SC3010
Computer Security

Lecture 6: Operating System Security (II)

Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

Confinement

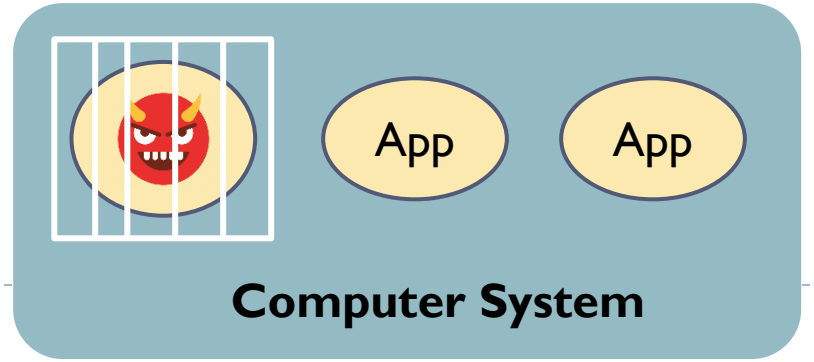
An important security strategy in OS protection

- When some component (e.g., application) in the system is compromised or malicious, we need to prevent it from harming the rest of system.
- Confinement: restricts the impact of each component on others.
- Follow the principle of least of privilege

Application scenario

- Cut off the propagation chain.
- Malware testing and analysis

Can be implemented at different levels



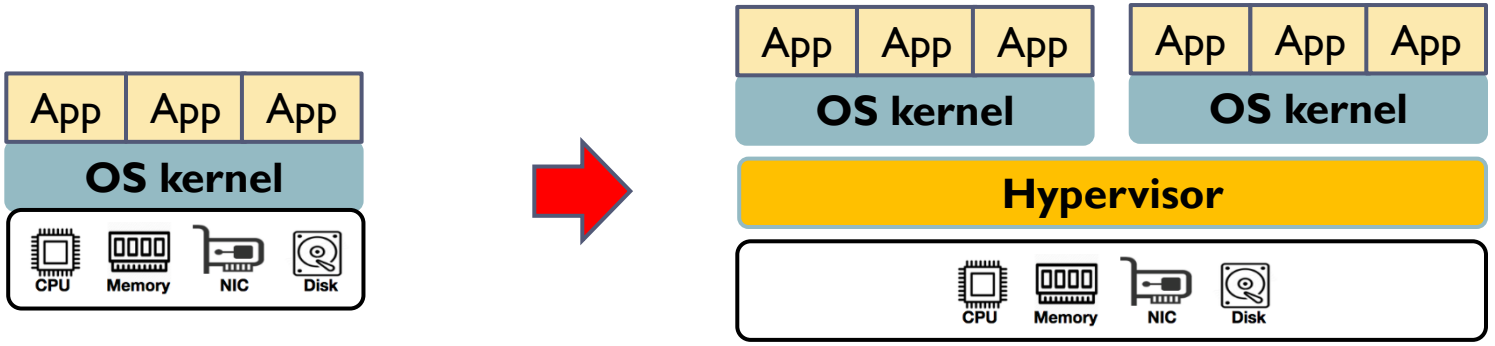
OS Level Confinement: Virtual Machine

Virtualization: the fundamental technology for cloud computing

- Different operating systems (virtual machines) run on the same machine
- Each virtual machine has an independent OS, logically isolated from others

Technical support

- Software layer: hypervisor or virtual machine monitor (VMM) for virtualizing and managing the underlying resources, and enforcing the isolation
- Hardware layer: hardware virtualization extensions (Intel VT-x, AMD-V) for accelerating virtualization and improving performance



Virtual Machine for Malware Analysis

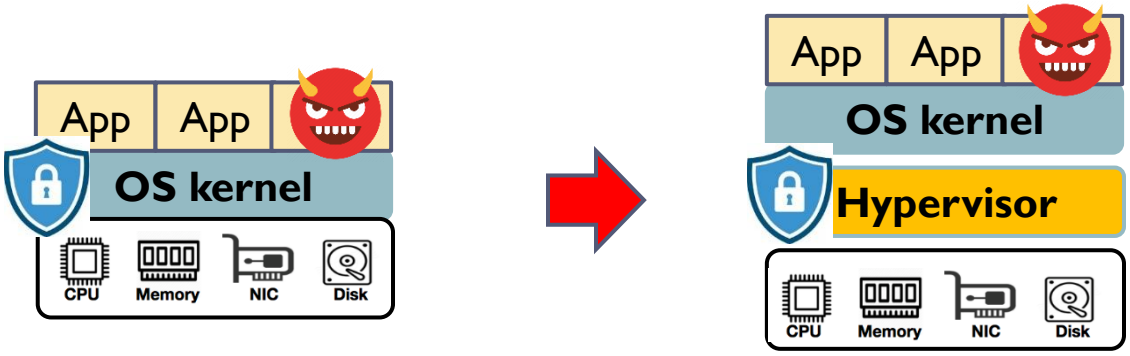
Malware analysis: deploy the malware and observe its behaviors.

Deploying the malware in the native OS

- The malware could compromise the entire OS (e.g., rootkit)
- The observation results are not reliable and could be manipulated.

Virtual machine: an ideal environment for testing malware

- The malware cannot cause damages outside of the VM
- The malware's behavior can be observed from the hypervisor/host OS



Limitations of Virtualization

The introduction of hypervisor can incur large attack surface

- The hypervisor has big code base, and inevitably brings more software bugs
- The hypervisor has higher privilege than the OS kernel. If it is compromised, then the attacker can take control of the entire system more easily.

