Operating Systems Security
Lesson Preview

- Understand the important role an **operating system** plays in computer security
- Learn about the **need for hardware support** for isolating OS from untrusted user/application code
- Understand **key trusted computing** base concepts

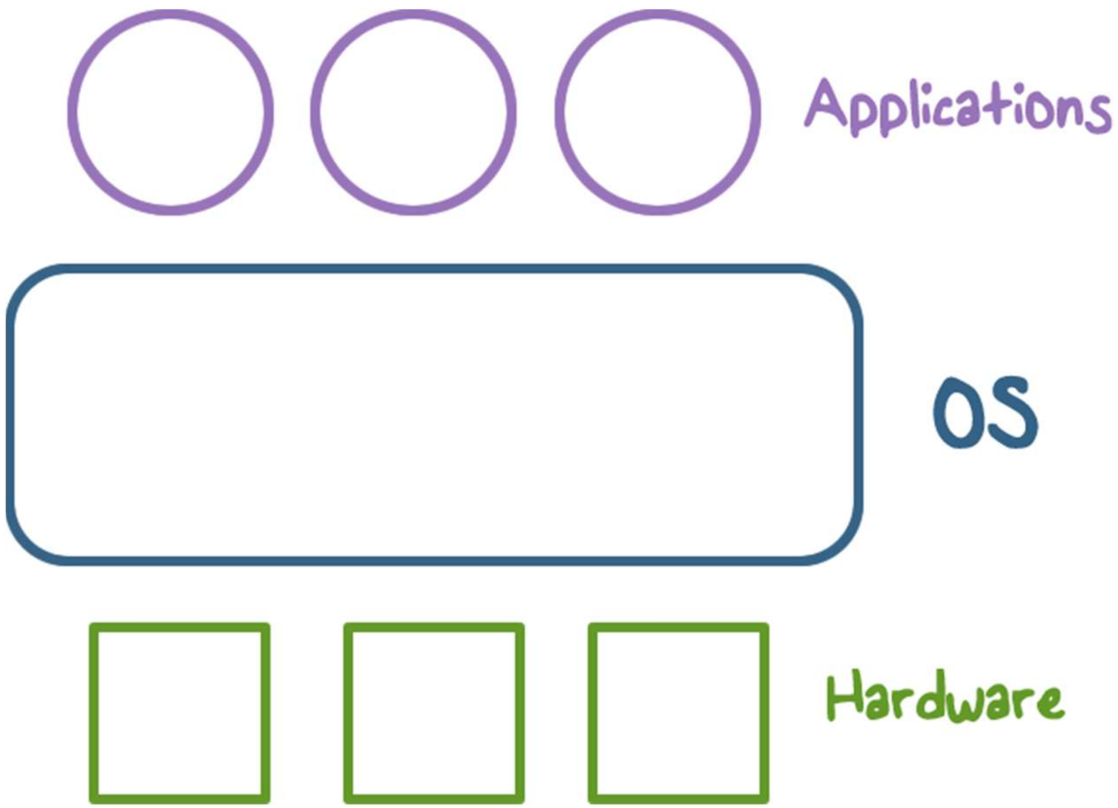
Pop Quiz

Canvas -> Quizzes -> Pop Quiz 1

Passcode: 0108

Closes at 1:40pm

Operating Systems (OS)



Operating Systems

Operating System:

- Provides easier to use and high level abstractions for resources such as address space for memory and files for disk blocks.
- Provides controlled access to hardware resources.
- Provides isolation between different processes and between the processes running untrusted/application code and the trusted operating system.

Operating Systems Hardening

- First critical step in securing a system is to secure the base operating system
- Basic steps
  - Removing unnecessary services, applications, and protocols
  - Configuring users, groups, and permissions
  - Configuring resource controls
  - Install and configure additional security controls, such as anti-virus, host-based firewalls, and intrusion detection system (IDS)
  - Test the security of the basic operating system to ensure that the steps taken adequately address its security needs

Remove Unnecessary Services, Applications, Protocols

- If fewer software packages are available to run the risk is reduced
- System planning process should identify what is actually required for a given system

- When performing the initial installation the supplied defaults should not be used
  - Default configuration is set to maximize ease of use and functionality rather than security
  - If additional packages are needed later they can be installed when they are required

