Operating System Security

Operating Systems Defined
Hardware: I/o...Memory….CPU
Operating Systems: Windows or Android, etc
Applications run on operating system

Operating Systems
-Makes it easier to use resources. Allows for high level abstractions - like files
-Hardware is controlled by the OS
-Provides isolation - (each process believes it is the only one running on the system)

Need for Trusting an OS
TCB = trusted computing base
The operating system has direct control of the hardware resources.
The OS must determine who is an authorized user of the resources.

TCB Requirements:

1. Complete mediation - the OS is between the hardware resources and applications. The OS must make sure the application has the necessary authorizations.
2. The OS must be tamperproof.
3. The OS must be correct-- the protected resources are used properly.

OS and Resource Protection
OS controls access to protected resources.
To do this it must:

- establish the source of the request (authentication)
- Authorization or access control - does the source of the request have the right to access the resource.
- the OS follows the policies for authorization and authentication

System Calls
system calls - a request to the operating system, often called a protected procedure call
A system call has a higher cost than a regular procedure call

Complete Mediation
complete mediation - ensures that the OS cannot be bypassed when accessing a protected resource.
To determine if the source making the request has authorization access the resource.
The OS checks the authorization of the user who requested the execution of the process.

How Can we Trust an OS

How to meet the requirement for isolation

- requires hardware support for memory protection
- the processor must keep track of what kind of code is being executed (user or system modes or execution rings)
- privileged instructions can only be executed in system mode

System Calls

Go from user mode to system mode,
System calls are used to transfer control between user and system code

- the calls come through “call gates” and return back to the user code. The processor execution mode or privilege ring changes when call and return happen.

Call Gates - transition from user to system level.
Must keep track of return location
The x86 systems have sysenter and sysexit instructions

System calls are more expensive because of the information that must be saved, the memory mapping that must be done, and the special instructions

Untrusted User Code Isolation

How is untrusted user code handled? It must be isolated.
Rely on the hardware to protect memory.

- Memory protection - the hardware determines if memory belongs to the OS, and is therefore unwritable to users.

User Isolation Quiz

Sometimes the secure memory for the OS can be modified by hackers.

- Mac’s Thunderbolt interface was hacked
- The refresh mechanism on DRAMs have been exploited.

Address Space: Unit of Isolation

What data or code can a certain process access?
Each process gets an address space for it to use - it is a unit of isolation.

Each process will view its address as contiguous. It may be more than the available physical memory. For 64 bit machines the address space is 2^{64} bits.
Each process has its own mapping.

Physical addresses point to actual RAM or physical memory.
Logical addresses point to the address space.
There needs to be a translation between the two, with isolation between the processes must be maintained.

Address Translation

The translation between the logical and physical memory is performed through the use of pages and frames.

The logical addresses are stored on pages.
The physical addresses are stored on frames.

Then there is a page table to translate between the pages and frames.
The OS builds and protects the page table.

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, as long as the page tables are protected and managed by the OS.

Process Protection through Memory Management
Memory management unit (MMU) handles the memory mapping. It uses page tables to resolve virtual addresses to physical addresses.

TLB - translation lookaside buffers store the translation tables.
RWX - read, write, execute -- the accesses for each source
RWX are bits on the pages which determine the level of access to addressable memory

So the MMU determines where the process can go in memory and what the process can do with that memory.

Stack Overflow Quiz - revisited
The stack can be exploited through:

- overflowing the buffer to change the return address to alter program execution:
    - the return address must be altered to point to the shell code
    - the return address in the shell code must point to the original return address.
    - the shell code must be installed somewhere reachable on the system

Preventing Malicious Code Execution on the Stack through a Non-Executable Stack
How can we use a non-executable stack to help prevent code injection via the stack buffer?

If the stack is non-executable, instructions can be injected on the stack but they cannot be executed. The stack is only readable and writable. (RW) The page table entries for this portion of the stack have their execution permissions turned off.

OS Isolation from Application Code
The same methods can be used to isolate the OS from application code.

-The OS(kernel) resides in a portion of each process's address space.
-This is true for each process, processes can cross the fence only in controlled and limited ways. When the process accesses the user address portion of memory, the process cannot access the OS portion.

The OS has to manage the page tables.
The OS will protect itself and the processes from each other.

- 32-bit Linux: the lower 3GB for user code/data, top 1GB for kernel
-Corresponds to x86 privilege ring transitions
-Windows and OSx are similar

DOS did not have the 'fence'. Any process could alter DOS and viruses could spread by hooking DOS interrupt handlers via kernel changes.

