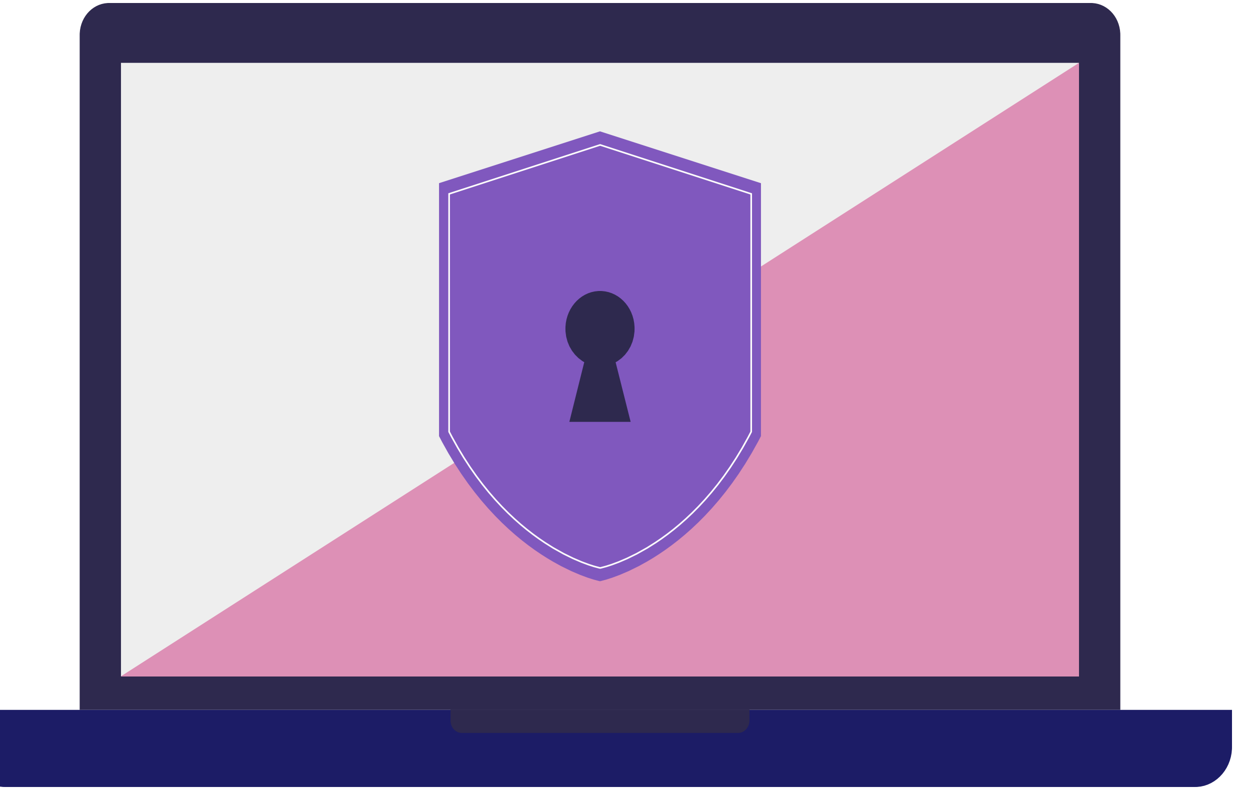
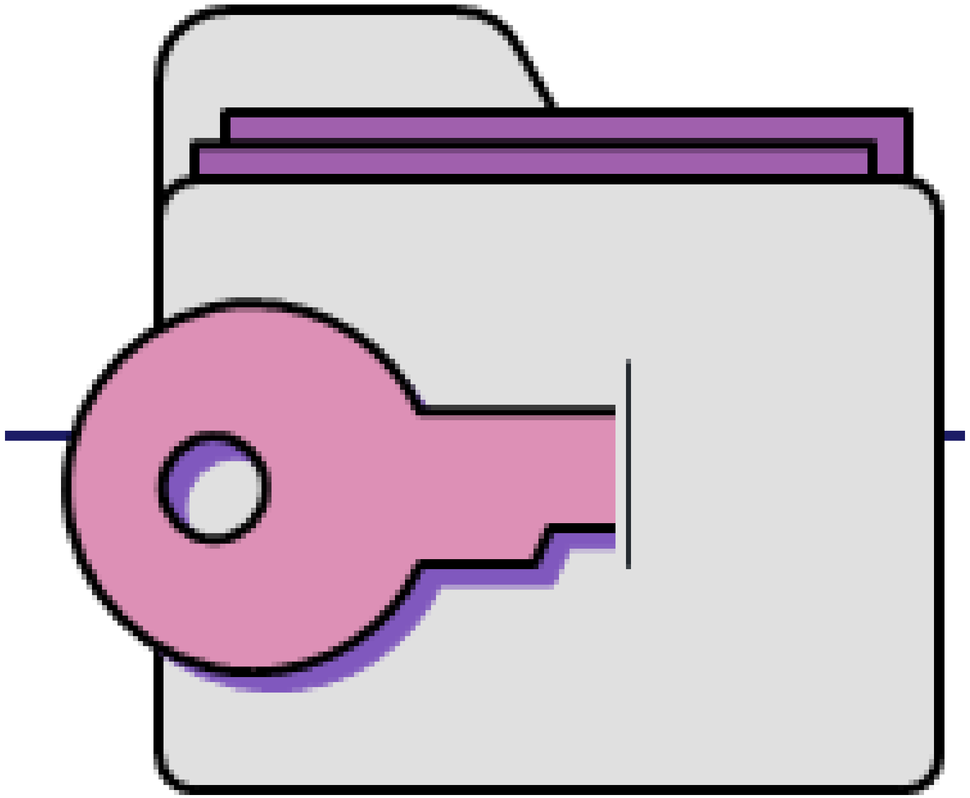