Configure Users, Groups, and Authentication

- Not all users with access to a system will have the same access to all data and resources on that system
- Elevated privileges should be restricted to only those users that require them, and then only when they are needed to perform a task

- System planning process should consider:
  - Categories of users on the system
  - Privileges they have
  - Types of information they can access
  - How and where they are defined and authenticated
- Default accounts included as part of the system installation should be secured
  - Those that are not required should be either removed or disabled
  - Policies that apply to authentication credentials configured

Configure Resource Controls

- Once the users and groups are defined, appropriate permissions can be set on data and resources
- Many of the security hardening guides provide lists of recommended changes to the default access configuration

Install Additional Security Controls

- Further security possible by installing and configuring additional security tools:
  - Anti-virus software
  - Host-based firewalls
  - IDS or IPS software
  - Application white-listing

Test the System Security

- Final step in the process of initially securing the base operating system is security testing
- Goal:
  - Ensure the previous security configuration steps are correctly implemented
  - Identify any possible vulnerabilities

- Checklists are included in security hardening guides
- There are programs specifically designed to:
  - Review a system to ensure that a system meets the basic security requirements
  - Scan for known vulnerabilities and poor configuration practices
- Should be done following the initial hardening of the system
- Repeated periodically as part of the security maintenance process

Need for Trusting an Operating System

Why do we need to **trust** the operating system?
(AKA a **trusted computing base or TCB**)
What requirements must it meet to be trusted?

TCB Requirements:

1. Tamper-proof,
2. Complete mediation, and
3. Correct

TCB and Resource Protection

TCB Controls access to protected resources

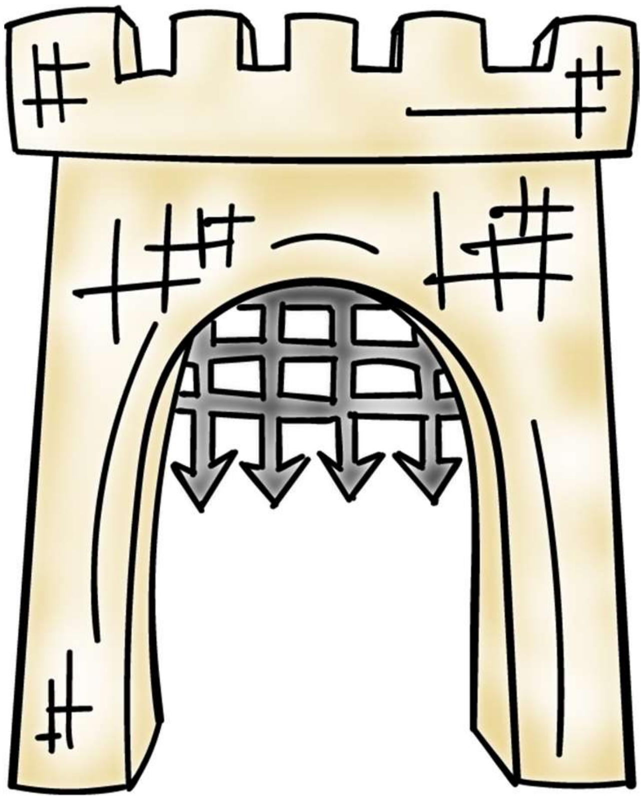
- Must establish the source of a request for a resource (authentication is how we do it)
- Authorization or access control
- Mechanisms that allow various policies to be supported

Isolating OS from Untrusted User Code

How do we meet the first requirement of a TCB (e.g., isolation or tamper-proof)?

- Hardware support for memory protection
- Processor execution modes (system AND user modes, execution rings)
- Privileged instructions which can only be executed in system mode
- System calls used to transfer control between user and system code

System Calls: Going from User to OS Code



System calls used to transfer control between user and system code

- Such calls come through **"call gates"** and return back to user code. The processor execution mode or privilege ring changes when call and return happen.
- x86 Sysenter/sysexit instructions

