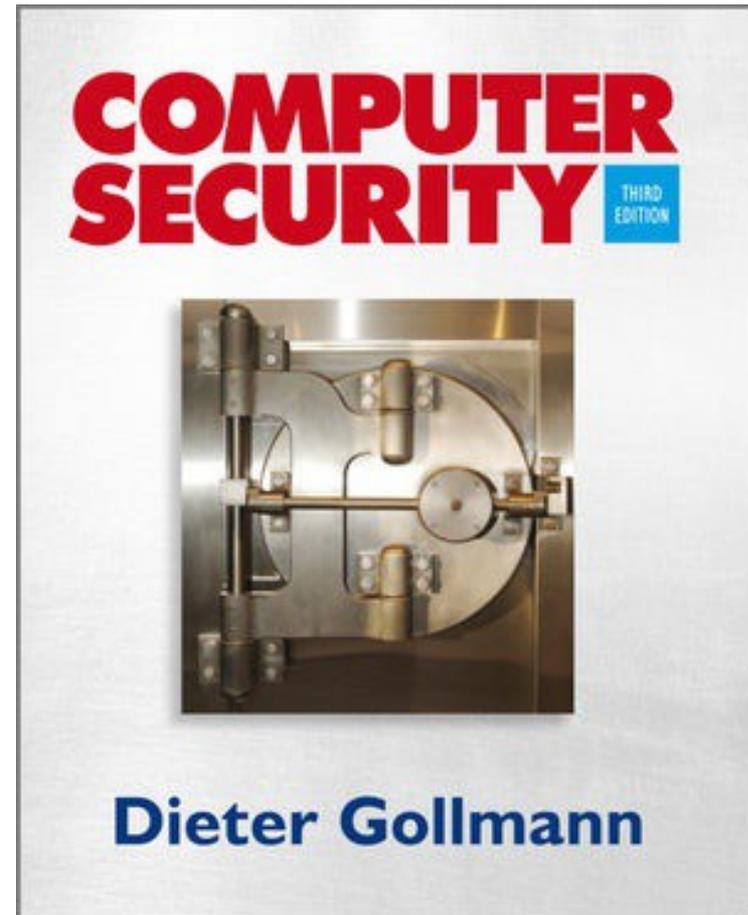


Computer Security 3e

Dieter Gollmann



Chapter 6: Reference Monitors

Agenda

- Reference monitor, security kernel, and TCB
 - Placing the reference monitor
- Status information & controlled invocation
- Security features in microprocessors
 - Confused deputy problem
- Memory management and access control
- Historic examples, to keep matters simple

Security Mechanisms

- How can computer systems enforce operational policies in practice?
- Questions that have to be answered:
 - Where should access control be located?
(Second Fundamental Design Decision)
 - Are there any additional security requirements your solution forces you to consider?
- The following definitions are taken from the Orange Book.

Reference Monitor (RM)

- **Reference monitor**: access control concept that refers to an abstract machine that mediates all accesses to objects by subjects.
- **Security Kernel**: hardware firmware, and software elements of a TCB that implement the reference monitor concept. It must mediate all accesses, be protected from modification, and be verifiable as correct.

Trusted Computing Base (TCB)

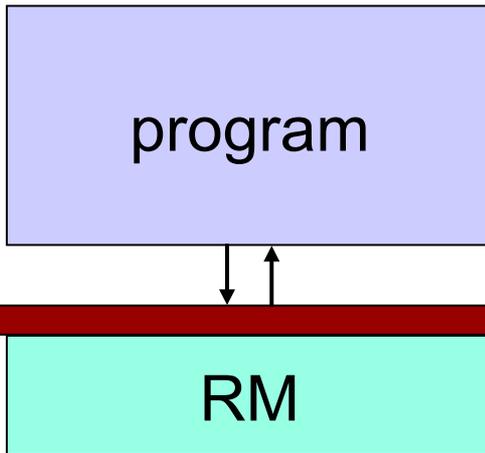
- The totality of protection mechanisms within a computer system – including hardware, firmware, and software – the combination of which is responsible for enforcing a security policy.
- A TCB consists of one or more components that together enforce a unified security policy over a product or system.
- The ability of the TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters related to the security policy.

Placing the RM

- **Hardware**: access control mechanisms in microprocessors
- **Operating system kernel**: e.g. **hypervisor**, i.e. a virtual machine that emulates the host computer it is running on.
- **Operating system**: e.g. access control in Unix and Windows 2000.
- **Services layer**: access control in database systems, Java Virtual Machine, .NET Common Language Runtime, or CORBA middleware architecture.
- **Application**: security checks in the application code to address application specific requirements.

