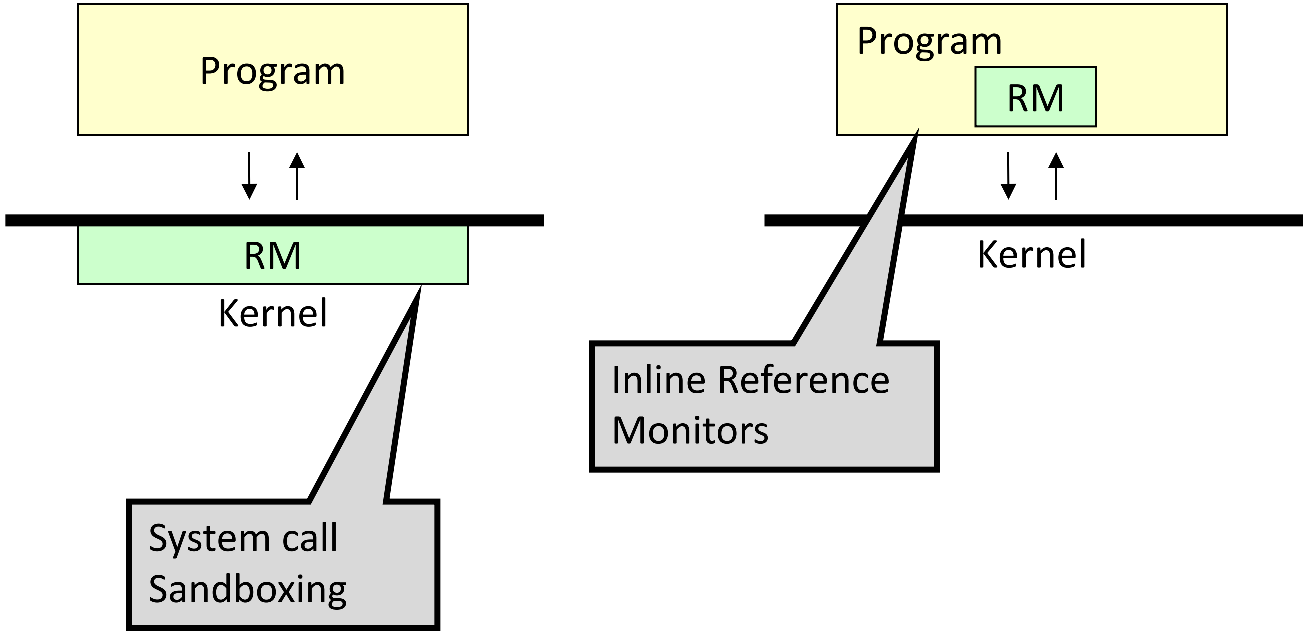
CS5231: Systems Security

Lecture 11: Virtualization and TEE

Recap: Reference Monitors

<u>Reference Monitor</u>: A piece of code that *checks all* **references** to an **object**

<u>Syscall Sandbox</u>: A reference monitor for protecting OS resource objects from an app



Slide from Shmatikov et. al.

Recap: Policy vs. Enforcement Mechanism

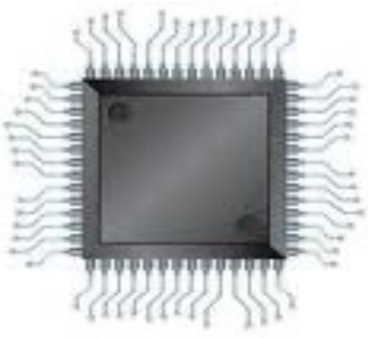
- Access Control Policies
- Enforcement:
  - Process sandboxing
  - Inline Reference Monitors
  - *Virtualization*
  - *Hardware-based isolation / Trusted Execution Env.*

Isolation: Virtualization

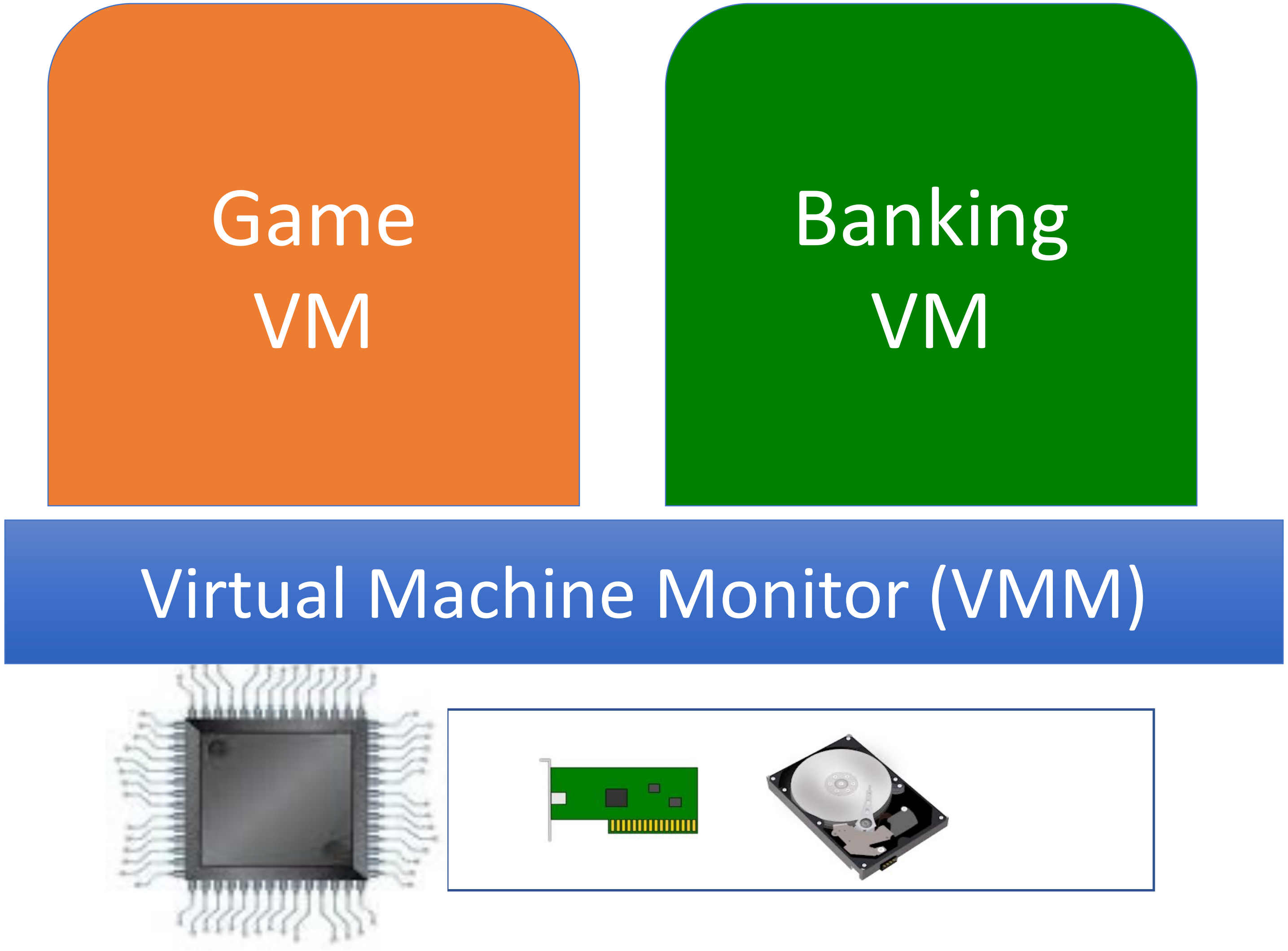
Problem: Isolated Computation on Shared CPU

Operating System

Operating System



Defense(I): Virtualization



A Bit of History…

- Virtual Machines – 1960's
  - Motivation: Sharing of machines between users
  - Many implementations by IBM
- Virtual Machines on RISC / CISC – Late 90's
  - Motivation: Unify under-utilized machines, ease-of-maintenance, security
  - E.g. VMWare
- Heavy utilization in cloud computing…

Public Clouds: EC2

- Virtual Machine Monitor (VMM): **Xen**
- Instance: **A running OS image of virtual machine**
- ECU: **EC2 Compute Unit** ~= 1.2GHz Opteron/Xeon CPU

Amazon CloudWatch

Region: Asia Pacific (Singapore)

Detailed Monitoring for Amazon EC2 Instances

- $3.50 per instance per month, provided at 1-minute frequency

Basic Monitoring for Amazon EC2 instances

- $0.00 (free of charge) per instance per month, provided at 5-minute frequency

Monitoring for Custom Metrics

- $0.50 per metric per month

Assumptions

- Goal: Isolation of Code, data, resources between:
  - Guest VM and Host VMM
  - Between VMs

- Assumptions:
  - Bug-free TCB: Host OS, VMM
  - Malware can affect the guest OS & apps.