The performance of a VM could be affected by other VMs due to the sharing of hardware resources.

Challenges of malware analysis with virtualization

- Although hypervisor has a complete view of VMs, there exists semantic gaps between high-level activities inside VMs and observed low-level behaviors
- This solution is not compatible with Trusted Execution Environment (TEE)
- A smart malware can detect that it is running inside a VM, not the actual environment, e.g., larger memory latency variance, reduced TLB size, etc. Then it behaves like normal applications,

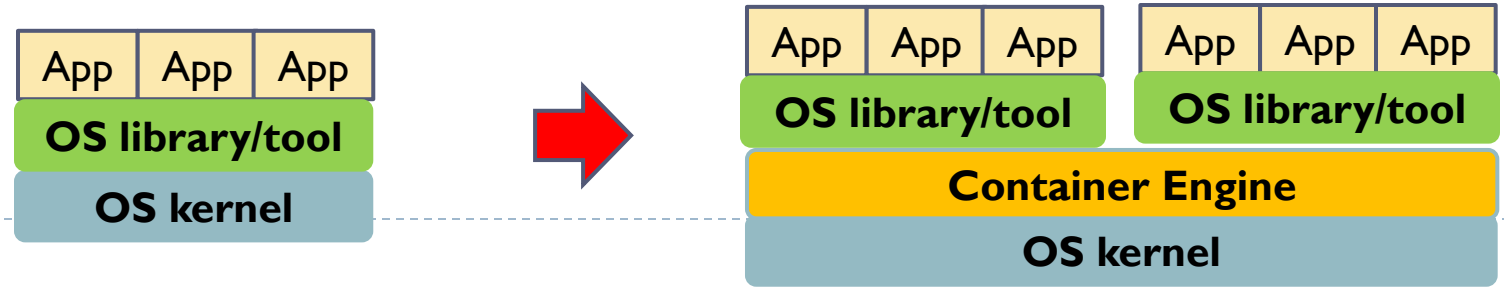
Process Level Confinement: Container

A standard unit of software

- A container is a lightweight, standalone, executable software package that packages everything needed to run the application
  - Code, system tools and libraries, configurations.
- A Container Engine (e.g., Docker) is introduced to manage containers

Advantages of containers

- Portability: containers can run consistently across different environments, from development to production, reducing compatibility issues.
- Efficiency: sharing OS reduces overhead, with high resource utilization.
- Isolation: Applications operate in their own environment, minimizing conflicts and enhancing security.



Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

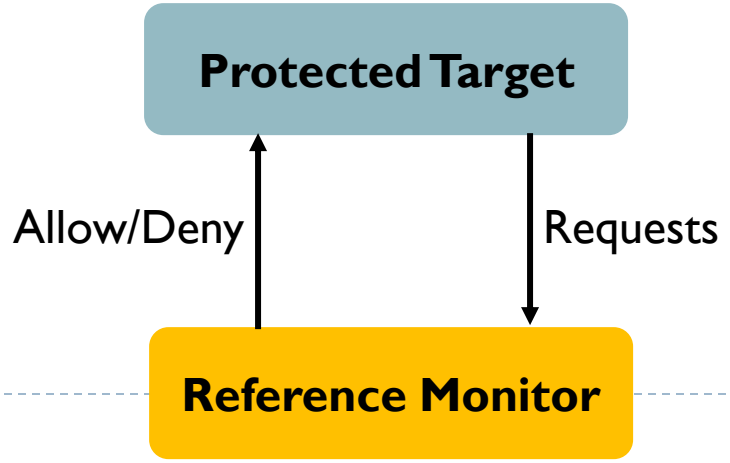
Reference Monitor (RM)

A conceptual framework

- Enforces access control policies over any protected target in a system.
- Mediates all access requests, and deny any request that violates policy

Significance

- Trusted Computer System Evaluation Criteria (TCSEC) emphasizes the necessity of a reference monitor in achieving higher security
- RM serves as the foundation for various security models, ensuring that the access control policies are consistently enforced across the system



Requirements of RM

Function requirement

- RM must intercept and evaluate every access request without exception.
- RM is able to deny the malicious requests

Security requirement

- RM must be tamper-proof, and protected from unauthorized modification to maintain its integrity

Assurance requirement

- The validation mechanism must be small enough to be thoroughly analyzed and tested for correctness.

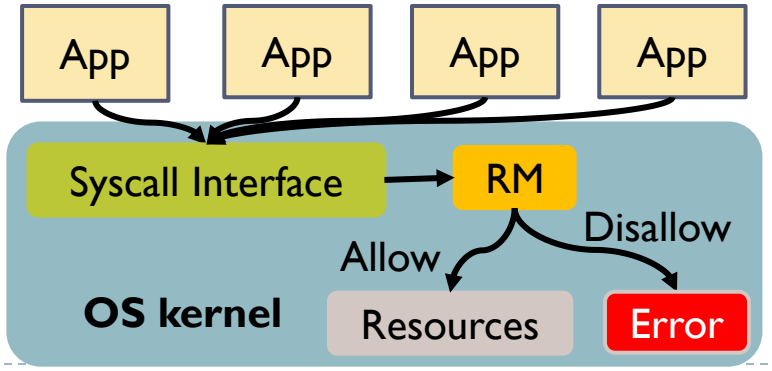
Example: OS-based RM

A core component within the OS kernel

- Enforce access control policies by monitoring and mediating all system calls made by applications.
- Ensure that all applications operate within their authorized permissions, preventing unauthorized access to system resources, including file operations, network communications, and process control.

