mapped out this correlation.



IP address (position) vs. zone (color)



IP address (position) vs. instance type (color)

Detecting co-residence

- EC2-specific:
  - Internal IP address are close
- Xen-specific:
  - Obtain and compare Xen Dom0 address
- Generic:
  - Network latency
  - Cross-VM architectural channels:
    send HTTP requests to target and observe correlation with cache utilization

Achieving co-residence

- Overall strategy:
  - Derive target's creation parameters
  - Create similar VMs until co-residence is detected.
- Improvement:
  - Target fresh (recently-created) instances, exploiting EC2's sequential assignment strategy
  - Conveniently, one can often *trigger* new creation of new VMs by the victim, by inducing load (e.g., RightScale).
- Success in hitting a given (fresh) target: ~40% for a few dollars

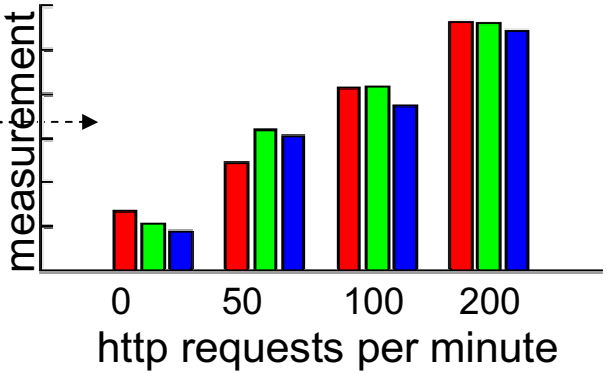
  Reliable across EC2 zones, accounts and times of day.

Exploiting co-residence: cross-VM attacks

- Demonstrated:
  - Measuring VMs load (average/transient)
  - Estimating web server traffic
  - Robust cross-VM covert channel
  - Detecting keystroke timing in an SSH session across VMs

    (on a similarly-configured Xen box)

  → keystroke recovery [Song Wagner Tian 01]



- Work in progress:
  - Key leakage
  - Data exfiltration

* * *

http://csail.mit.edu/~tromer/cloudsec

To appear in ACM CCS'09.