Security Applications of Virtualization

- Virtual Machine Isolation
  - Red-Green Systems
    - E.g. Banking VM vs. Normal VM
  - Dynamic Analysis / Containment of Malware

- Virtual Machine Introspection
  - E.g. Run an anti-virus in the VMM

Enforcement Goals for a VMM

- Security VMM Goals:
  - Complete Mediation
  - Trap on all MMU, DMA, I/O accesses
  - Transparency

- Commercial VMM Goals:
  - Performance
  - Compatibility: Run on commodity OSes

Compatibility Challenges: An Example

```
mov eax, (ebp)
```

Virtual Address	Physical Address	Protection Bits (R,W)

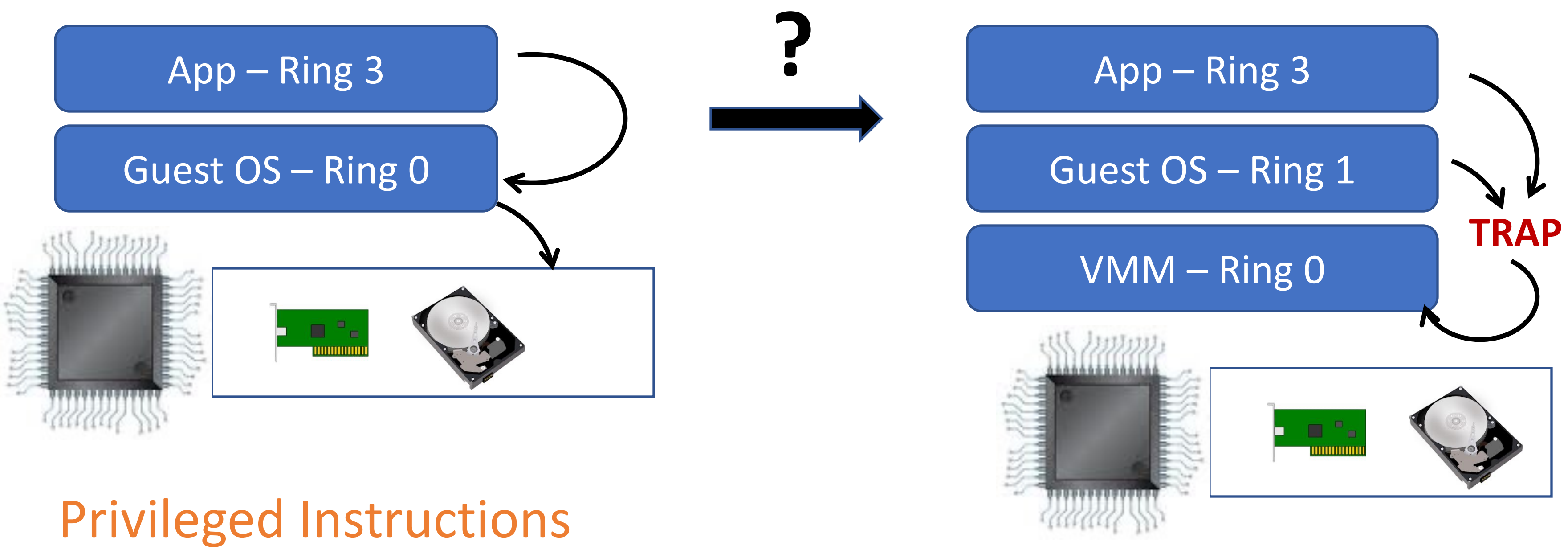
Page Table

OS uses to isolate kernel code / data

Compatibility Challenges

Non-virtualized execution

Virtualized execution



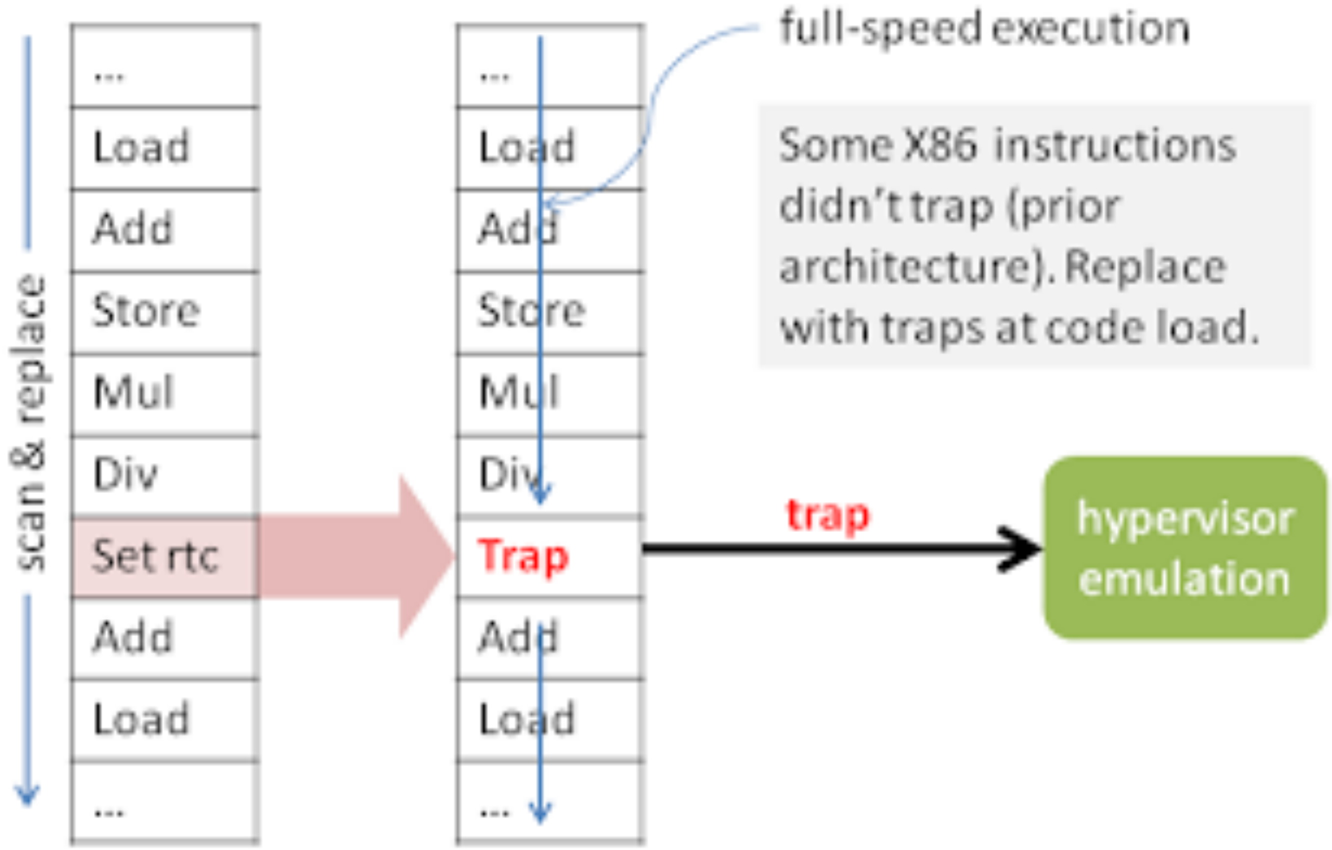
Privileged Instructions

- that trap: e.g. `cli`

Privileged Instructions

- That don’t trap: e.g. `popf`

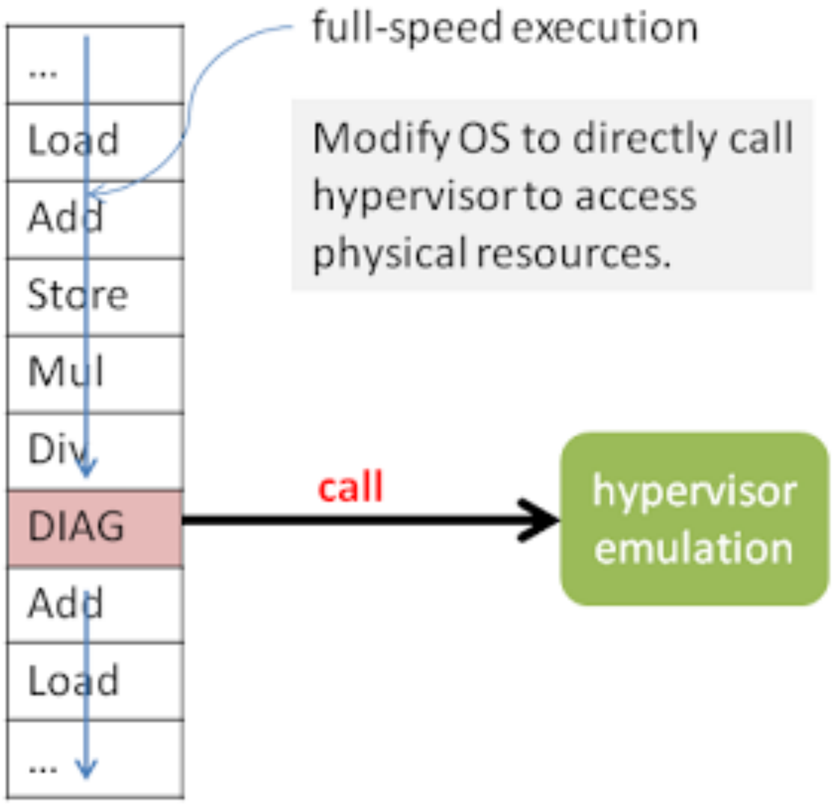
Virtualization Techniques: Binary Translation (VMware)