Implementation

- Intercept all system calls, check permissions and allow/disallow execution.
- Typical examples: Security-Enhanced Linux (SELinux)



Example: Application-based RM

A security mechanism embedded within applications

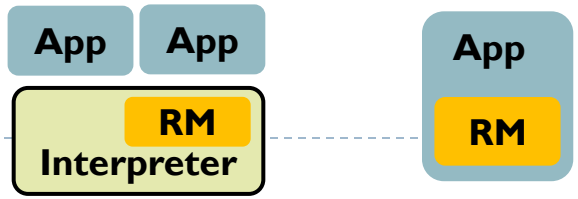
- Enforce access control policies, provide fine-grained control over application behaviors, and prevent unauthorized actions.

Integrating RM with interpreter

- Every operation will be checked against security policies before execution
- Example: JavaScript engine enforces sandboxing by restricting access to certain APIs or resources during script execution.

Inline RM

- Inserting RM directly into the application's code. This could be achieved with source code instrumentation, or binary rewriting.
- Example: StackGuard



Example: Hardware-based RM

Responsible for monitoring and regulating all the software activities, including OS kernel.

- Any operation violating the security policy will throw a hardware exception

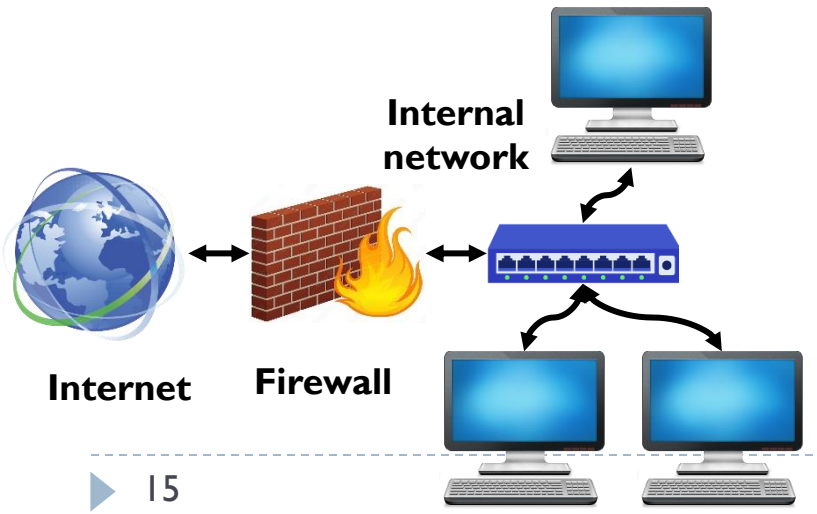
Hardware-based RMs conduct various checking

- Memory access management.
  - If each memory access is within the process' memory range.
  - If each access follows the allowed permission (read, write, executable, set in the Page Table Entry). Recall the Non-executable Memory mechanism.
- Privilege mode management.
  - At any time, CPU can be in one mode, either user or kernel.
  - Privileged instructions can only be issued in kernel mode.
  - Context switch is required for user mode to call privileged functions.

Example: Network-based RM

Firewall

- Monitor and regulate the network traffics based on the security policy.
  - Outbound policy: define what traffic is allowed to exit the network
  - Inbound policy: define what traffic is allowed to enter the network
- Possible actions:
  - Allow: permitted through the firewall.
  - Deny: not allowed through the firewall.
  - Alert: send alert to the administrator.



Protocol	Source Addr	Source Port	Dest. Addr	Dest. Port	Action
TCP	Any	Any	192.168.42.0/24	>1023	Allow
TCP	192.168.42.1	Any	Any	Any	Deny
TCP	Any	Any	192.168.42.1	Any	Deny
TCP	Any	Any	192.168.42.55	25	Allow
TCP	**Any**	**Any**	**Any**	**Any**	**Deny**

Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

Using Hardware to Protect Software

Software is not always trusted

- Privileged software (OS, hypervisor) usually has very large code base, which inevitably contains lots of vulnerabilities.
- Once it is compromised, the attacker can do anything to any apps running on it.

SW	Line of codes
Linux Kernel 5.12	28.8M
Windows 10	50M
VMWare	6M
Xen	0.9M

Commercial software typically has 20 to 30 bugs for every 1k lines of code

Hardware is more reliable

- After the chip is fabricated, it is hard for the attacker to modify it. The integrity of hardware is guaranteed.
- It is also very hard for the attacker to peek into the chip and steal the secret (e.g., encryption key). The confidentiality of hardware is guaranteed.
- It is more reliable to introduce security-aware hardware to protect the operating system and applications

Basic Functionality: Encryption

Encryption performed using dedicated hardware

- Trusted Platform Module (TPM)
- Hardware Security Modules (HSM)
- Advanced Encryption Standard New Instructions (AES-NI)

Benefits

- Performance efficiency: faster execution with optimized hardware
- Energy efficiency: lower power consumption compared to software solutions
- Security: resistant to software-level attacks and malware
- Ease of use: transparent encryption with minimal user interaction.