RM – Design Choices

kernel supported
(e.g. in O/S)



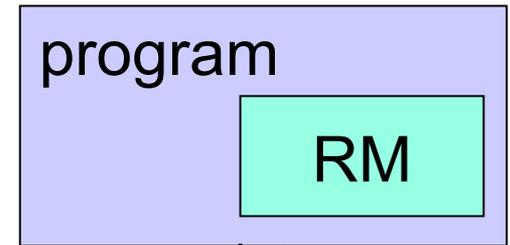
kernel

interpreter



kernel

modified
application (IRM)



kernel

Operating System Integrity

- Assume that your O/S prevents unauthorized access to resources (as long as it works as intended).
- To bypass protection, an attacker may try to disable the security controls by modifying the O/S.
- Whatever your initial concern was, you are now facing an **integrity problem**. The O/S is not only the arbitrator of access requests, it is itself an object of access control.
- **New security policy: Users must not be able to modify the operating system.**
- This generic security policy needs strong and efficient support.

Operating System Integrity

- To make life more complicated, you have to address two competing requirements.
 - Users should be able to use (**invoke**) the O/S.
 - Users should not be able to misuse the O/S.
- Two important concepts commonly used to achieve these goals are:
 - **status information**
 - **controlled invocation**, also called **restricted privilege**
- These concepts can be used in any layer of an IT system, be it application software, O/S, or hardware.

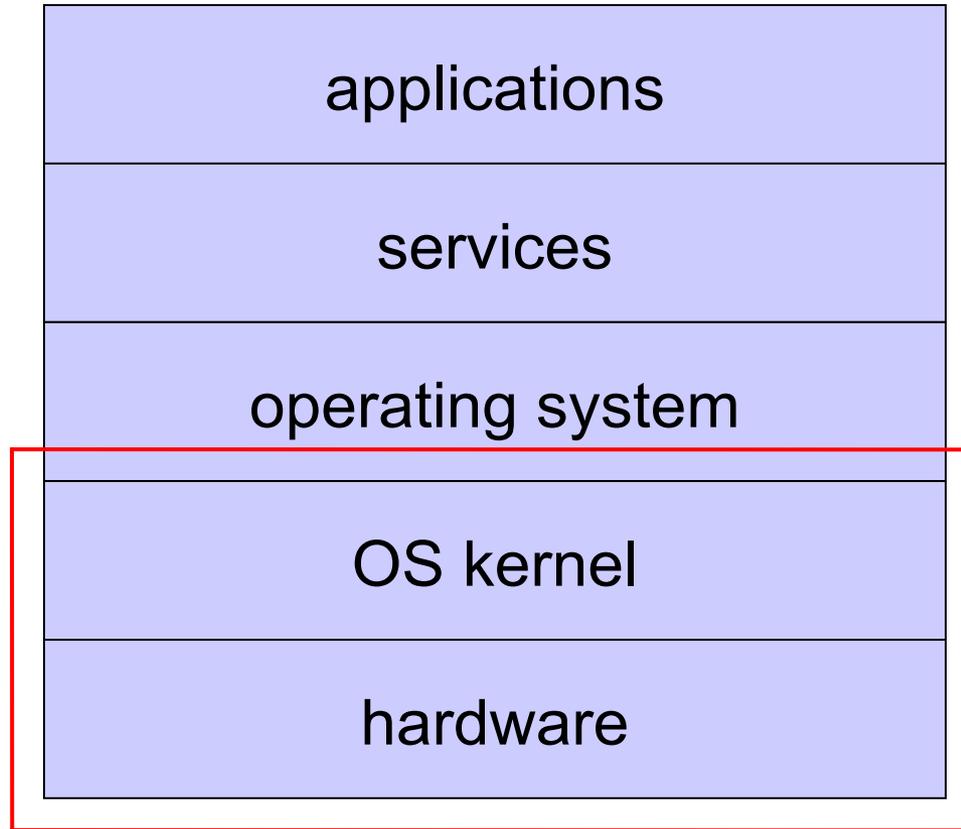
Modes of Operation

- To protect itself, an O/S must be able to distinguish computations ‘on behalf’ of the O/S from computations ‘on behalf’ of a user.
- **Status flag** allows system to work in different **modes**.
 - Intel 80x86: two status bits and four modes
 - Unix distinguishes between **user** and **superuser (root)**
- E.g., to stop users from writing directly to memory and corrupting the logical file structure, the O/S grants write access to memory locations only if the processor is in supervisor mode.

Controlled Invocation

- Example continued: A user wants to write to memory (requires supervisor mode).
- The system has now to switch between modes, but how should this switch be performed?
- Simply changing the status bit to supervisor mode would give all supervisor privileges to the user without any control on what the user actually does.
- Thus, the system should only perform a predefined set of operations in supervisor mode and then return to user mode before handing control back to the user.
- Let's refer to this process as **controlled invocation**.

Core Security Mechanisms



Why Mechanisms at the Core?

- For security evaluation at a higher level of assurance.
- Security mechanisms in a given layer can be compromised from a layer below.
- To evaluate security, you must check that security mechanisms cannot be bypassed.
- The more complex a system, the more difficult this check becomes. At the core of a system you may find simple structures which are amenable to thorough analysis.

Why Mechanisms at the Core?