Early Systems: Dynamo [1998]

Source: http://perilsofparallel.blogspot.sg/2010/06/how-hardware-virtualization-works-part.html

Virtualization: Paravirtualization (Xen)



Source: http://perilsofparallel.blogspot.sg/2010/06/how-hardware-virtualization-works-part.html

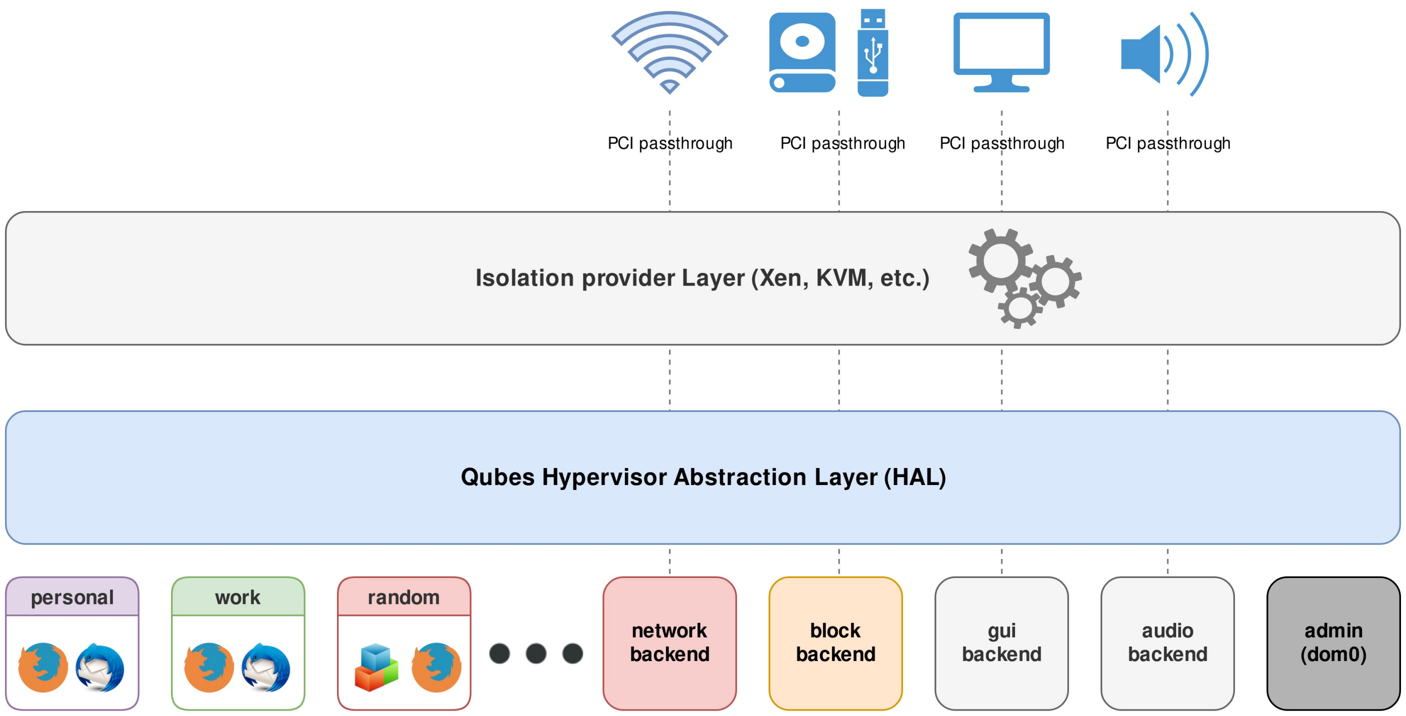
Hardware Assisted Virtualization

- CPUs adding support over the years
- Goal: Better Performance, Security
  - Intel VT-x
    - MMU virtualization using EPT (2009),
    - Nested virtualization - VMCS (2012)
    - I/O virtualization – IOMMU (2009)

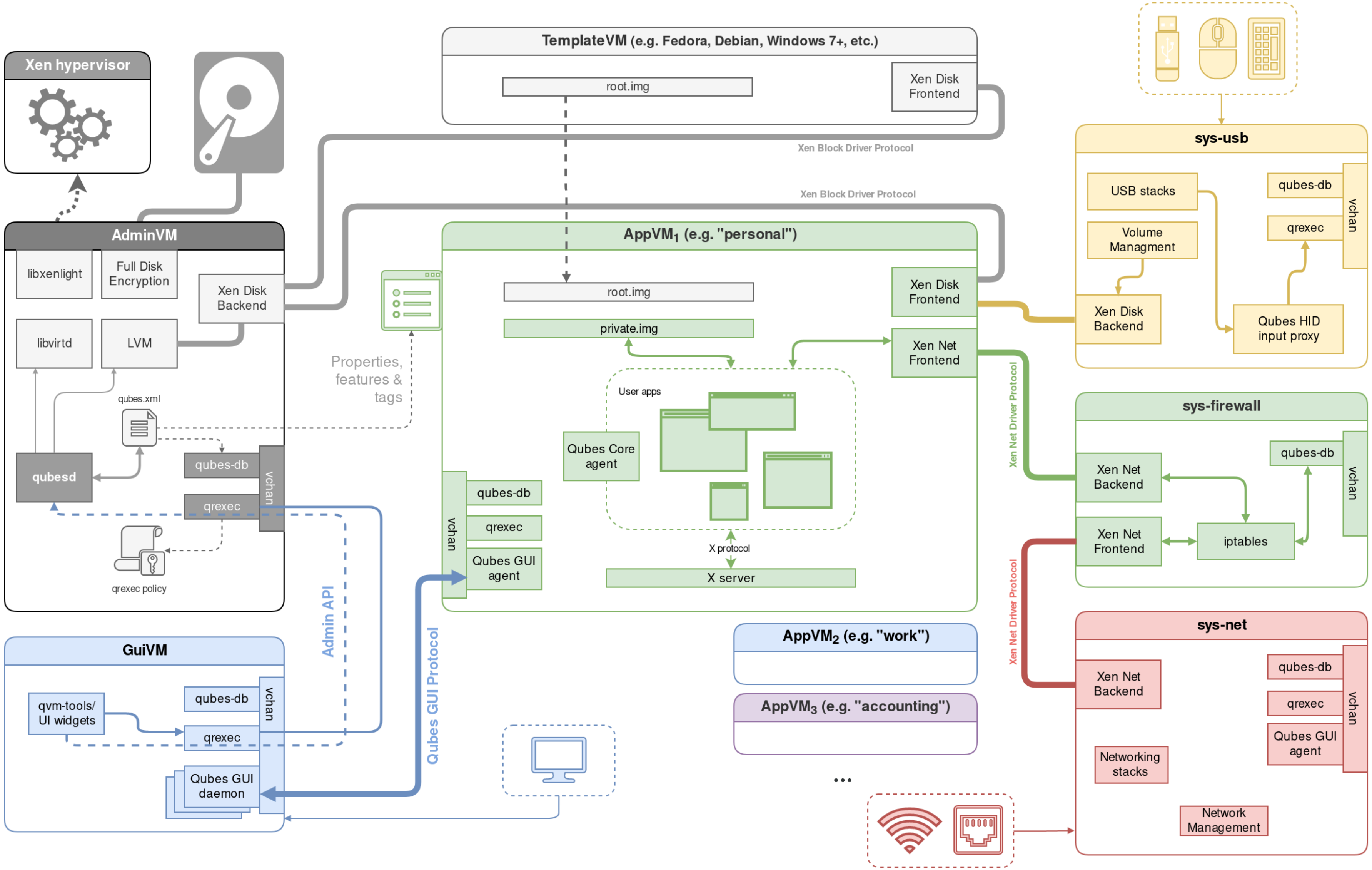
  - Intel VT-d
    - DMA remapping (2009)

Qubes OS

- A reasonably secure operating system
  - A network of virtual machines in a computer

