Reference Monitors

Enforcement of Access Control

Overview of Today's Lecture:

- Introduction
- Operating System Integrity
- Hardware Security Features
- Protecting Memory

Introduction:

Fundamental Concepts:

- *Reference Monitor* – an abstract concept
- *Security Kernel* – its implementation
- *Trusted Computing Base (TCB)* – kernel + other protection mechanisms

Reference Monitor (RM):

"*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 object required
- Must be small enough to be *verifiable* / subject to analysis to ensure its *correctness*

Security Kernel:

"*The hardware, firmware, and software elements of a TCB that implement the reference monitor.*"

- Must *mediate all access*
- Must be *protected from modification*
- Must be *verifiable* for *correctness*
- Ideally in the bottom layers of a system

Trusted Computing Base (TCB):

"*The totality of protection mechanisms within a computer system responsible for enforcing a security policy*"

- One or more components
- Enforce a unified security policy over a product or system
- Correct enforcement depends on components within
- and input by system administrators

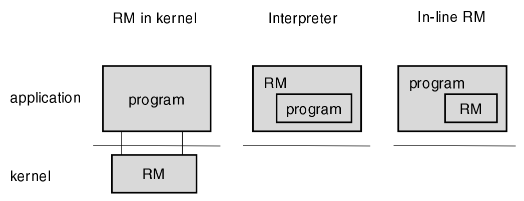
Reference Monitor Placement:

Can be placed anywhere

- Hardware
  - Dedicated registers for defining privilages
- Operating System Kernel
  - Virtual machine emulating users
- Operating System
  - Windows Security Reference Monitor
- Services Layer
  - JVM, .NET
- Application
  - Firewall

Reference Monitor Placement:

In relation to application it should control:



Operating System Integrity:

- OS is not only the arbitrator of access requests
- OS is itself an object of access control

"Users must not be able to modify the operating system"

- Users should be able to use the OS
- Users should not be able to misuse the OS

Modes of Operation:

Distinguish computations done "on behalf of":

- the OS
- the user

A *Status flag* allows the OS to operate in different *modes*.

e.g. In Unix – *supervisor (root)* and *user* modes

Controlled Invocation:

- User requiring *supervisor* mode for an operation
- Processor switches between modes
- Only predefined set of operations performed in *supervisor* mode
- System returns to *user* mode

Hardware Security Features:

Reasons for placing security in lower system levels:

- Possibility to evaluate security to a higher degree
  - *reasonably simple structures*
  - *security mechanism compromised if layer below attacked*
- Performance overheads reduced
- Access control decisions far removed from decisions made by applications

Memory Structures:

Security characteristics of memory structures:

1. *RAM* – (*R/W*) - Cannot guarantee integrity or confidentiality
2. *ROM* – built-in integrity guarantee, good for storing parts of an OS
3. *EPROM* – useful for storing parts of OS or crypto keys, advanced attacks may pose a threat
4. *WROM* – good for storing crypto keys, disks used for audit trail logs

continued…

Volatile memory

- loses its contents on power off
- neither instantaneous nor complete
- reconstructable using special electronics
- *defence* – repeated overwrites

Non-volatile (permanent) memory

- if attacker has access by bypassing CPU
- further measures required (e.g. cryptography)

continued…

Memory

- main memory
- cache
- buffers
- etc..

Data object may exist simultaneously in *more than one* location!

Copy held in an unprotected memory = *risk*

Processes and Threads:

Process – program in execution, important unit of control in an OS and for security

- Works in its own address space
- Communicates with other processes with help of OS
- Separation useful for security

Thread – a strand of execution within a process

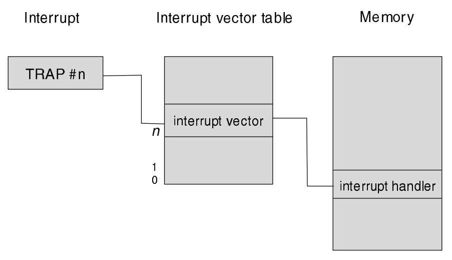
Controlled Invocation - Interrupts:

Exceptions/Interrupts/Traps

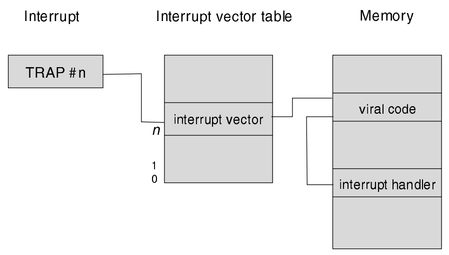
Interruptions of executions due to errors, user request, hardware failure, etc…

- Handled by CPU
- Improper handling leads to security flaws
  - *CTRL-C during supervisor mode operations*
  - *Interrupt table entry change*

Processing Interrupt:



Processing Interrupt:



Protecting Memory:

- *OS integrity* – preserved by separation of user & kernel space

Separation of users:

- File management – logical memory object
- Memory management – physical memory objects

continued…

Segmentation – divides data into logical units

- Good basis for enforcing security policy
- Variable length – difficult memory management

Paging – divides memory into pages of equal size

- Popular – efficient memory management
- Not good for access control
- A page might contain objects requiring different protection

Secure Addressing:

- Confinement of processes to separate address spaces
- Control access to data objects in memory

1. OS modifies addresses received from user
   (address sandboxing e.g. mask)
2. OS constructs effective addresses from relative ones
   (relative addressing)
3. OS checks whether address within given bounds
   (base register addressing)

Summary:

- How Access Control is enforced
- Why OS integrity is important
- Security features of existing hardware
- How to control access to memory

End