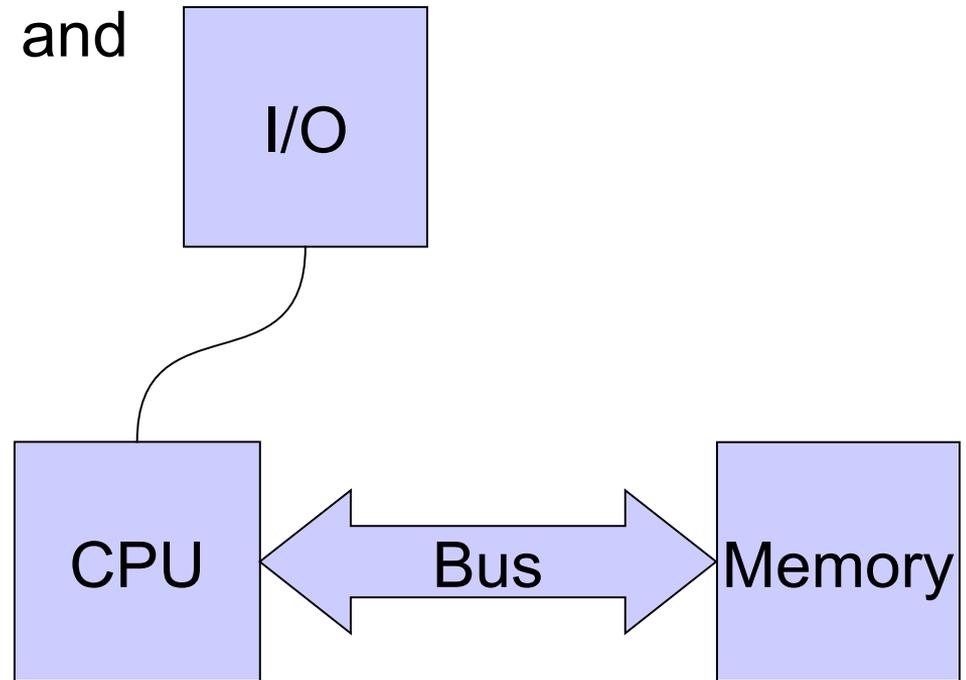
- Putting security mechanisms into the core of the system can reduce performance overheads caused by security.

Processor performance depends on the right choice and efficient implementation of a generic set of operations that is most useful to the majority of users. The same holds for security mechanisms.

- Note: Some sources assume that TCBs and security kernels must enforce multi-level security policies.

Computer Architecture

- Simple schematic description:
 - central processing unit (CPU)
 - memory
 - bus connecting CPU and memory
 - input/output devices



Core CPU Components

- **Registers**: general purpose registers and dedicated registers like:
 - **program counter**: points to the memory location containing the next instruction to be executed.
 - **stack pointer**: points to the top of the **system stack**.
 - **status register**: here CPU keeps essential **state information**.
- **Arithmetic Logic Unit (ALU)**: executes instructions given in a machine language; executing an instruction may also set bits in the status register.

Memory Structures

Security characteristics of different types of memory:

- **RAM (random access memory)**: read/write memory; no guarantee of integrity or confidentiality.
- **ROM (read-only memory)**: provides integrity but not confidentiality; ROM may store (part of) the O/S.
- **EPRM (erasable & programmable read-only memory)**: could store parts of the O/S or cryptographic keys; technologically more sophisticated attacks threaten security.
- **WROM (write-once memory)**: memory contents are frozen once and for all, e.g. by blowing a fuse placed on the write line; WROM could hold cryptographic keys or audit logs.

Memory Structures

- **Volatile memory** loses its contents when power is switched off.
 - Memory contents still present after a short power loss.
 - Can be reconstructed by special electronic techniques if power has been switched off for some time.
 - To counter such attacks, memory has to be **overwritten repeatedly** with suitable bit patterns.
- **Non-volatile (permanent) memory** keeps its content when power is switched off; if attacker can directly access memory bypassing the CPU, **cryptographic or physical measures** are needed to protect sensitive data.
 - E.g., a light sensor in a **tamper resistant** module may detect an attempted manipulation and trigger the deletion of the data kept in the module.