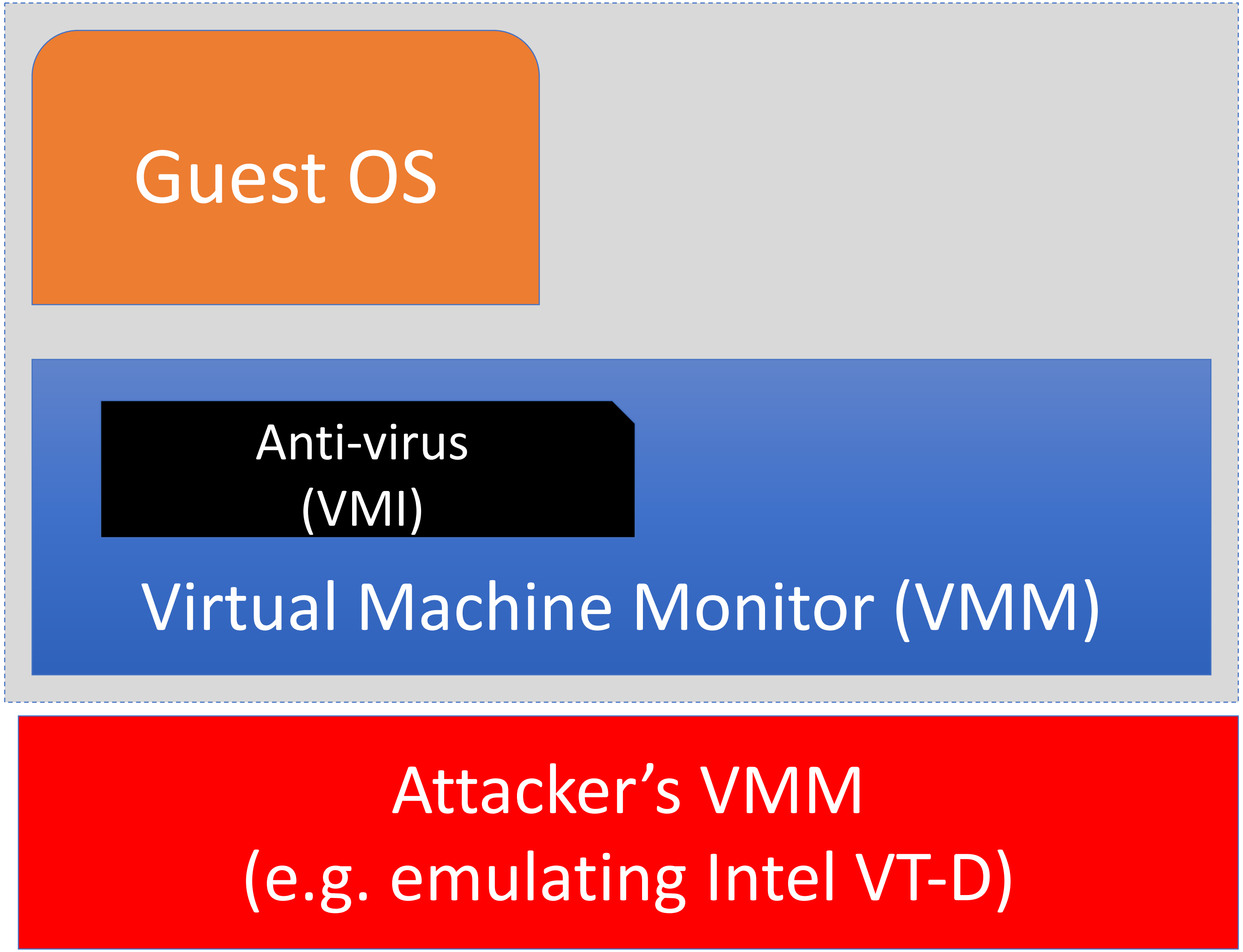


Qubes OS Architecture

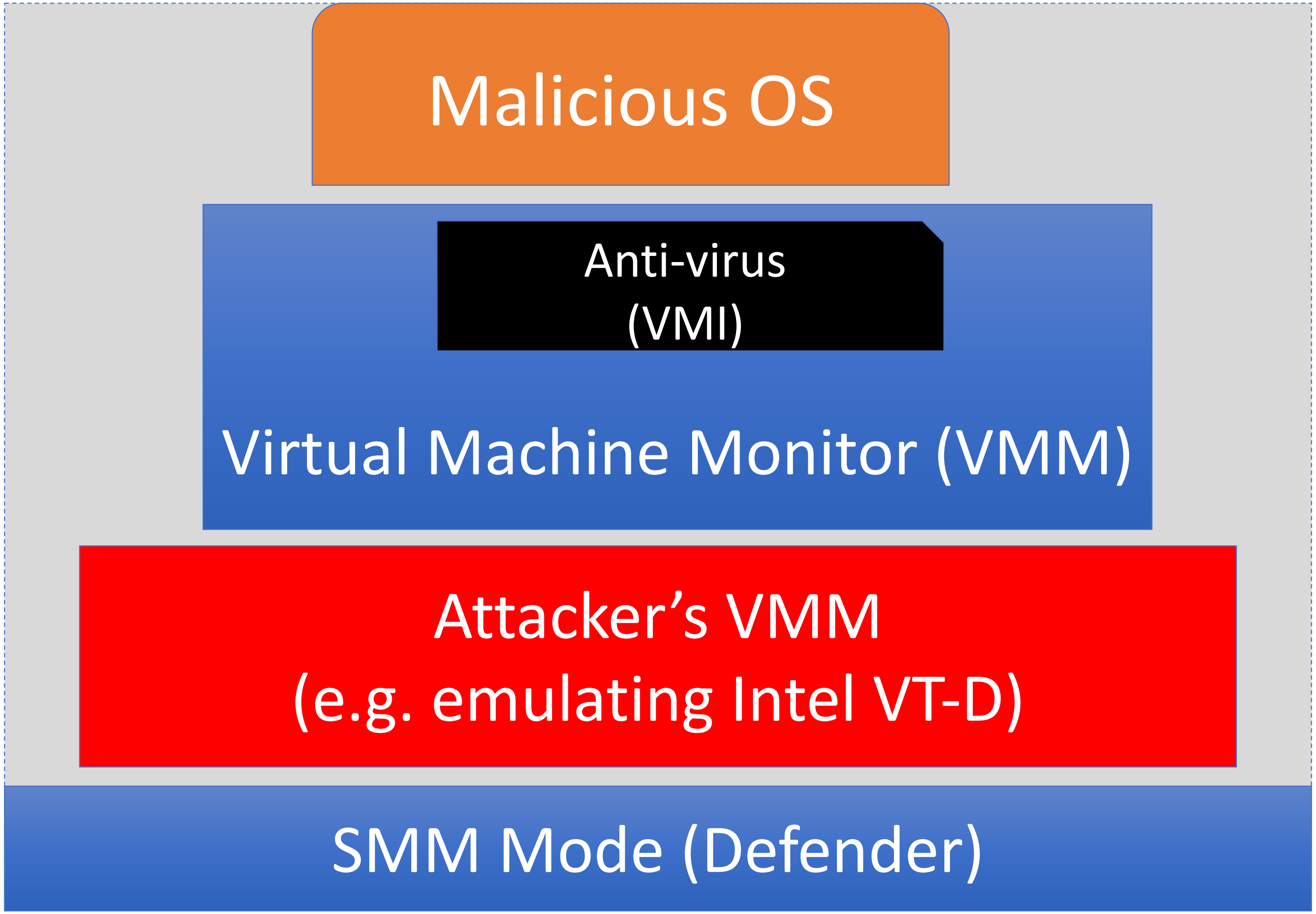


Limitations of Virtualization

Virtual Machine Based Rootkits



The Problem of Secure "Root of Trust":
Is highest layer of privilege malicious?

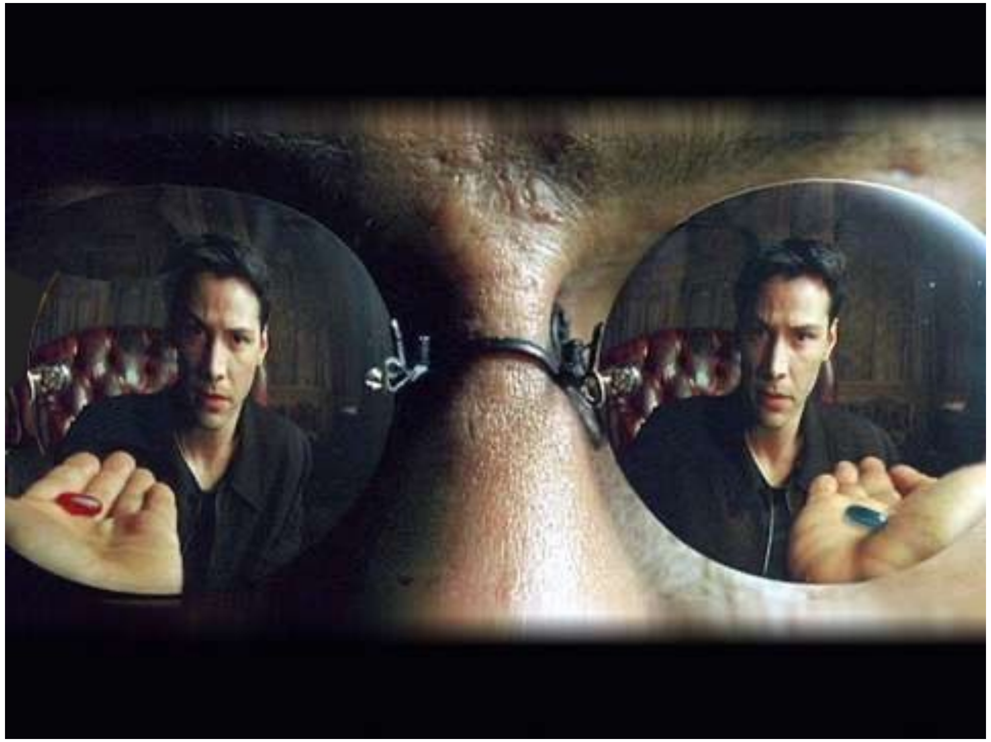


Implication on Malware Containment

- In principle, is some containment possible?
  - Yes, When the highest layer of privilege is trusted
  - E.g. the VMM is trustworthy

Virtual Machine Based Rootkits: Can the software know its virtualized?