Isolating User Processes from Each Other

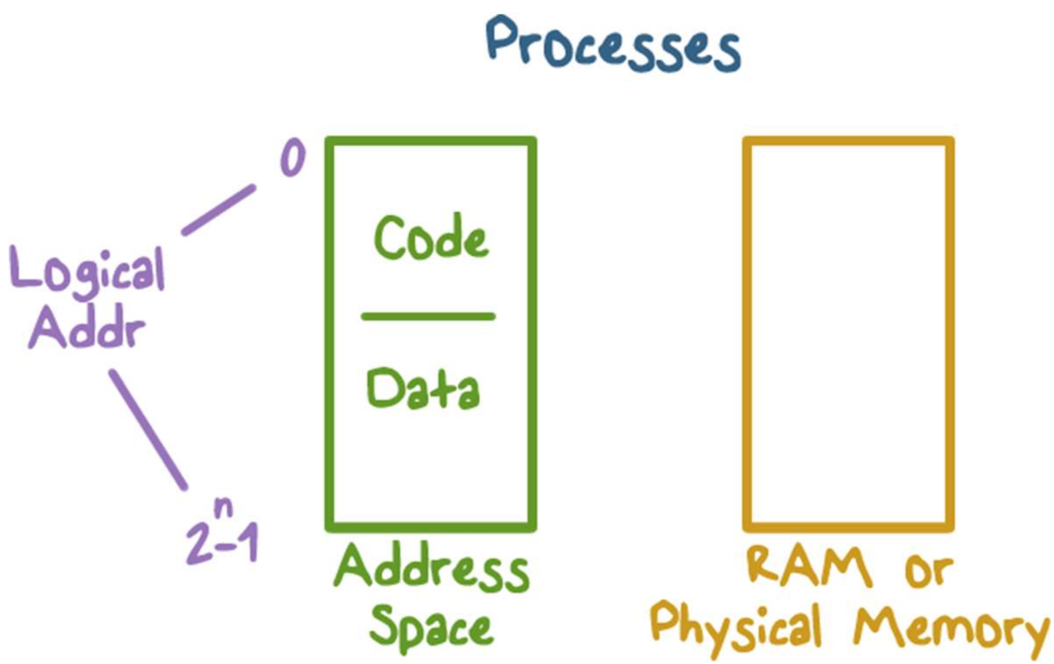
How do we meet the user/user isolation and separation?

OS uses hardware support for memory protection to ensure this.

Address Space: Unit of Isolation

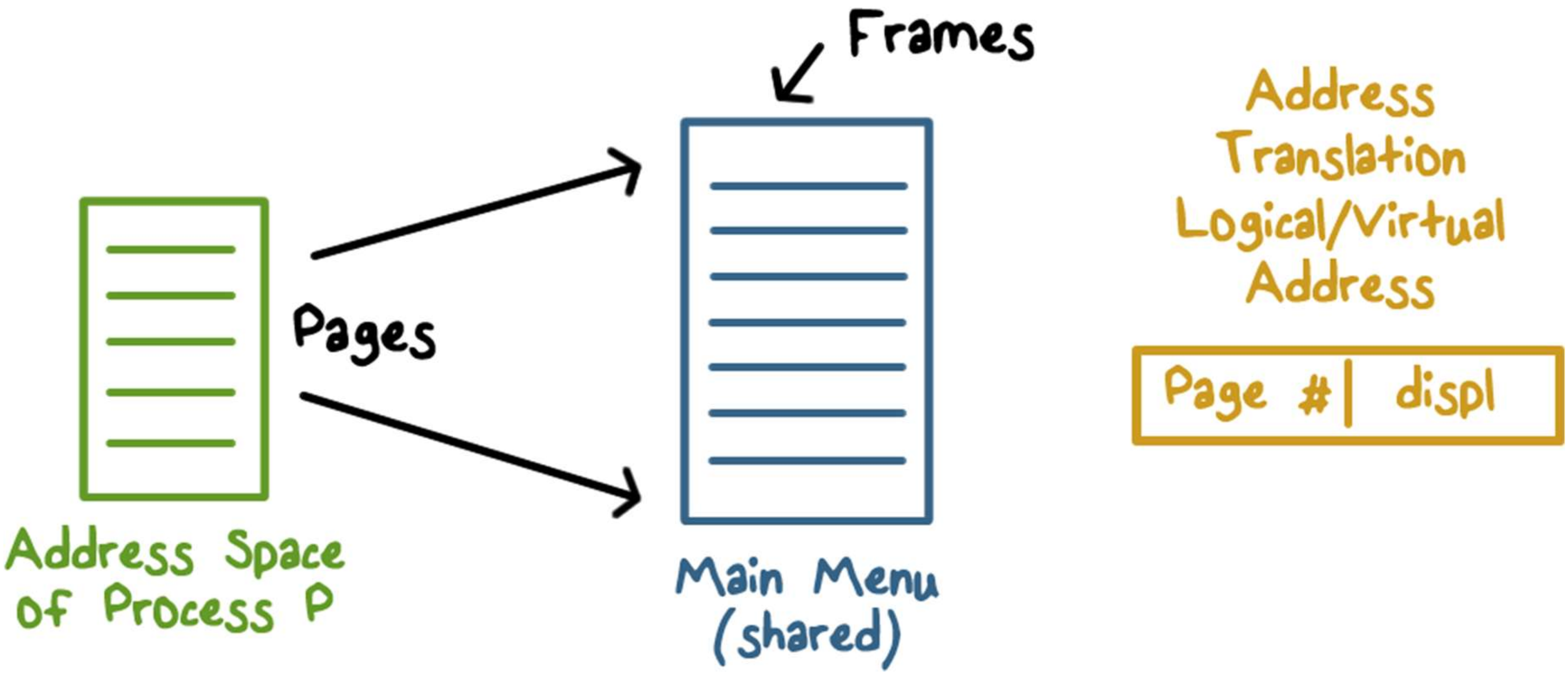
Processes view memory as **contiguous often larger than available physical memory**