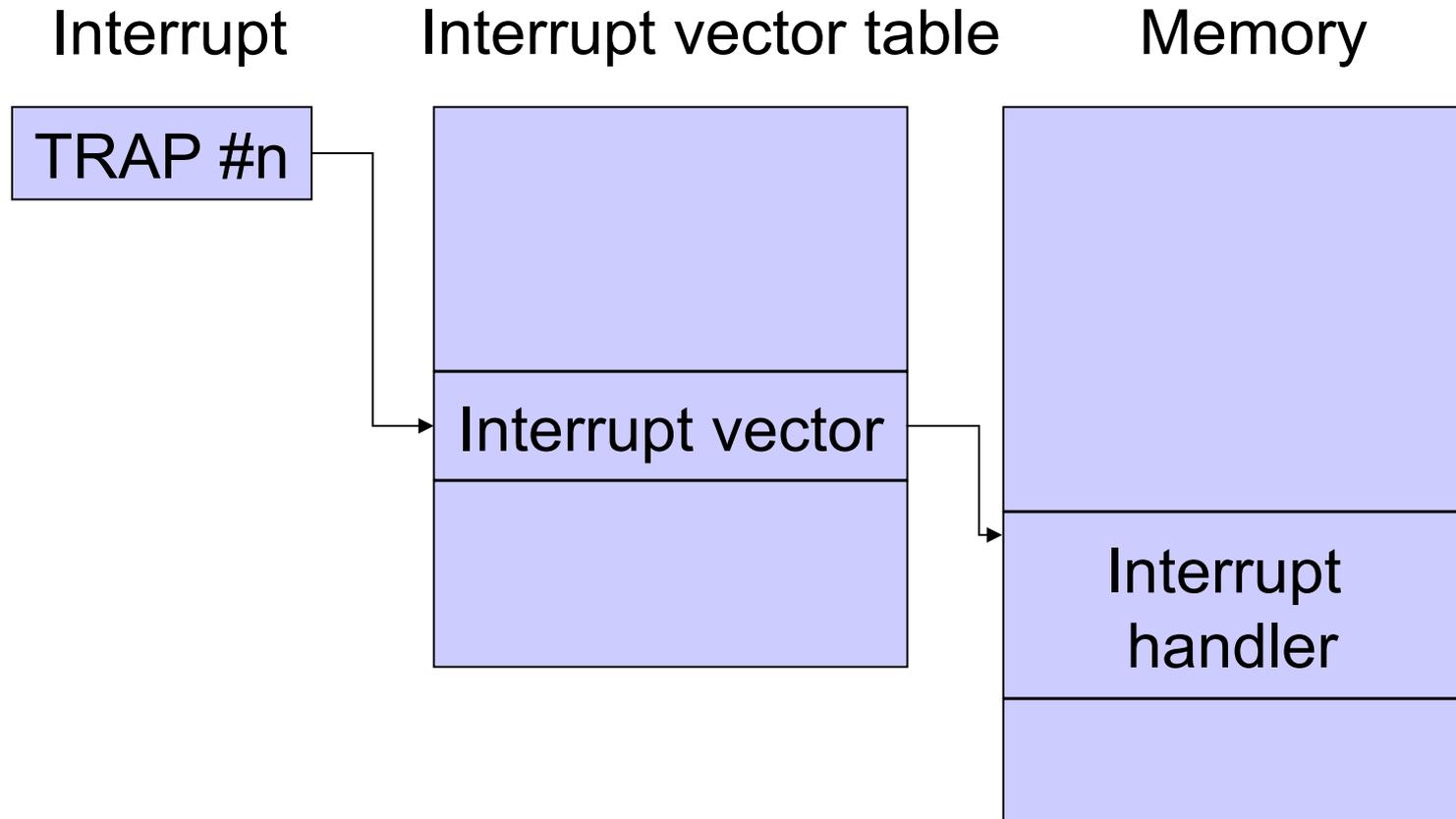
Processes and Threads

- **Process**: a program in execution, consisting of **executable code**, **data**, and the **execution context**, e.g. the contents of certain CPU registers.
 - A process has its own **address space** and communicates with other processes only through O/S primitives.
 - Logical separation of processes as a basis for security.
 - A **context switch** between processes can be an expensive operation.
- **Threads**: strands of execution within a process. Threads share an address space to avoid the overheads of a full context switch, but they also avoid potential security controls.
- **Processes and threads are important units of control for the O/S, and for security. They are the 'subjects' of access control.**

Traps – Interrupts

- CPU deals with interruptions of executions created by errors in the program, user requests, hardware failure, etc., through **exceptions**, **interrupts**, and **traps**.
- These terms refer to different types of events; we use trap as the generic term.
- A **trap** is a special input to the CPU that includes an address (**interrupt vector**) in an **interrupt vector table** giving the location of the program (**interrupt handler**) that deals with the condition specified in the trap.
- When a trap occurs, the CPU saves its current state on the stack and then executes the interrupt handler.
- The interrupt handler has to restore the CPU to a proper state, e.g. by clearing the supervisor status bit, before returning control to the user.

