VM Security in Cloud Computing

Dr. Amit Praseed

Managing Compute Assets

- Compute assets typically take data, process it, and do something with the results.
  - For example, a very simple compute resource might take data from a database and send it to a web browser on request, or send it to a business partner, or combine it with data in another database.
- Compute resources may also store data, particularly temporary data.
  - With some types of regulated data, it may be necessary to ensure that you're tracking every place that data could be

Detecting Virtualization

- Can a malicious entity detect whether it is operating in a virtualized environment?
  - Malware was designed to lay dormant on virtualized environments
  - With the spread of virtualization, malware is now configured to detect virtualization and exploit hypervisor specific vulnerabilities
    - VMWare “get version” command
    - Resource discrepancies
    - Timing discrepancies

VM Security

- VMs share the same *physical* system with other cloud customers.
  - "noisy neighbor" problems : using up all of the processor time, network bandwidth, or storage bandwidth
- Two types of attacks against VMs
  - Hypervisor Breakout / VM Escape
  - Side Channel Attacks

Hypervisor Breakout

- An attacker runs code on a virtual machine that allows an operating system running inside the hypervisor to break out and interact directly with it.
- This type of attack could allow the attacker access to the host operating system as well as all virtual machines running on that host.
- Several techniques are available for launching these attacks

Side Channels in the Cloud

- Hypervisors are meant to isolate virtual machines
- Side channel attacks use a medium that is not explicitly meant for communication
  - Eg: CPU data caches
- Co-residence is often an essential condition for executing these attacks
- Verifying co-residence
  - Same Dom0 IP
  - Small packet RTT
  - Numerically close IP addresses

Side Channels in the Cloud

- A malicious instance can utilize side channels to learn information about co-resident instances
  - time-shared caches allow an attacker to measure when other instances are experiencing computational load
  - any physical machine resources multiplexed between the attacker and target forms a potentially useful channel: network access, CPU branch predictors and instruction cache, DRAM memory bus, CPU pipelines , scheduling of CPU cores and timeslices, disk access etc

Side Channels in the Cloud

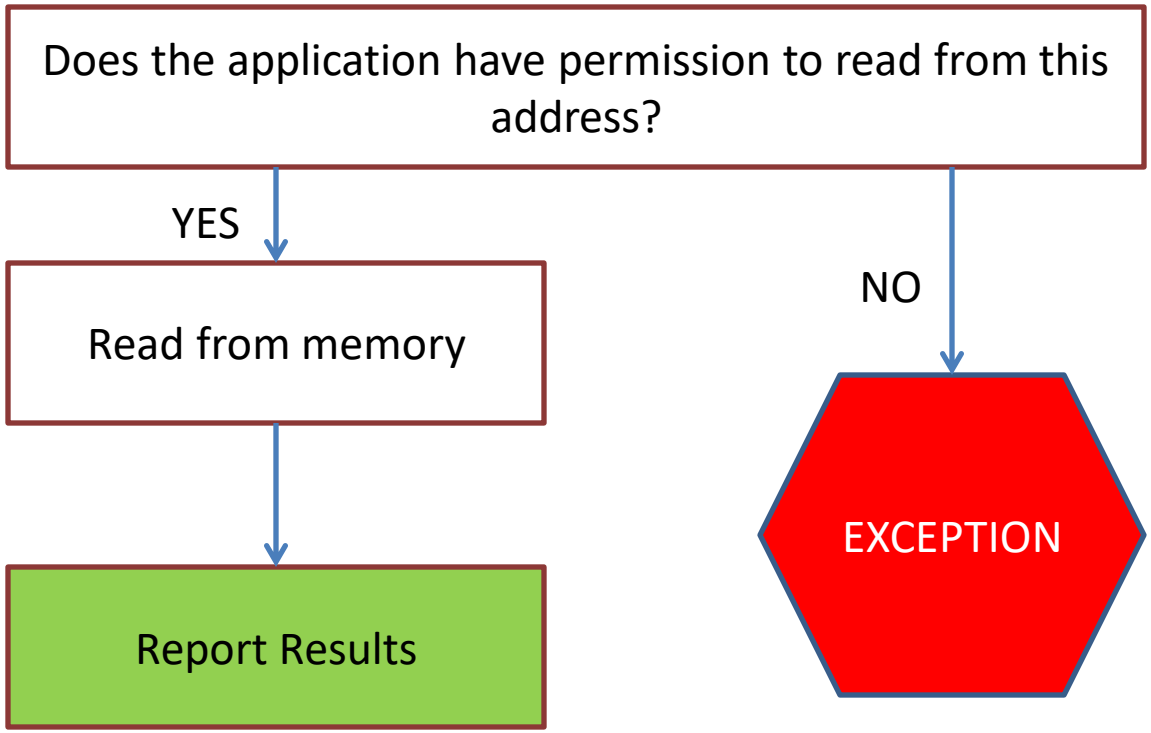
- An attacking instance can measure the utilization of CPU caches on its physical machine.
  - These measurements can be used to estimate the current load of the machine
  - Variants of Flush + Reload
    - Flush the cache lines
    - Wait for victim program to run
    - Access the cache and check the timing
      - Less time ➔ victim accessed the cache line
      - More time ➔ victim did not access the cache line