- Blue Pill: “blissful ignorance of illusion”
- Red Pill: “Detects you are virtualized”



VMM Detection: The Red Pill

- Red Pill: “Detects you are virtualized”
- Ways to achieve a “red pill” attack:
  - Commercial VMMs aren’t fully transparent
    - E.g. VmWare emulates i440bx chipset (old)
  - Virtualization Timing latencies Measurements
  - Many other measurement channels [HotOS’07]

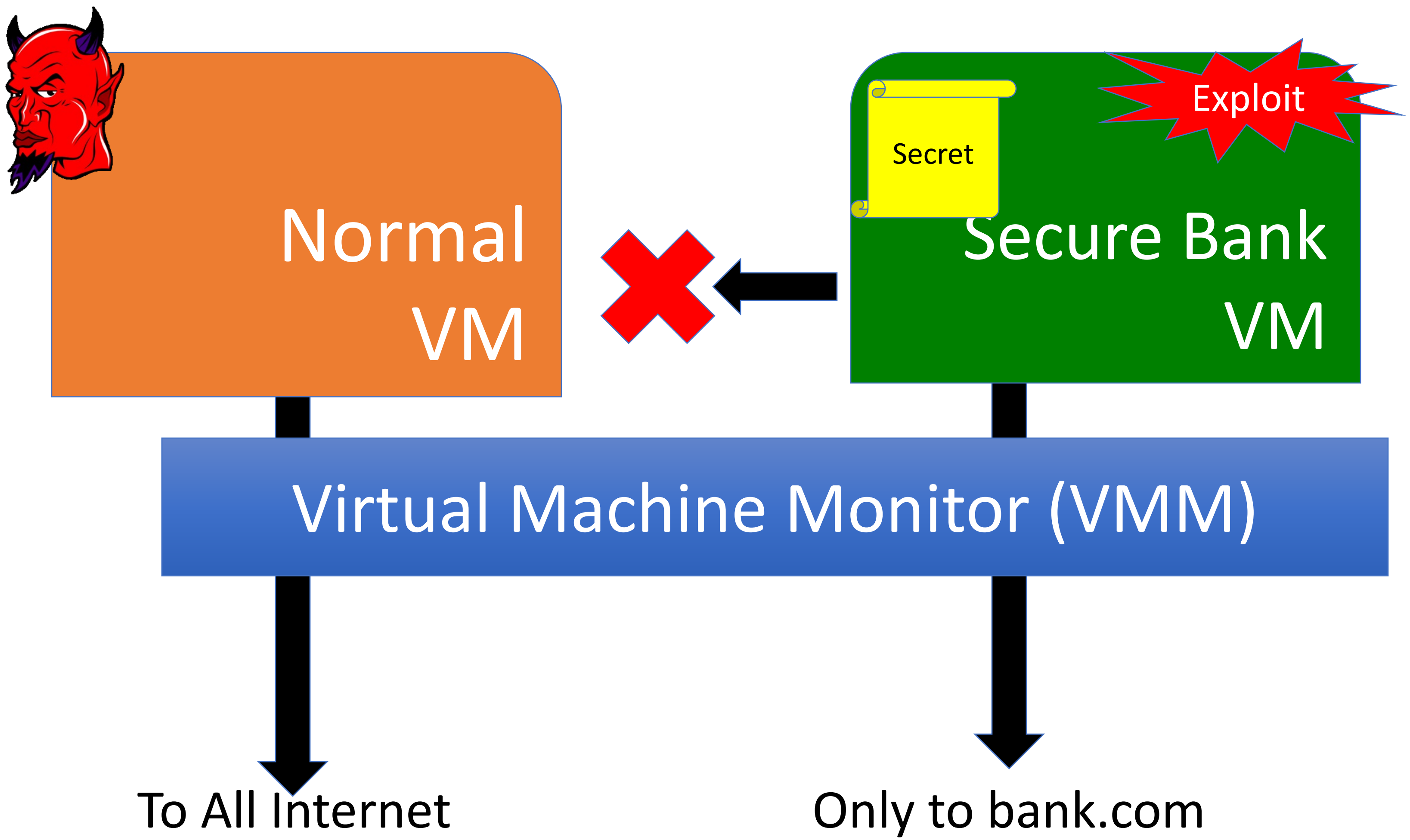
- Applications of VM Detection:
  - Malware can detect introspection software (e.g. AV)
    - Can utilize “Anti-VM” techniques
  - Benign use: Copy protection by VM duplication

Compatibility is Not Transparency: VMM Detection Myths and Realities

Implication on Malware Containment

- In principle, is some containment possible?
  - Yes, When the highest layer of privilege is trusted
  - E.g. the VMM is trustworthy

- Detecting virtualization is easy
  - It can thwart malware analysis (introspection)



Can attacker leak secret document to evil.com?

The Problem of Covert Channels

- Definition: "An unintended channel of communication between 2 untrusted programs"

- E.g. Shared Cache Latency
  - Sender
    - Send bitval 1: Perform random memory access
    - Send bitval 0: Do nothing
  - Receiver
    - Rcv bitval 1: If long read time for a fixed memory loc.
    - Rcv bitval 0: If short read time for fixed memory loc.

  - Can get 0.02 bits/sec on Amazon EC2 [CCS'09]
  - Many channels: Disk, I/O, Virtualization latency, ...

Implication on Malware Containment

- In principle, is some containment possible?
  - Yes, When the highest layer of privilege is trusted
  - E.g. the VMM is trustworthy
- Detecting virtualization is easy
  - It can thwart malware analysis (introspection)
- Which containment using VMs is possible: integrity vs. confidentiality?
  - Yes, for Integrity policy – I.e., protecting contained malware from corrupting benign data outside the VM
  - No, for confidentiality, covert channels are a problem