Interrupt Vectors



Interrupting Interrupts

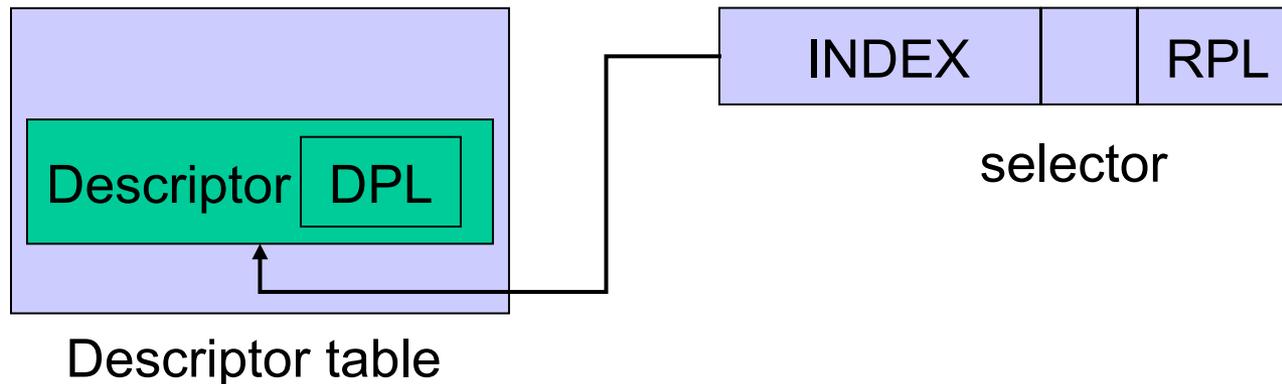
- A further interrupt may arrive while the CPU deals with a current interrupt, so the CPU may have to interrupt the current interrupt handler.
- Improper handling of such a situation can cause security failures.
 - Consider a system where a user can interrupt the execution of a program by typing **CTRL-C** so that the CPU returns to the O/S prompt with the status bit of the current process. A user could then enter supervisor mode by interrupting the execution of an O/S call.
- The interrupt table is a particularly interesting point of attack and has to be protected adequately. Redirecting pointers is an efficient way of compromising the integrity of the O/S.

Example: Intel 80x86

- Support for access control at machine language level based on **protection rings**.
- Two-bit field in the **status register**: four **privilege levels**; Unix, Windows 2000 use levels 0 (O/S) and 3 (user).
- Privilege levels can only be changed through **POPF**.
- Processes can only access objects in their ring or in outer rings; processes can invoke subroutines only within their ring; processes need **gates** to execute procedures in an inner ring.
- Information about system objects like memory segments, access control tables, or gates is stored in **descriptors**. The privilege level of an object is stored in the **DPL field** of its descriptor.

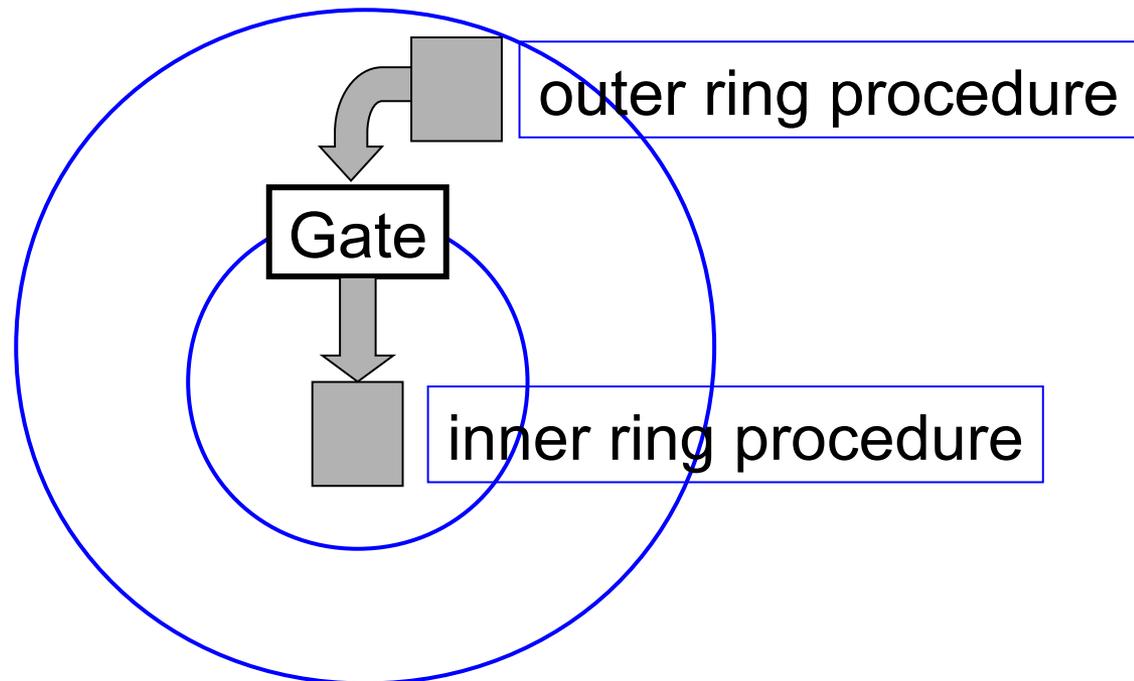
Intel 80x86 - Access Control

- Descriptors held in descriptor table; accessed via selectors.
- **Selector**: 16-bit field, contains index for the object's entry in the descriptor table and a **requested privilege level (RPL)** field; only O/S has access to selectors.
- **Current privilege level (CPL)**: code segment register stores selector of current process; access control decisions can be made by comparing CPL (subject) and DPL (object).