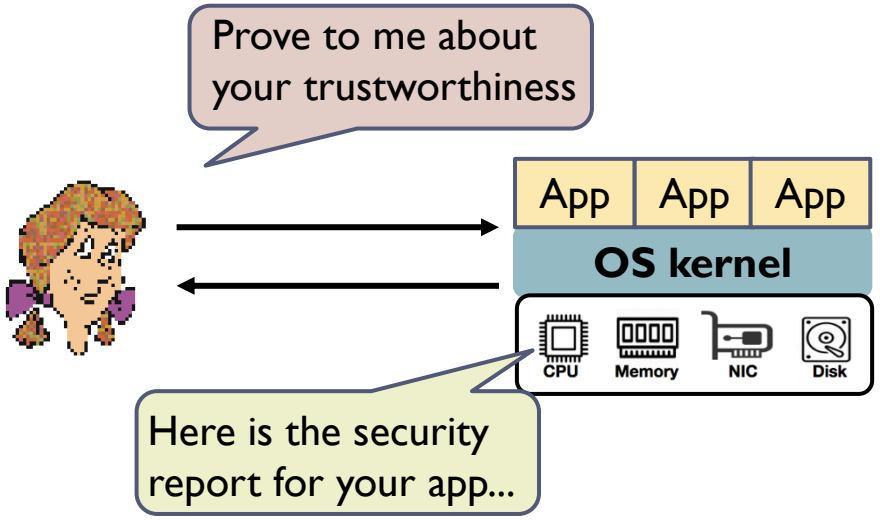
Applications

- Data protection in storage
- Secure boot
- Cloud security

Basic Functionality: Remote Attestation

A mechanism that allows a user to know whether her app executes securely on a trusted platform.

- A remote platform provides unforgeable evidence about the security of its software to a client.
- A common strategy to prove the software running on the platform are intact and trustworthy.



Major components for remote attestation

- Integrity measurement architecture: provide reliable and trustworthy security report
- Remote attestation protocol: ensuring the attestation report is transmitted to the client without being modified by attackers in OS, apps or network

Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

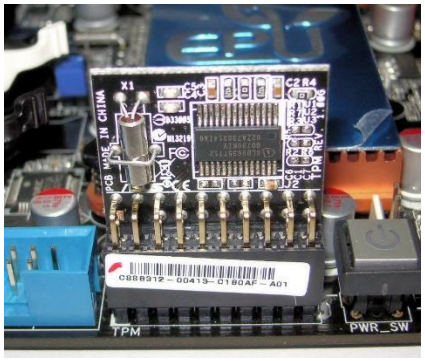
Trusted Platform Module (TPM)

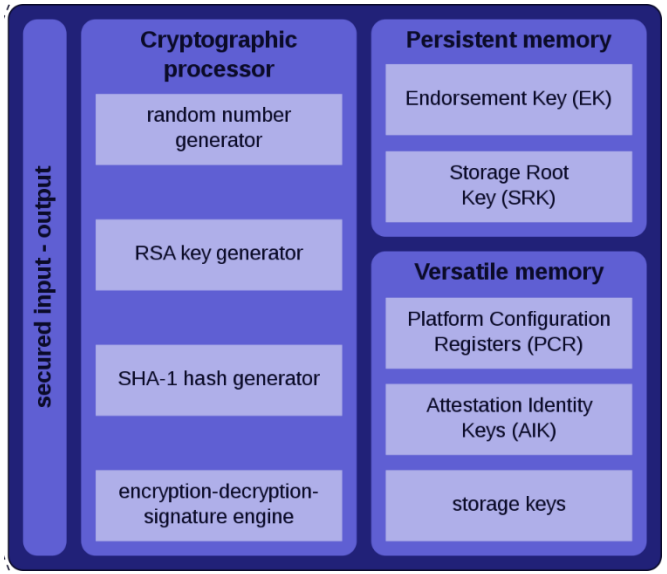
A chip integrated into the platform

- A separated co-processor
- Its state cannot be compromised by malicious host system software

Inside the chip

- Random number and key generators
- Crypto execution engine
- Different types of crypto keys.





Development and Implementation

Designed by Trusted Computing Group (TCG)

- First version: TPM 1.1b, released in 2003.
- An improved version: TPM 1.2, developed around 2005-2009
  - Equipped in PCs in 2006 and in servers in 2008
  - Standardized by ISO and IEC in 2009
- An upgraded version: TPM 2.0, released on 9 April 2014.

Application of TPM

- Intel Trusted Execution Technology (TXT)
- Microsoft Next-Generation Secure Computing Base (NGSCB)
- Windows 11 requires TPM 2.0 as a minimal system requirement
- Linux kernel starts to support TPM 2.0 since version 3.20
- Google includes TPMs in Chromebooks as part of their security model
- VMware, Xen, KVM all support virtualized TPM.