Optional Reading Material for the curious…

- SubVirt: Implementing malware with virtual machines

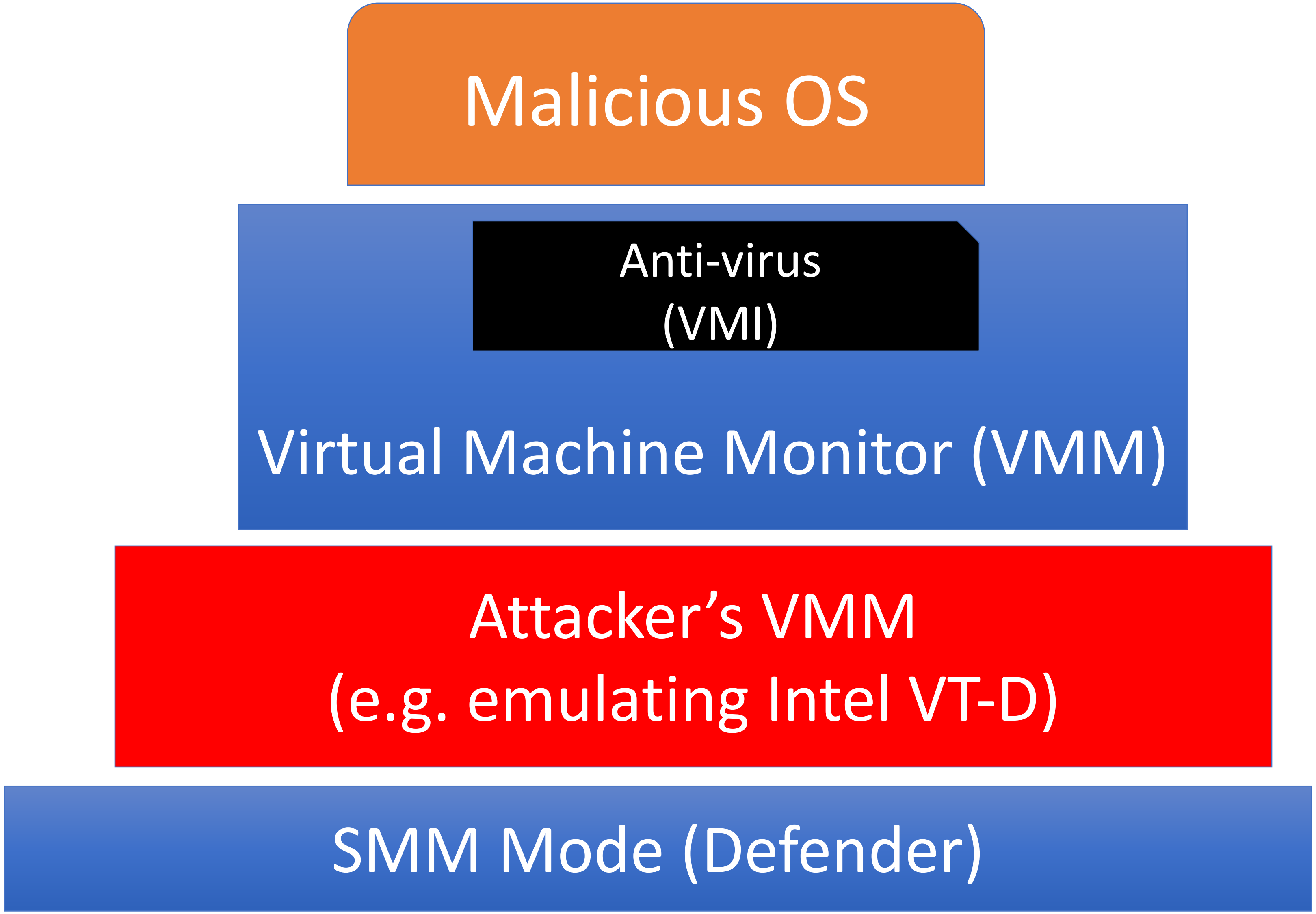
- Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems

- HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity

- Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds

Trusted Execution Environments:
The Basic Idea

The Problem of "Root of Trust": Is highest layer of privilege malicious?

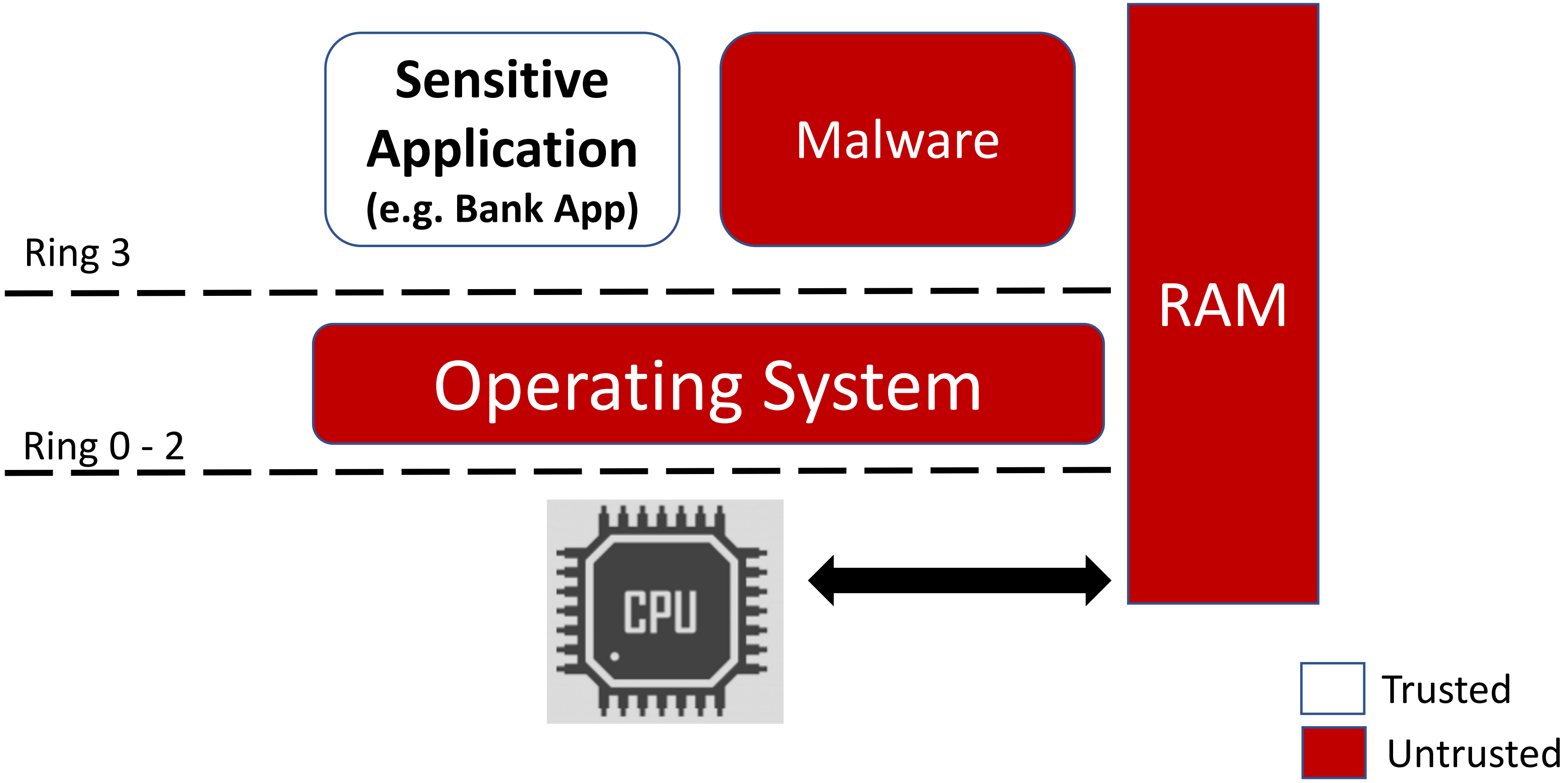


Solution: TEEs ensure Secure “Root of Trust”

- Trusted Execution Environments (TEEs)
  - A **hardware root-of-trust**
  - Can assume that all software is malicious
- Why trust hardware?
  - Tamper-resistant from all software malware
  - Perhaps less complex, easier to verify?