Intel 80x86: Controlled Invocation

- **Gate:** system object pointing to a procedure, where the gate has a privilege level different from that of the procedure it points to.
- Allow execute-only access to procedures in an inner ring.



Intel 80x86: Controlled Invocation

- A subroutine call saves state information about the calling process and the return address on the stack.
 - Should this stack be in the inner ring? Violates the security policy forbidding write to an inner ring.
 - Should this stack be in the outer ring? The return address could be manipulated from the outer ring.
- Therefore, part of the stack (how much is described in the gate's descriptor) is copied to a more privileged stack segment.

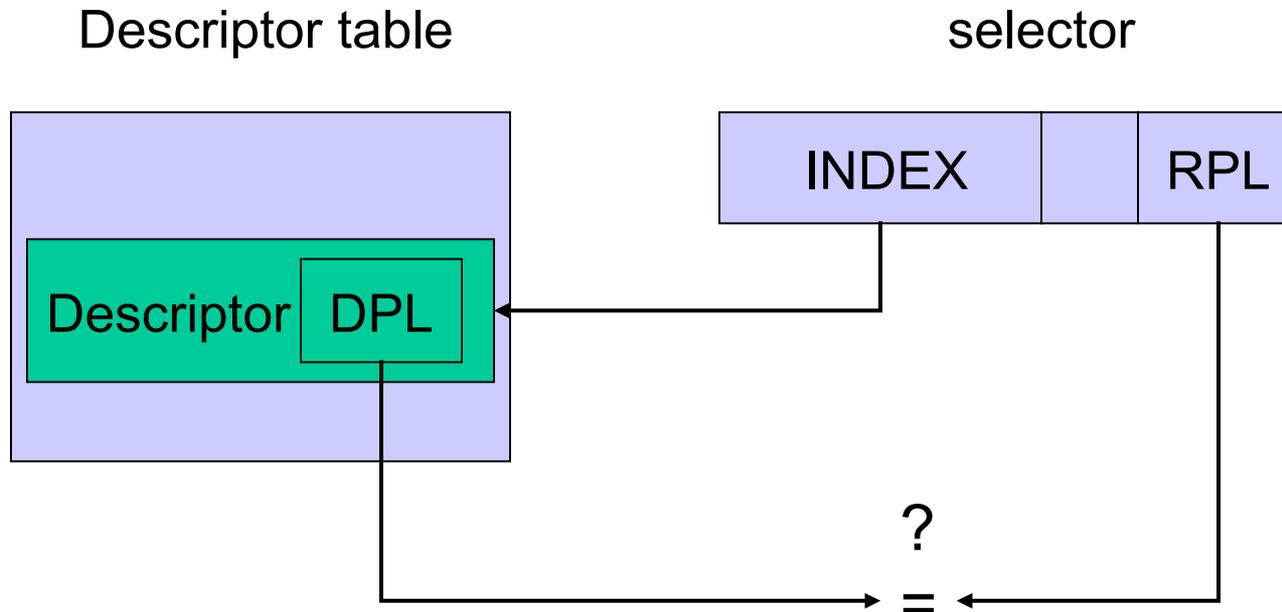
A Loophole?

- When invoking a subroutine through a gate, the CPL changes to the level of the code the gate is pointing to; on returning from the subroutine, the CPL is restored to that of the calling process.
- The outer-ring process may ask the inner-ring procedure to copy an inner ring object to the outer ring; this will not be prevented by any of the mechanisms presented so far, nor does it violate the stated security policy.
- Known as **luring attack**, or as **confused deputy problem**.

Remedy

- To take into account the level of the calling process, use the **adjust privilege level (ARPL) instruction**.
- This instruction changes the **RPL** fields of all selectors to the **CPL** of the calling process. The system then compares the **RPL** (in the selector) and the **DPL** (in the descriptor) of an object when making access control decisions.

Comparing RPL and DPL



Security Mechanisms in O/S

- O/S manages access to data and resources; multitasking O/S interleaves execution of processes belonging to different users. It has to
 - separate user space from O/S space,
 - logically separate users,
 - restrict the memory objects a process can access.
- Logical separation of users at two levels:
 - file management, deals with logical memory objects
 - memory management, deals with physical memory objects
- For security, this distinction is important.