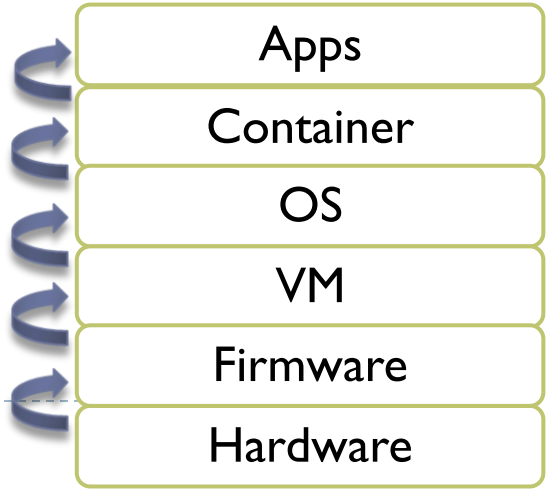
Building Chain of Trust with TPM

Chain of Trust: Establish verified systems from bottom to top

- From a hierarchic view, a computer system is a layered system.
  - Lower layers have higher privileges and can protect higher layers.
  - Each layer is vulnerable to attacks from below if the lower layer is not secured appropriately.
- TPM serves as the root of trust: establish a secure boot process from TPM, and continue until the OS has fully booted and apps are running.
  - The bottom layer validates the integrity of the top layer.
  - It is safe to launch the top layer only when the verification passes.

Potential applications

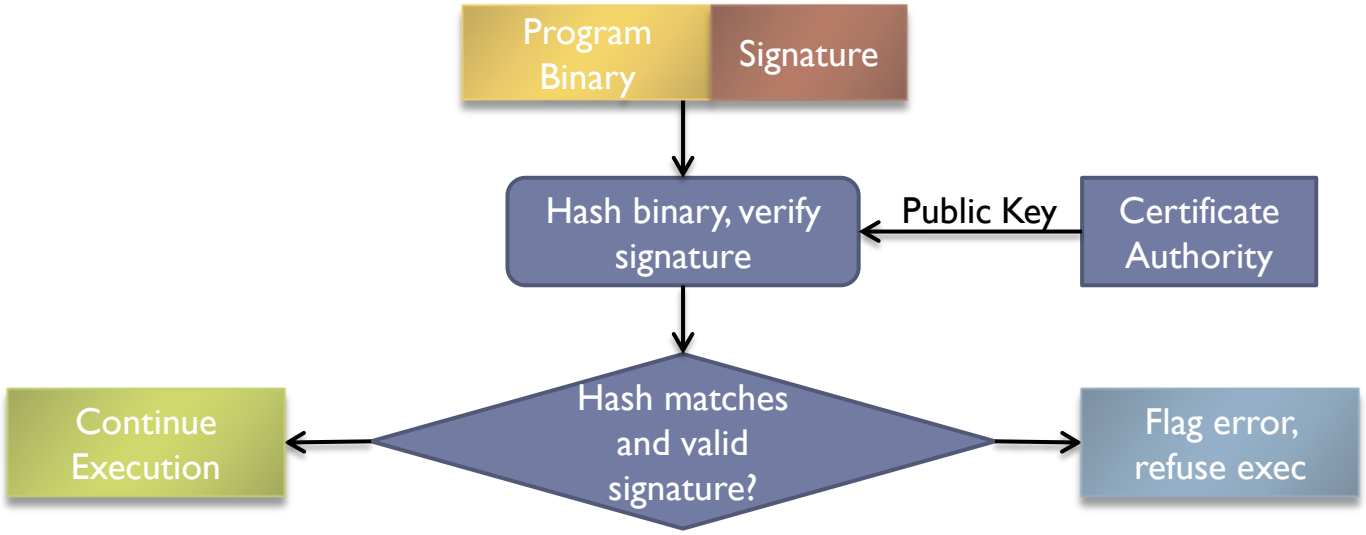
- Digital right management
- Enforcement of software license, e.g., Microsoft Office and Outlook
- Prevention of cheating in online games.



Integrity Verification

Only launch the layer that passes the integrity verification

- Load the code from the memory.
- Compute the hash value and verify the signature.
- Launch the code if the hash value matches and signature is valid.
- Otherwise, abort the boot process.



Data Encryption with TPM

Full disk encryption

- Encrypt the data with the key in TPM.
- It is difficult for any attacker to steal the key, which never leaves TPM.
- TPM can also provide platform authentication before data encryption

Application: Windows BitLocker

- Disk data are encrypted with the encryption key FVEK.
- FVEK is further encrypted with the Storage Root Key (SRK) in TPM.
- When decrypting the data, BitLocker first asks TPM to verify the platform integrity. Then it asks TPM to decrypt FVEK with SRK. After that, BitLocker can use FVEK to decrypt the data
- With this process, data can only be decrypted on the correct platform with the correct software launched.

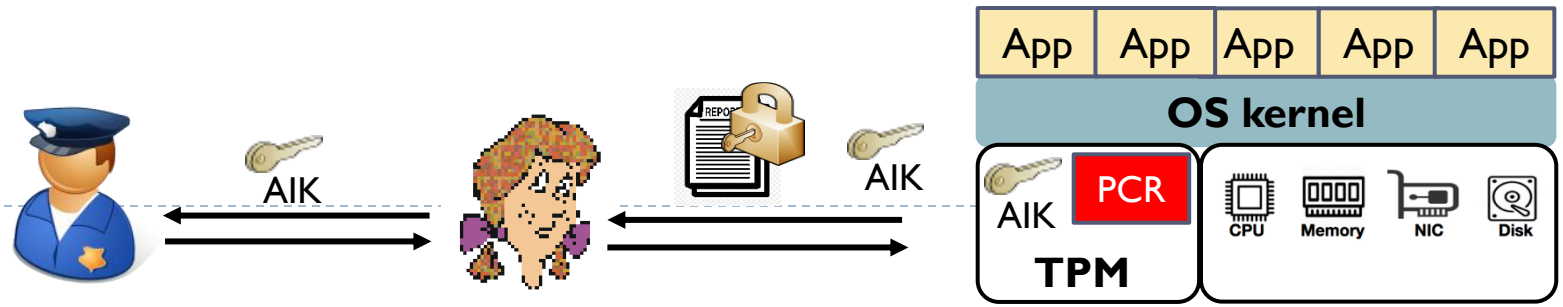
Remote Attestation with TPM

Integrity measurement architecture:

- TPM measures hash values of each loaded software, as integrity report.
- The hash values are stored in the Platform Configuration Registers (PCR) in TPM and could not be compromised by OS or any apps.