Trusted Execution Environment (TEE): Security Model

- Trust the hardware
- Don't trust other software on the system



slide courtesy: Shweta Shinde

Evolution of TEEs

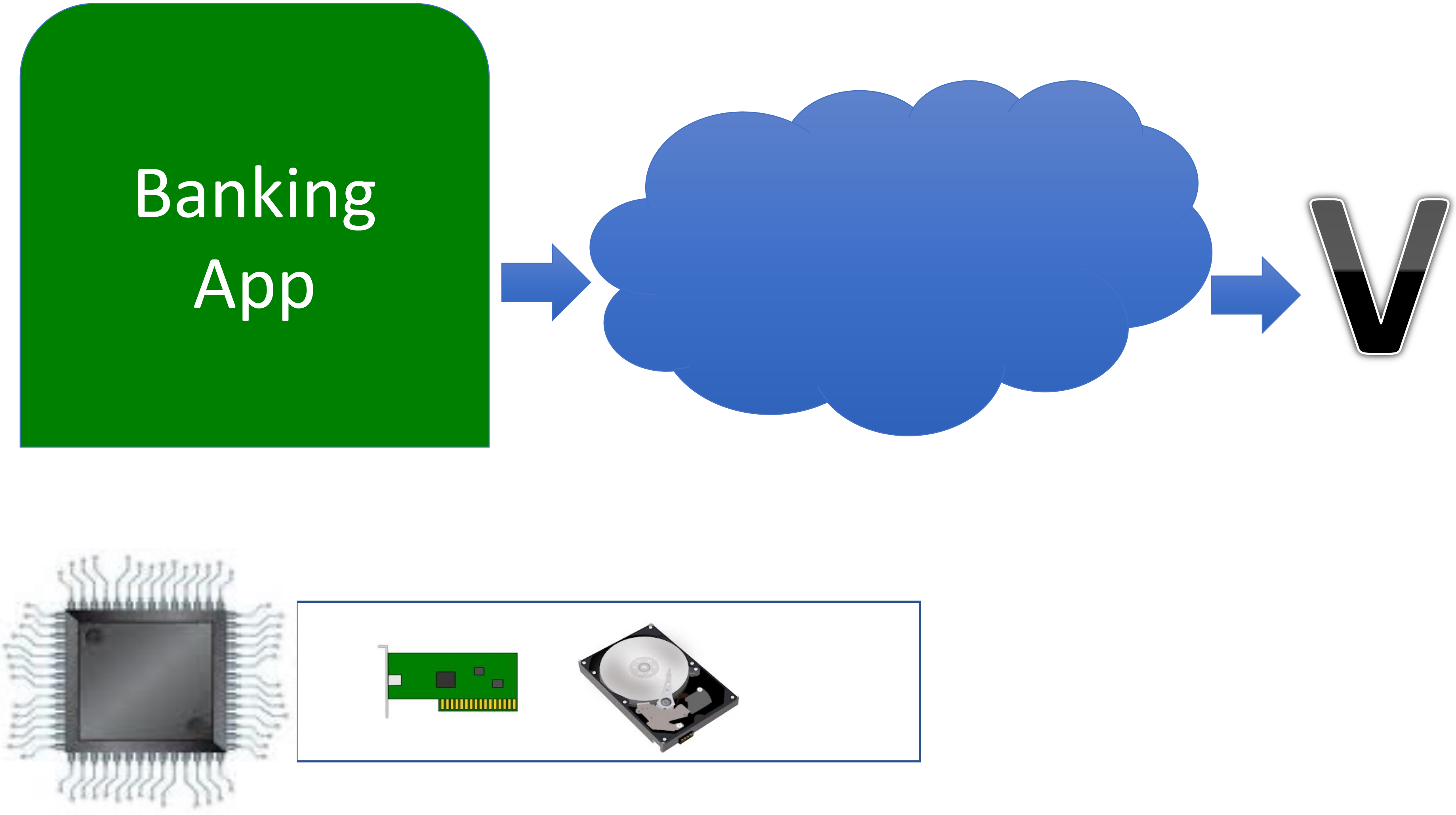
ARM TRUSTZONE
System Security

intel Xeon inside

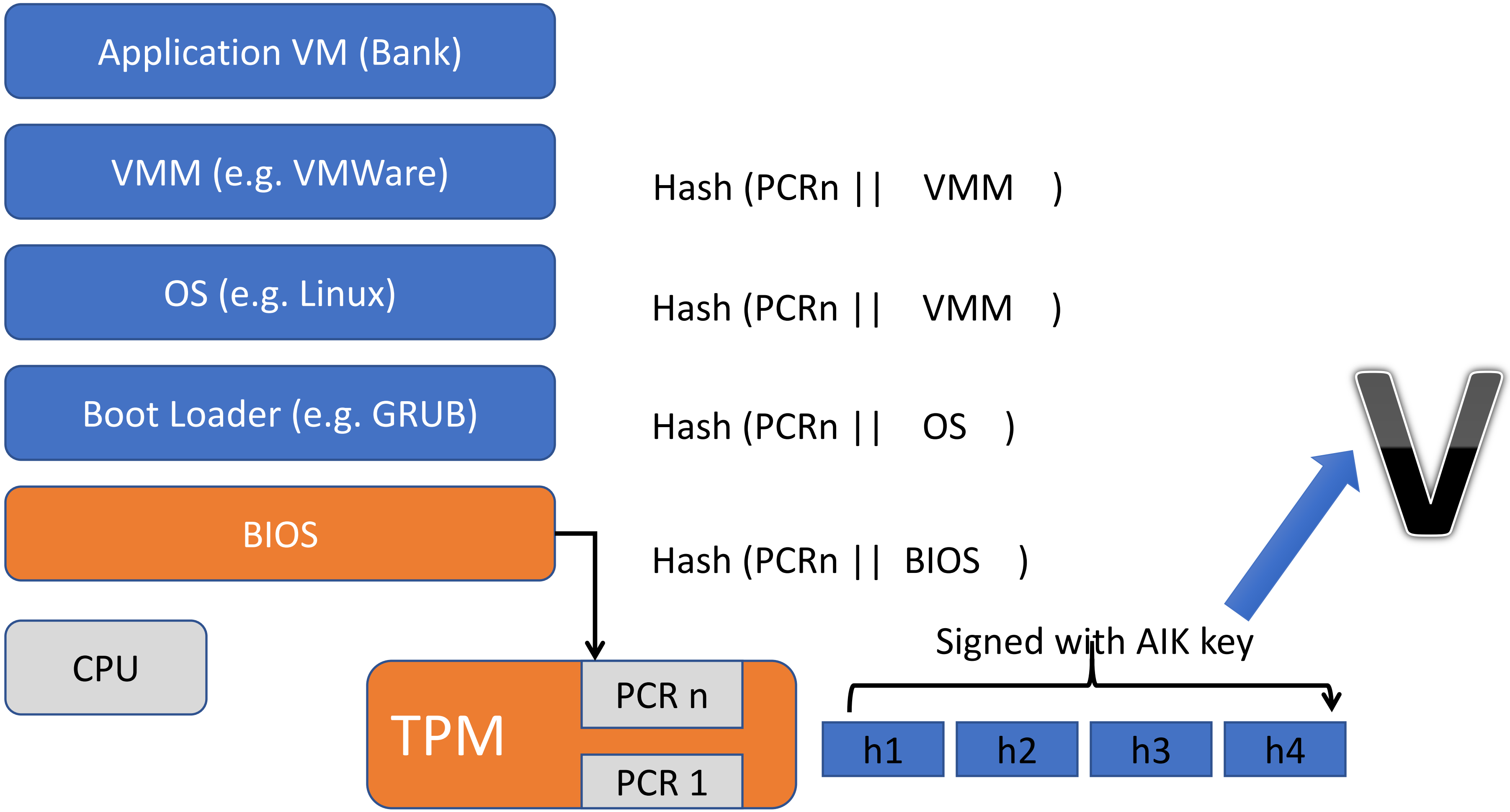
2004	2005	2006	2014	2016
TPM 1.2	TrustZone	Intel TXT	TPM 2.0	Intel SGX