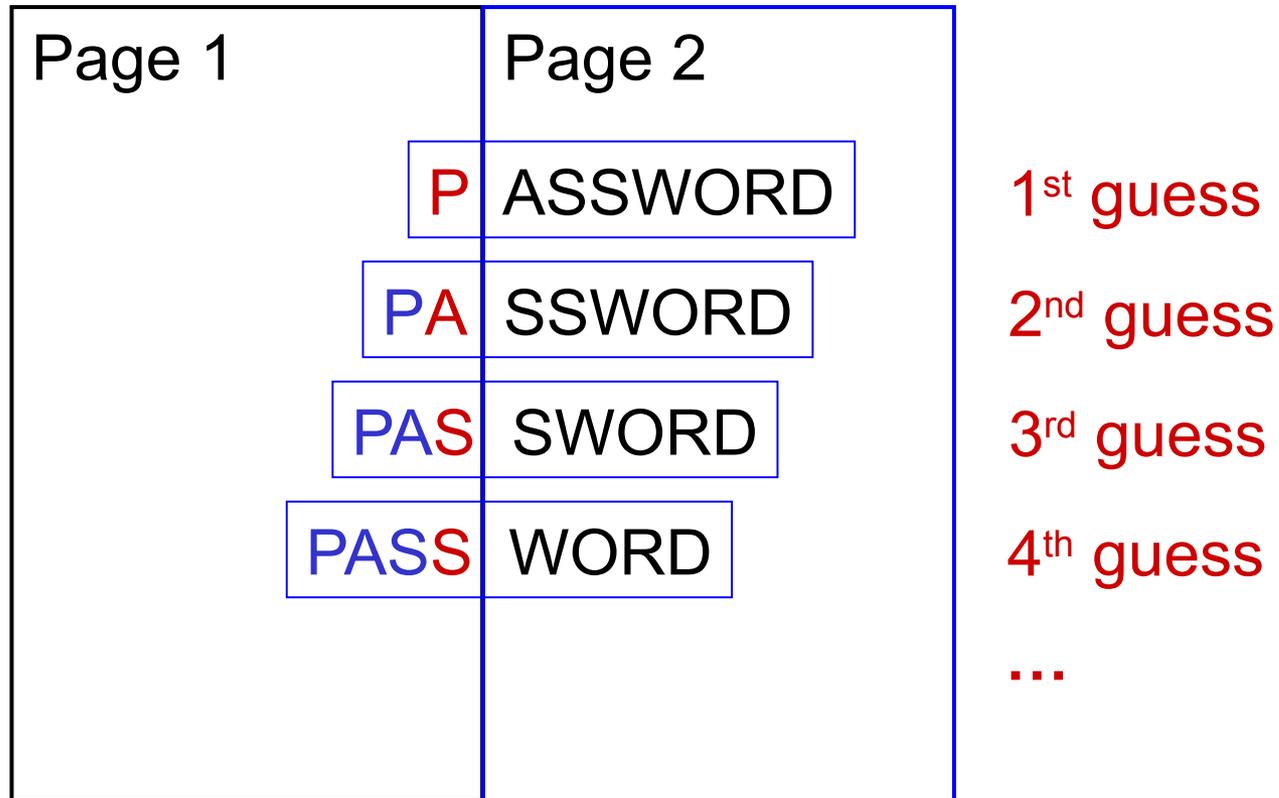
Segments and Pages

- **Segmentation** divides memory into logical units of variable lengths.
 - + A division into logical units is a good basis for enforcing a security policy.
 - Units of variable length make memory management more difficult.
- **Paging** divides memory into pages of equal length.
 - + Fixed length units allow efficient memory management.
 - Paging is not a good basis for access control as pages are not logical units. One page may contain objects requiring different protection. **Page faults** can create a covert channel.

A Covert Channel

- When a process accesses a logical object stored on more than one page, a **page fault** occurs whenever a new page is requested.
- A **covert channel** exists if page faults are observable.
- Consider a password scheme where the password entered is compared character by character with the reference password stored in memory; access is denied the moment an incorrect match is found.
- If a password is stored across a page boundary, then observing a page fault indicates that the piece of the password on the first page has been guessed correctly; if the attacker can control where the password is stored on the page, password guessing becomes easy.