Remote attestation protocol

- TPM generates an Attestation Identity Key (AIK), to sign the hash values.
- The hash values together with AIK will be sent to client.
- A trusted third party, Privacy Certification Authority (PCA) is called to verify this AIK is indeed from the correct platform.
- Client uses this AIK to verify that received hash values are authentic.
- By checking the hash values, client knows if the loaded software is correct



Outline

- **Protection Strategies**
  - Confinement
  - Reference Monitor
- **Hardware-assisted Protection**
  - Basic Functionalities
  - Trusted Platform Module
  - Trusted Execution Environment

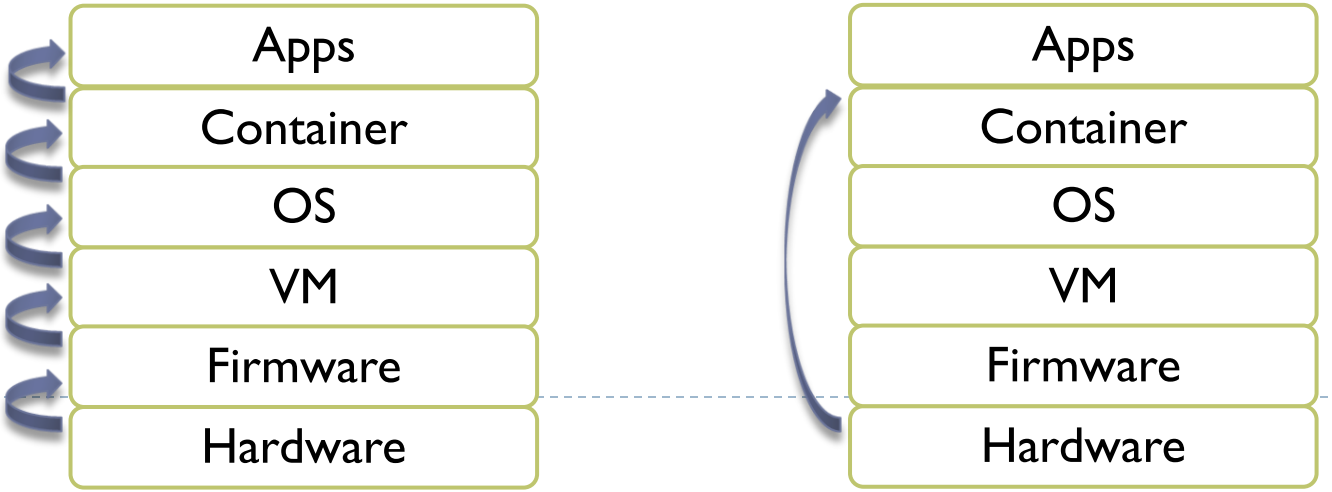
Untrusted Privileged Software

Chains of Trust can guarantee the integrity of secure booting, but not runtime security

- Even the privileged software (OS, hypervisor) is booted with integrity verification, it may still be compromised at runtime.
- How to protect applications with untrusted privileged OS or hypervisor?

Trusted Execution Environment (TEE)

- New hardware to protect the apps from untrusted OS or hypervisor.
- OS or hypervisor can support execution of apps, but not access their data



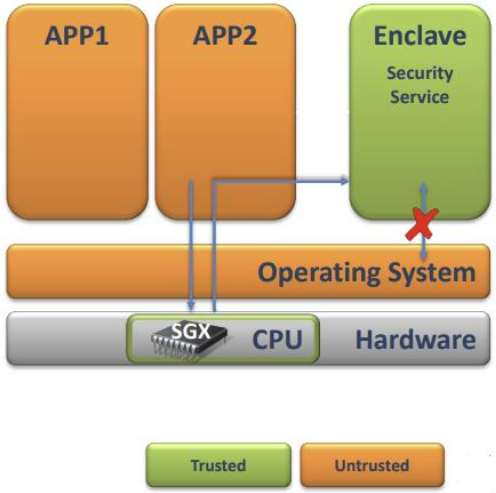
Intel Software Guard Extensions (SGX)

A security technology that safeguards application's data and code

- 2013: Intel introduced SGX in research papers
- 2015: officially launched with Intel's Skylake processor family
- 2016-2019: Improvements in SGX capabilities, expanding memory enclave sizes and strengthening security.
- 2021: SGX support removed from consumer desktop but retained in server.