Trusted Execution Environment

Trusted Execution Primitives (I): Remote Attestation

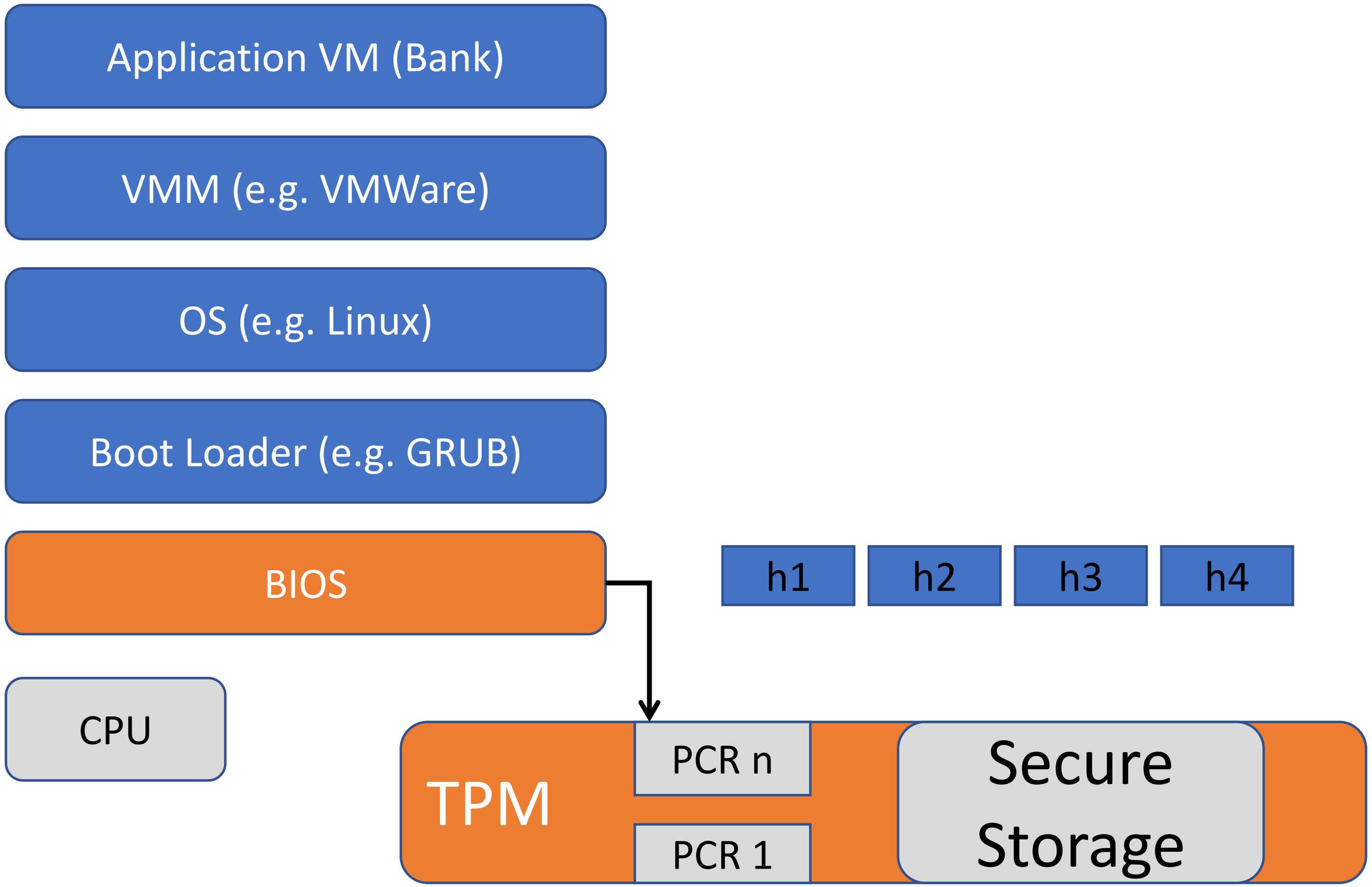


Trusted Execution Primitives (I): Remote Attestation



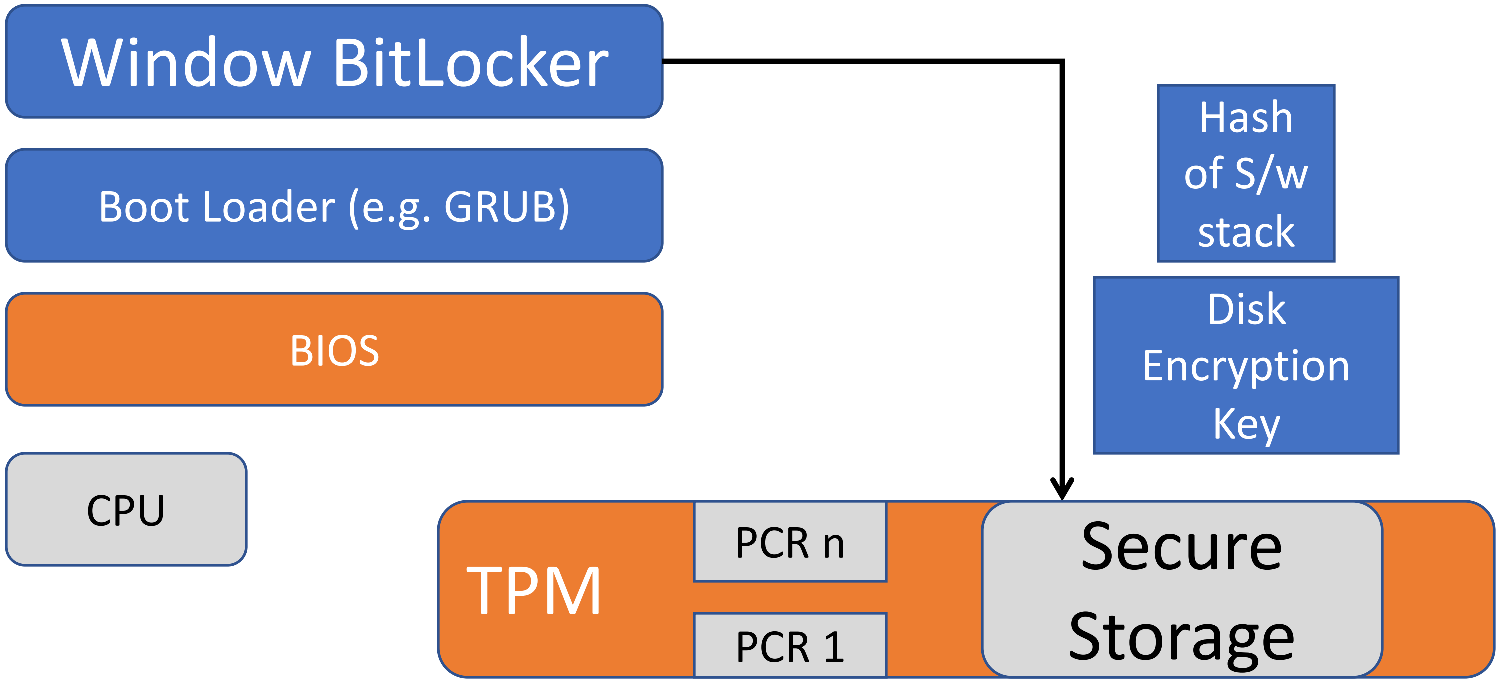
Each TPM has a AIK signing key

Trusted Execution Primitives (II): Trusted Boot

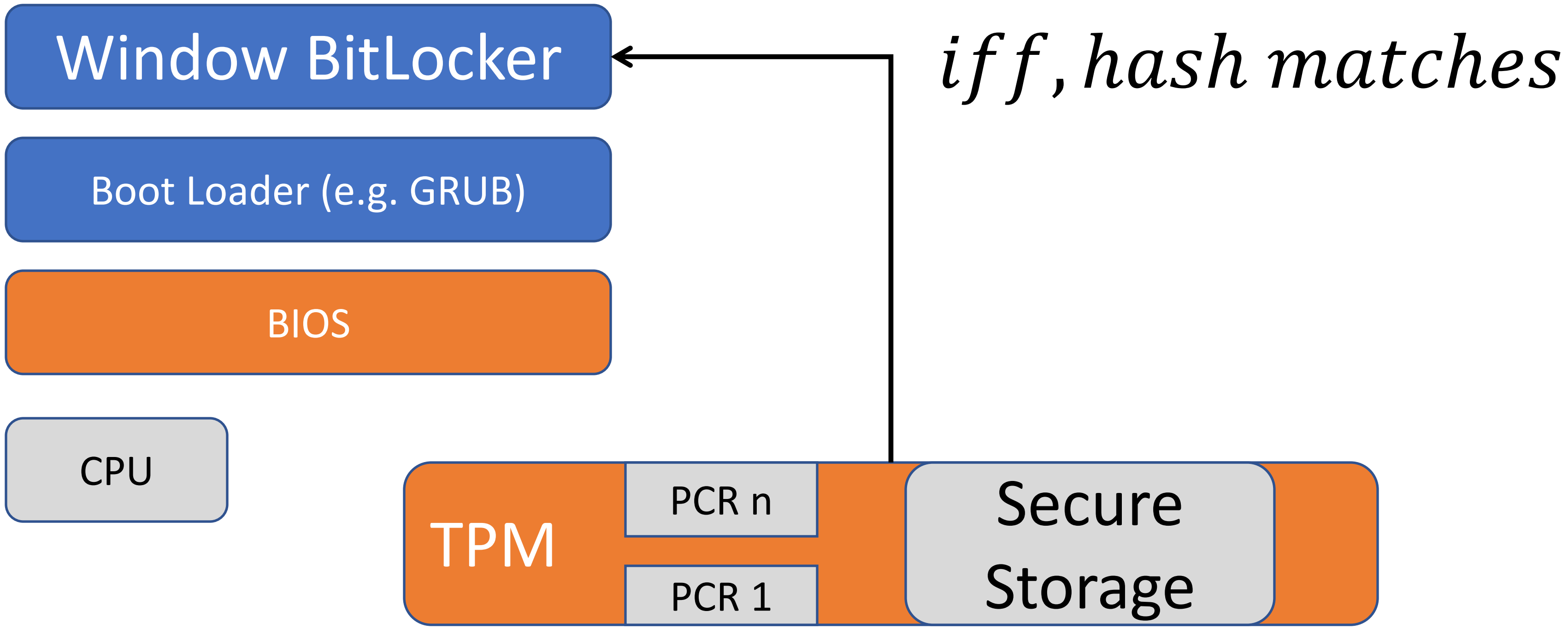


Trusted Execution Primitives(III): Sealing Data

Use TPM Measurements & Secure Storage for Disk Encryption Systems?



Trusted Execution Primitives (III): Unsealing Data



Trusted Execution Primitives (III): Sealed Storage

```
# echo 'Secret!!!' | tpm_sealdata -z -i/proc/self/fd/0
-o./mysecret.blob -p17 -p18 -p19


// assuming PCR's are the same
# tpm_unsealdata ./mysecret.blob
Secret!!!

// assuming PCR's are different
# tpm_unsealdata ./mysecret.blob
error 24: Tspi_Data_Unseal: 0x00000018 - layer=tpm,
code=0018 (24), Wrong PCR value
```

Trusted Computing Primitives (I – II): Static Root-of-Trust

- So far, we've seen SRTM systems
  - Checks / Verification at load time

- Many Applications
- Windows BitLocker
- Linux TrustedGrub (because TXT is too slow!)
- Build your own "secure apps"
  - eXtensible, Modular Hypervisor Framework (from CMU)

Use Case: Full Volume Encryption

- Encryption at the block level underneath file system
- Everything in the volume is encrypted.
- BitLocker is used by Microsoft since Windows Vista
- BitLocker takes advantage of TPM (TEEs)
  - Top level root key sealed in hardware
  - Root key encrypts disk encryption key, which encrypts sector data
- CPU protects disk encryption key by encrypting it
- CPU releases key only after comparing hash of early (unencrypted) boot files with previous hash

To Continue

- Other trusted computing technology
  - ARM TrustZone
  - Intel SGX