Contents

1 Reference Monitor

An access control concept that refers to an abstract machine that **mediates all access** to objects by subjects

- Must be tamper proof / resistant
- Must **always be invoked** when access to an object is required
- Must be small enough to be verifiable / subject to analysis to ensure correctness

2 Placement

Can be placed anywhere within the system, a variety of locations relative to the program being run

- Hardware Dedicated registers for defining privileges
- Operating system kernel E.g. Virtual Machine Hypervisor
- Operating system Windows security reference monitor
- Services Layer JVM, .NET
- Application Layer Firewalls

2.1 Lower Is Better

Using a reference monitor or other security features at a **lower level** means:

- We can assure a higher degree of security
- Usually simple structures to implement
- Reduced performance overheads
- Fewer layer below attack possibilities

However: **access control decisions** are far removed from applications

3 OS Integrity

The operating system: arbitrates access requests, is itself an resource that must be accessed. This is a **conflict**, we want to **use** the OS but **not mess with it**.

3.1 Modes of operation

Defines which actions are permitted in which mode, e.g. system calls, machine instructions, I/O. Distinguish between computations done on behalf of: **the OS and the user**. A *status flag* within the CPU allows the OS to operate in different modes

3.2 Controlled Invocation

Many functions are held at kernel level, but are quite reasonably called from within user level code

- Network and File IO
- Memory allocation
- Halting the CPU (at shutdown only!)

We need a mechanism to transfer between kernel mode (ring 0 - root) and user mode (ring 3 - user)

3.2.1 Controlled Invocation: Interrupts

Also known as exceptions / traps. In many ways is the **hardware equivalent** to a **software exception** not always bad. Handled by an *interrupt handler* which resolves the issue and returns to the original code. Given an interrupt, the CPU will **switch execution** to the location given in an *interrupt descriptor table*

4 Descriptors and Selectors

- Descriptors hold information on crucial system objects like **kernel structure locations**
- Descriptors are held in *descriptor tables*
- Contain a Descriptor Privilege Level (DPL)
- Descriptors are indexed by selectors
- Loaded when required, e.g. on jump calls
- The CPU protects the kernel by **checking** the Current Privilege Level (CPL) when a Selector is loaded

4.1 Interrupt-gates

- The code segment (CS) register in x86 CPUs has 2 bits reserved for the Current Privilege Level (CPL)
- Descriptors that have a privilege level **higher than where they point are called** gates
- Since these descriptors are **created by the kernel**, they offer a **secure means of entry into ring 0**

4.2 Patching the kernel

If you can run custom PL 0 code (compromised driver?), you can insert your own handler *Rootkit*

5 Processes and Threads

- A process is a program being executed. Important unit of control:
  - Exists in its own address space
  - Communicates with other processes via the OS
  - Separation for security
- A Thread is a strand of execution within a process. Shares a common address space

6 Memory Protection

Segmentation divides data into logical units

- Good for security
- Challenging memory management
- Not used much in modern OSs

Paging divides memory into pages of equal size

- Efficient memory management
- Worse for access control
- Extremely common in modern OSs

7 Side channel exploits

In most operating systems, the entire kernel is stored in the **upper address space**. Pages in this area are flagged as supervisor, and cannot be accessed in ring 0. In Intel (and some other) CPUs, its common to **speculatively evaluate** code prior to reaching it. If the branch isn't taken, changes are rolled back, however cache isn't.

7.1 Meltdown

Meltdown breaks the mechanism that keeps applications from accessing arbitrary system memory. Consequently, applications can access system memor

7.2 Spectre

Spectre tricks other applications into accessing arbitrary locations in their memory

Reference section

placeholder