Enclave

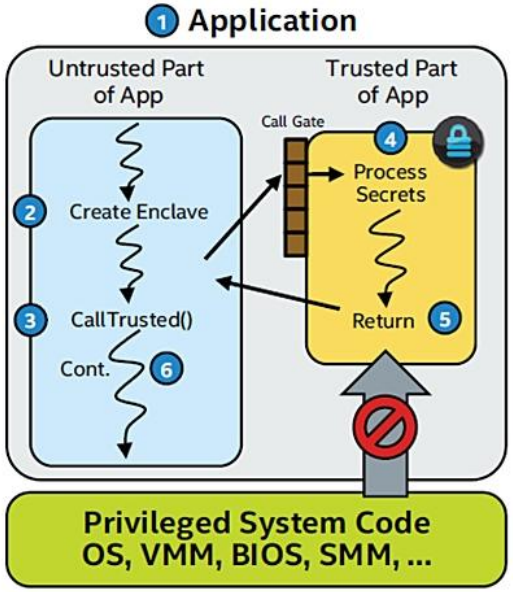
- An isolated and protected region for the code and data of an application
- Data in the enclave are encrypted by the processor when they are stored in the memory
  - Only the processor can access the data.
  - Attempts from other apps or OS will be forbidden and invoke exception



Application Execution in Enclave

The lifecycle of an application in enclave

1. An application is divided into a trusted part and an untrusted part.
2. The untrusted part creates an enclave and puts the trusted part into it.
3. When trusted code needs execution, the processor enters the enclave.
4. In the enclave, only trusted code can be executed and access the data.
5. After the code is completed, the processor exits from the enclave.
6. The untrusted part continues its execution.



Attestation with SGX

SGX also provides the attestation service.

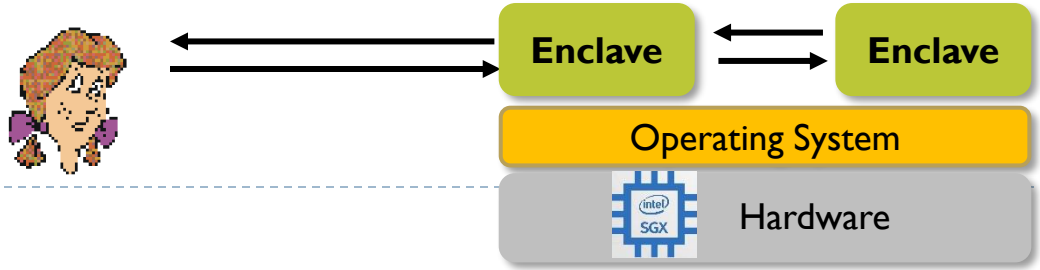
- Integrity measurement architecture: enclave measurement of the code, data, stack, heap, security flags, location of each page…
- Attestation protocol: attestation key and cryptographic protocol.

Remote attestation

- A remote client attests the integrity of the code in the enclave.

Local attestation

- In some scenarios, multiple enclaves collaborate on the same task, exchanging data at runtime.
- Collaborating enclaves have to prove to each other that they are trusted.



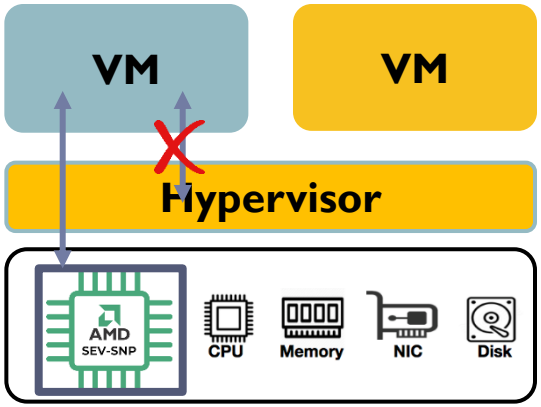
AMD Secure Encrypted Virtualization (SEV)

A hardware extension to protect VMs against untrusted hypervisor

- SEV: basic memory encryption for protecting VMs (release: 2016)
- SEV-ES (Encrypted State): encrypt CPU registers (release: 2018)
- SEV-SNP (Secure Nested Paging): adding integrity protection (release: 2020)

Mechanism

- The processor encrypts the data (memory page, registers, configurations) of the guest VMs, so the hypervisor is not allowed to access the data.
- Uses an AMD Secure Processor to manage encryption keys.
- Transparent encryption with minimal modifications to the VM.



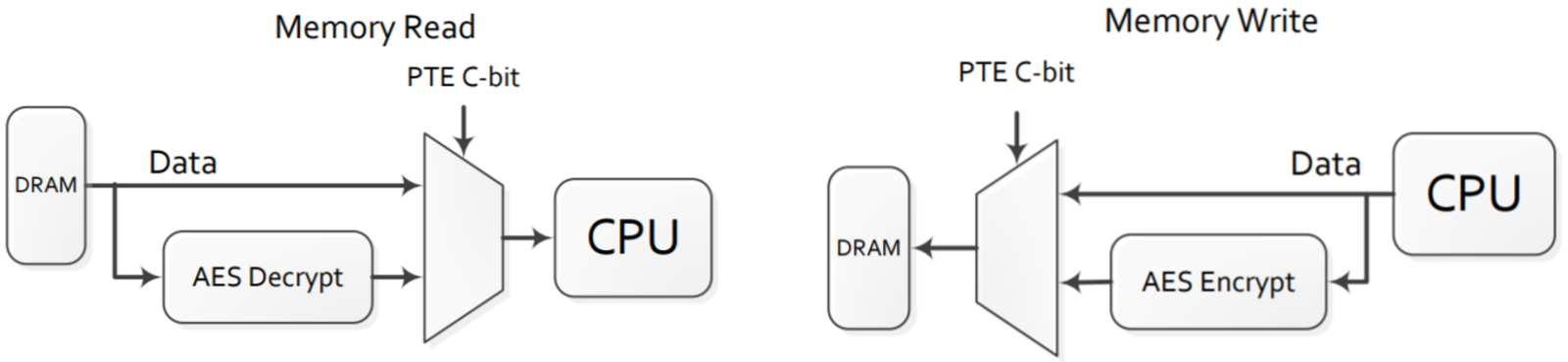
AMD Secure Memory Encryption (SME)

