Virtual Machine for Malware Analysis

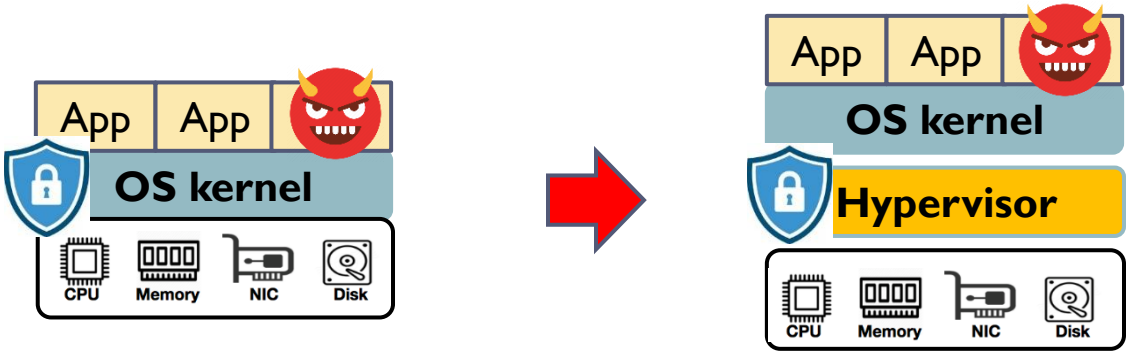
Malware analysis: deploy the malware and observe its behaviors.

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,

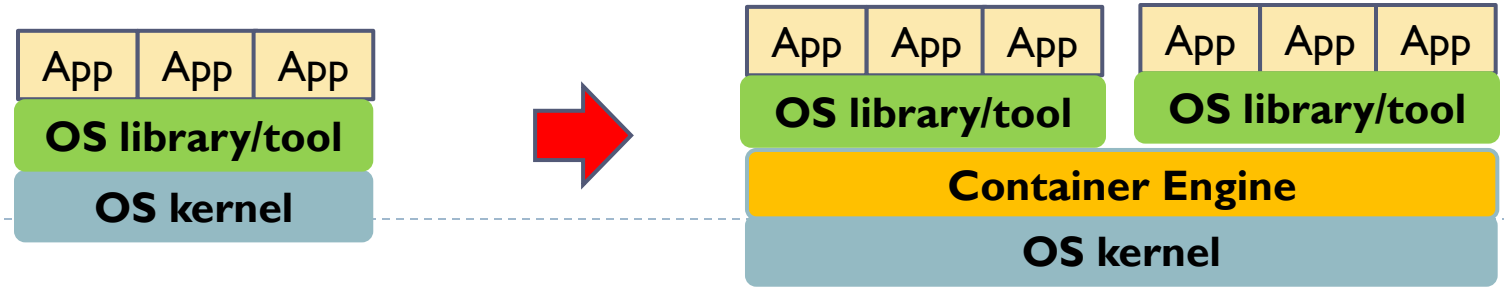
Process Level Confinement: Container

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

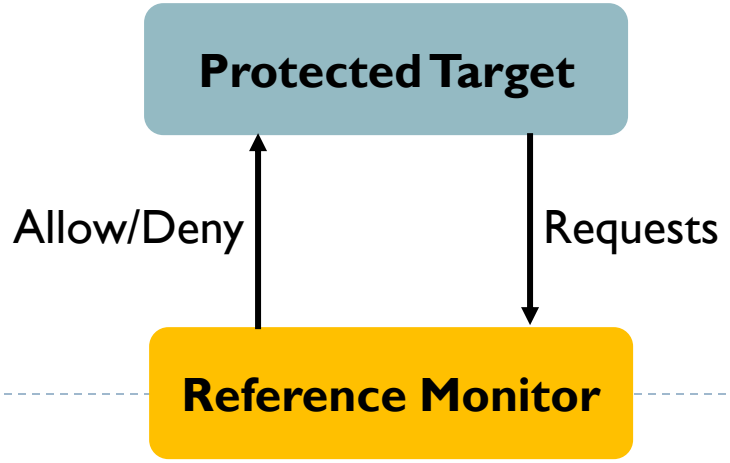
Reference Monitor (RM)

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)