- Usually 2^{32} or 2^{64} addresses
- Each process has its own mapping



Address Translation

Operating system **maps logical virtual addresses or pages onto physical memory frames**

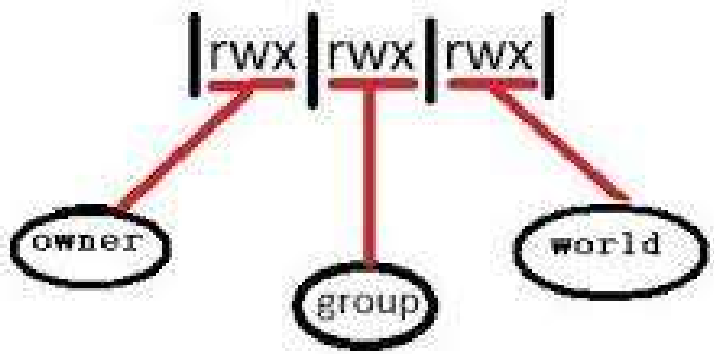


Process Data/Code Protection

OS will not map a **virtual page of process A** to a **physical page of process B** unless **explicit sharing** is desired.

- Process A **cannot access** process B’s memory because it has no way to name/reach its memory.
- Page tables **managed by OS**

Process Protection through Memory Management



- Processor memory management unit (MMU) **uses page tables to resolve virtual addresses to physical addresses.**

- RWX bits on pages **limit type of access** to addressable memory

Preventing Malicious Code Execution on the Stack through a Non-Executable Stack

Now think, how can we do a non-executable stack to **help prevent code injection via stack buffer?**

- **Used by Windows, OS X, Linux**

OS Isolation from Application Code

- OS (Kernel) resides in a portion of each process's address space.
- True for each process, **processes can cross the fence only in controlled/limited ways.**

OS Isolation from Application Code

Linux, DOS, OS X

- 32-bit Linux: Lower 3GB for user code/data, top 1GB for kernel
- Corresponds to x86 privilege ring transitions
- Windows and OS X similar
- DOS had no such fence, **any process could alter DOS and viruses could spread by hooking DOS interrupt handlers via kernel changes**

Complete Mediation:
The TCB

- Make sure that no protected resource (e.g., memory page or file) could be accessed without going through the TCB

- TCB acts as a reference monitor that cannot be bypassed

Complete Mediation: User Code

- User code cannot access OS part of address space without changing to system mode

- User code cannot access physical resources because they require privileged instructions (e.g. servicing interrupts) which can only be executed in system mode

Complete Mediation: OS

- OS virtualizes physical resources and provides an API for virtualized resources
- File for storing persistent data on disk
- Virtual resource must be translated to physical resource handle (e.g., file buffers) which can only be done by OS, which ensures complete mediation