Exploiting the Covert Channel

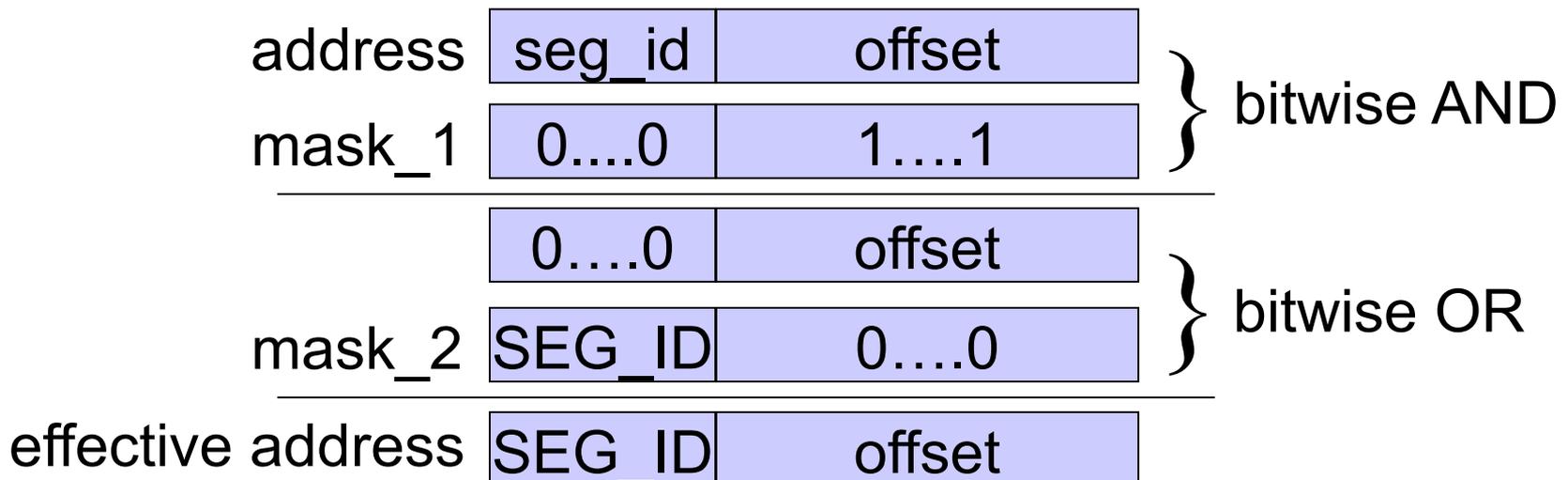


Memory Protection

- O/S controls access to data objects in memory.
- A data object is represented by a collection of bits stored in certain memory locations.
- Access to a logical object is ultimately translated into access operations at machine language level.
- Three options for controlling access to memory:
 - operating system modifies the addresses it receives from user processes;
 - operating system constructs the effective addresses from relative addresses it receives from user processes;
 - operating system checks whether the addresses it receives from user processes are within given bounds.

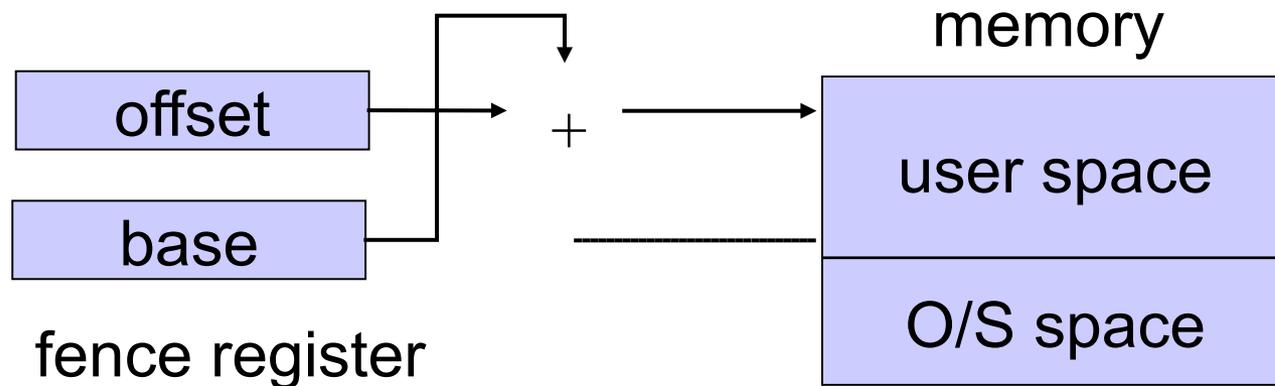
Address Sandboxing (Modification)

- Address consists of **segment identifier** and **offset**. When the operating system receives an address, it sets the correct segment identifier as follows:
- Bitwise AND of the address with **mask_1** clears the segment identifier; bitwise OR with **mask_2** sets the segment identifier to the intended value **SEG_ID**.



Relative Addressing

- Clever use of addressing modes can keep processes out of forbidden memory areas.
- **Fence registers**: **base register addressing** keeps users out of O/S space; fence register points to top of user space.
- **Bounds register** define the bottom of the user space. Base and bounds registers allow to separate **program** from **data** space.



Function codes

- Motorola 68000 **function codes** indicate processor status so that address decoder may select between user and supervisor memory or between data and programs.

FC2	FC1	FC0	
0	0	0	(undefined,reserved)
0	0	1	user data
0	1	0	user program
0	1	1	(undefined,reserved)
1	0	0	(undefined,reserved)
1	0	1	supervisor data
1	1	0	supervisor program
1	1	1	interrupt acknowledge

General Lessons

- The ability to distinguish between data and programs is a useful security feature, providing a basis for protecting programs from modification.
- From a more abstract point of view, memory has been divided into different regions. Access control can then refer to the **location** a data object or program comes from.
- This can serve as a first example for **location based access control**. Distributed systems or computer networks may use location based access control at the level of network nodes.

Tagged architectures

- Tagged architectures indicate type of each memory object.

tag	data
INT
OP
STR
...
...
...

Summary

- Security policies can be enforced in any layer of a computer system.
- Mechanisms at lower layers are more generic and are universally applied to all “applications” above, but might not quite match the requirements of the application.
- Mechanisms at upper layers are more application specific, but applications have to be secured individually.
- This fundamental dilemma is a recurring theme in information security.