Virtual memory encryption is realized by SME

- An AMD architectural capability for main memory encryption
- Performed via dedicated hardware in the memory controllers
- Use AES engine to encrypt data and control with C-bit in Page Table Entry

C-bit

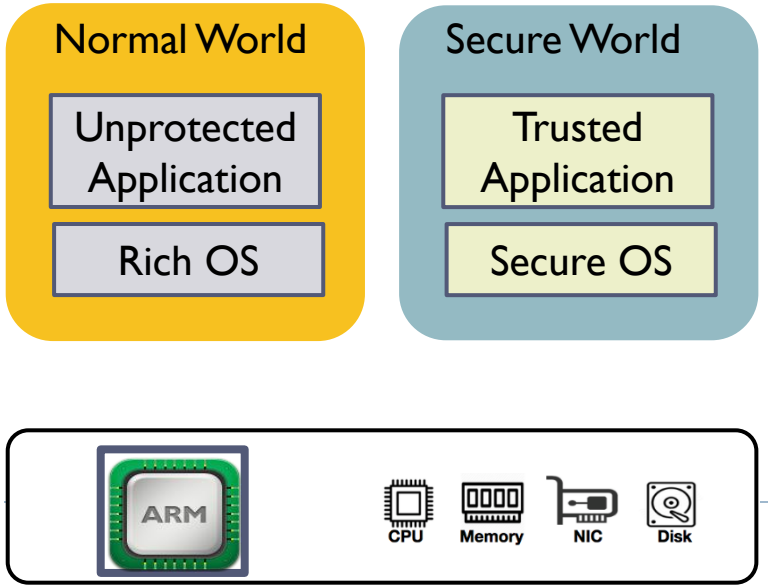
- Locate at physical address bit 47
- Set this bit to 1 to indicate this page is encrypted.
- Allow users to encrypt full memory of the VM, or selected memory pages



ARM TrustZone

The first commercial TEE processor (2003 in ARMv6 architecture)

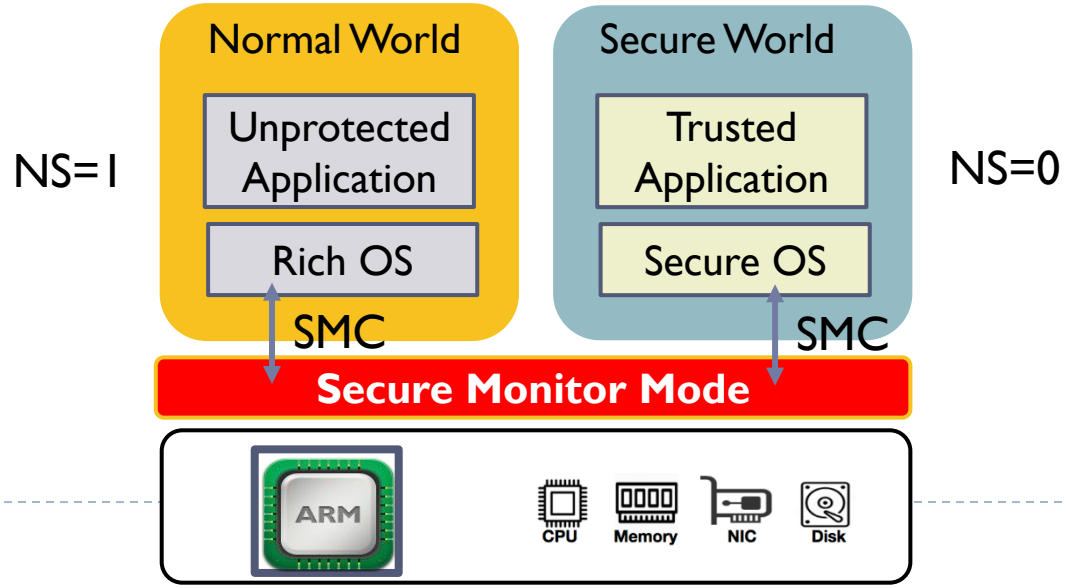
- Create two environments that can run simultaneously on the same processor. Each world has an independent OS
- Normal world: runs the normal unprotected applications and a rich OS. They have restricted access to the hardware resources in the secure world
- Secure world: runs the sensitive protected applications and a smaller secure OS, isolating them from the untrusted world. They have full access to the hardware resources in the normal world.



ARM TrustZone

Context switch

- The Non-secure bit in the Secure Configuration Register is used to determine which world the processor is currently running.
- A third privilege mode: secure monitor, in addition to user and kernel.
- When the processor wants to switch the world, it first issues a special instruction Secure Monitor Call (SMC) to enter the secure monitor mode. Then it performs some cleaning works and enter the other world.



Application of TEE: Double-edged Sword

Positive usage

- Cloud computing: you do not need to trust the cloud provider
- Digital right management
- Cryptocurrency and blockchain

Negative usage

- Adversaries leverage TEE to hide malicious activities for stealthier attacks (conflicting with malware analysis)