Side Channel Attacks

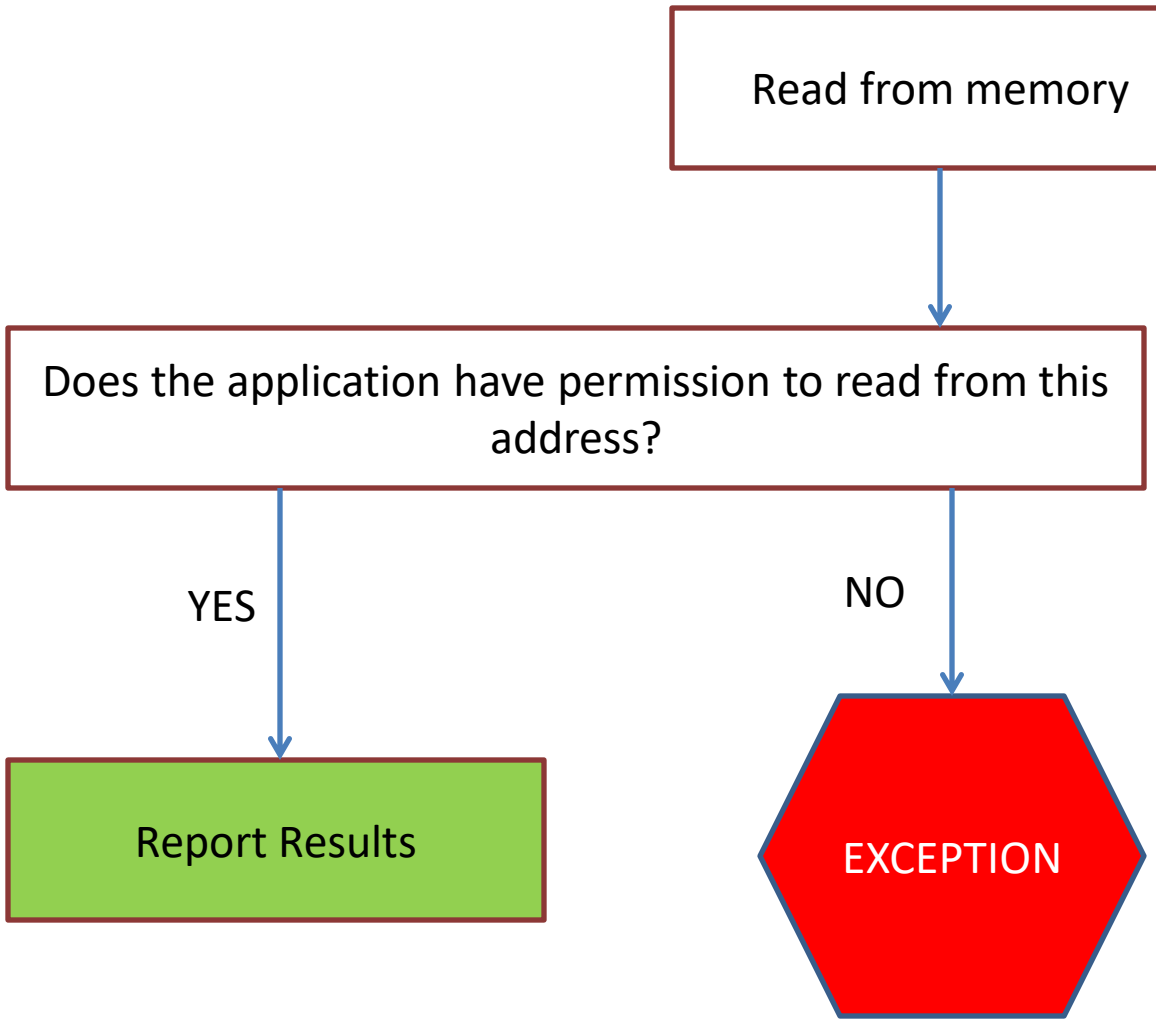
- The attacker tries to find any relation between an encryption process and accessed cache lines
  - To derive a profile of cache activities, the attacker manipulates cache by evicting memory lines of victim process.
- The attacker observes cache activities, i.e., memory lines which are accessed by an encryption process during its execution to obtain a sequence of cache hits and cache misses
  - Eg: observing which memory accesses to a lookup table lead to cache hits allows disclosing indices in the lookup table.
  - After capturing enough samples an offline phase permits to infer the secret data that is used by the victim process.

Spectre and Meltdown

- Side channel attacks exploiting speculative execution



How Meltdown Works



- **By the time the exception is raised, the secret data is already loaded into the cache!!**
- **An attacker can use the Flush + Reload technique to read this data!!**

Preventing Side Channel Attacks

- **Time-padding** ensures that the execution time of a protected function is independent of the function input secret data.
  - Padding prevents an attacker from measuring the execution time of the function.
- **Cache cleansing** prevents obtaining the state of the cache after running the sensitive function.
- **Cache partitioning** allows protecting resources of a trusted process from being accessed by an untrusted process during its execution