Executing Privilege Level
These tasks should be performed by the OS:
- Switching CPU from one process to another when the process blocks
- Page fault handling
- Changing who can access a protected resource such as a file

This task should be performed by the user processes:
- Setting up a new stack frame when an application program calls one of its functions

Complete Mediation: The TCB
To implement complete mediation:
- make sure that no protected resource can be accessed without going through the TCB (kernel).
- The TCB acts as a reference monitor that cannot be bypassed.

Complete Mediation: User Code
- User code cannot access the OS part of the address space without changing to system mode. The user code needs to execute a system call
- USer code cannot access physical resources because they require privilege instructions 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 handling which can only be done by the OS, which ensures complete mediation.

Virtualization

With regards to security and protection:

- the OS is large and complex. Even different OS may be desired by different customers.
- Compromise of an OS impacts all applications.

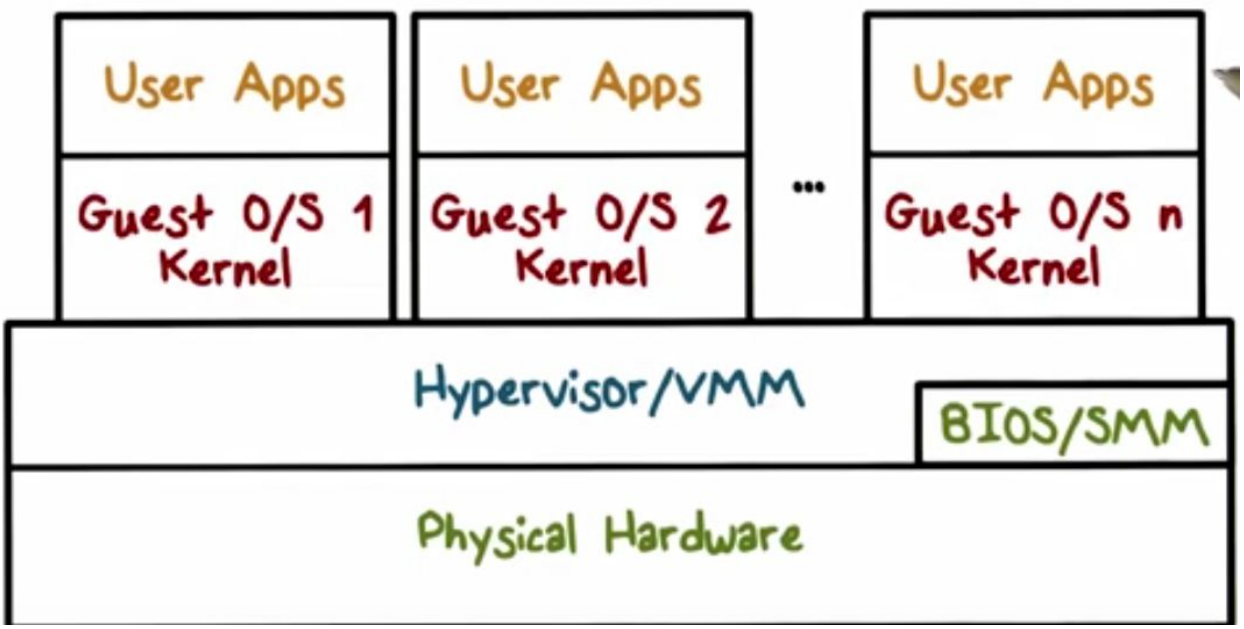
Limiting the Damage of a Hacked OS

Virtualization helps with limiting the damage caused by a compromised OS.

- Use a hypervisor - this is inserted between the OS and the hardware.
- On top of the hypervisor are virtual machines - that have their own OS (called a guest OS) and applications.

A compromise of the OS in VM1 only impacts applications running in VM1.
This leads to isolation between virtual machines. So do taxes on VM1 and browse the internet on VM2.
In this case the TCB is the hypervisor.

Virtualization Security Layers



VMM - virtual machine monitor

each guest OS is a different Virtual Machine.
Compromise of one OS will not affect the others.

Correctness: The Final TCB Requirement

- Compromise of OS (TCB) means an attacker has access to all resources on the system.
- Getting the TCB is extremely important - but difficult to achieve
- Make the TCB smaller and simpler - for example the hypervisor only partitions the physical resources among the different VMs and let the guest OS handle the management.
- Secure coding is really important when writing the OS. The OS is typically written in languages that are not type safe.

TCB Requirements Quiz
If a hacker turns off the check that is performed before access to a protected resource is granted, what TCB requirement is violated?
Tamper-proof requirement is violated.

Size of Security Code Quiz
The larger the size the harder to achieve correctness.
Windows OS is 10,000 times larger than MS DOS - millions of lines of code.

The vulnerabilities in OS are due to the high complexity and size of the OS.

Hypervisor Code Size Quiz
How many lines of code are in a typical hypervisor?
150,000 lines of code.