Virtualization

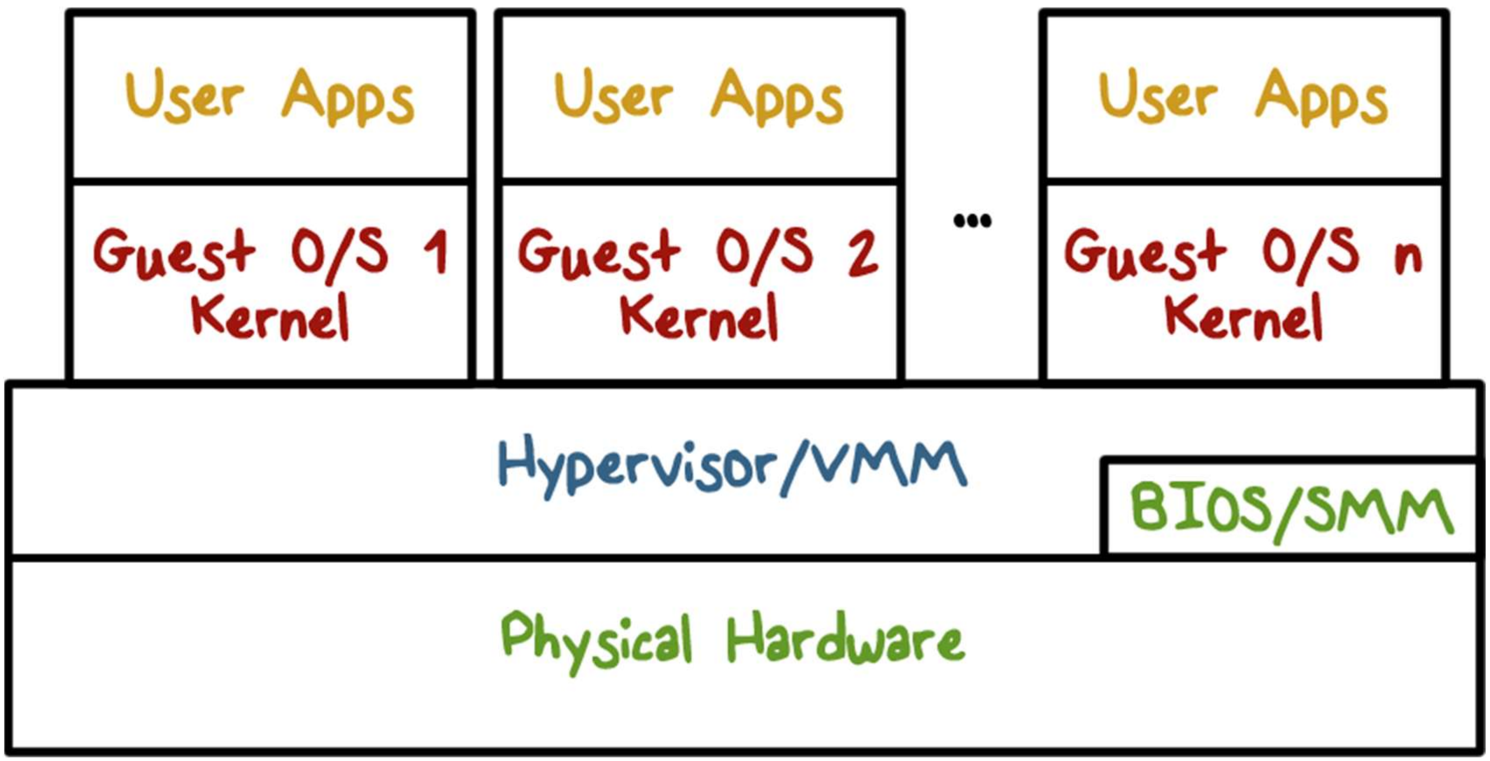
- OS is large and complex, even different operating systems may be desired by different customers
- Compromise of an OS impacts all applications

Limiting the Damage of a Hacked OS

Use: Hypervisor, virtual machines, guest OS and applications

Compromise of OS in VM1 **only impacts applications running on VM1**

Virtualization Security Layers



What is the TCB?

Correctness:
The Final TCB Requirement

- Compromise of OS (TCB) means an **attacker has access to everything.**
- Getting the TCB right is extremely important
- **Smaller and simpler** (hypervisor only partitions physical resources among VMs and let us guest OS handle management)
- **Secure coding** is really important when writing the OS which typically is written in languages that are not type safe

Operating Systems Security
Lesson Summary

- Understand the important role an OS plays in **protecting resources and applications**
- Understand how OS is **isolated from untrusted code** with hardware support for memory management
- Understand how **complete mediation** is provided.