Computer Security

CS433



Chapter 5

Operating Systems

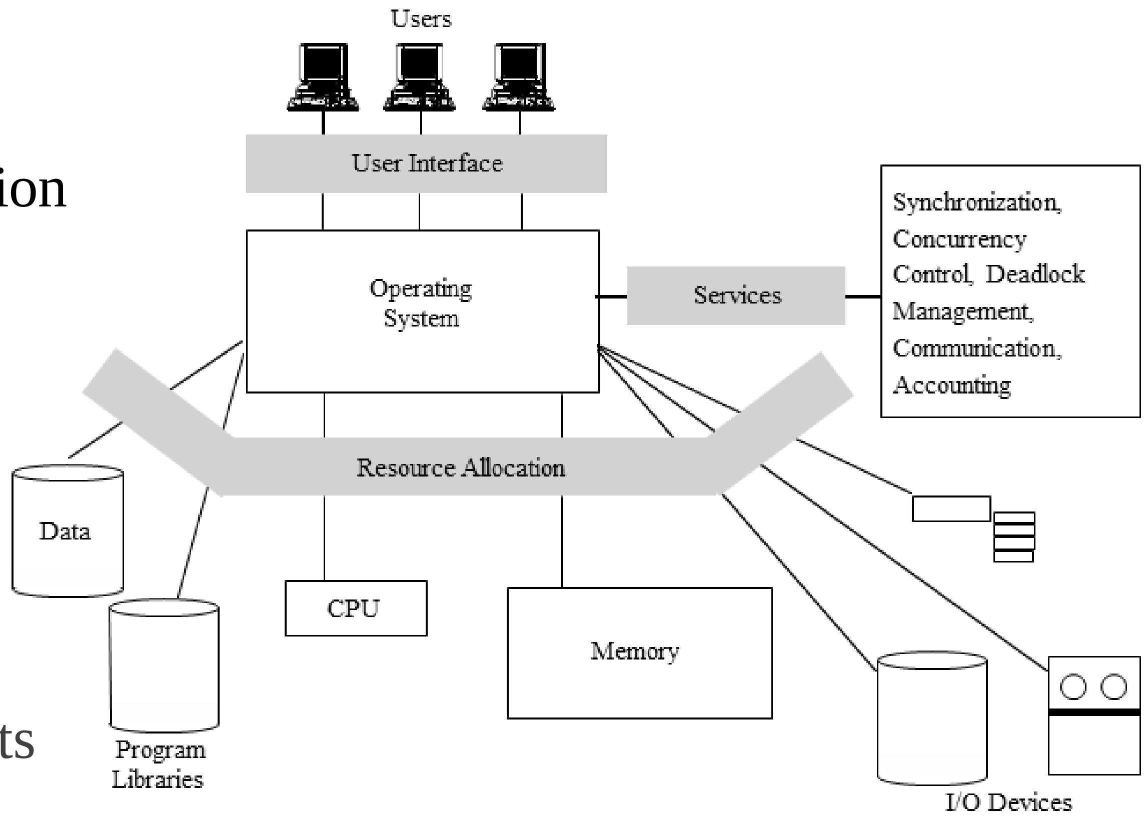
Objectives

Learn	Basic security functions provided by operating systems
What	System resources that require operating system protection
Learn	Operating system design principles
Learn	How operating systems control access to resources
Introduce	The history of trusted computing
Study	Characteristics of operating system rootkits

Operating System Functions

Security-relevant features:

- ✓ Enforce sharing
- ✓ Interprocess communication and synchronization
- ✓ Protection of critical data
- ✓ Guaranteed fair service
- ✓ Interface to hardware
- ✓ User authentication
- ✓ Memory protection
- ✓ File and I/O device access control
- ✓ Allocation and access control to general objects



Operating Systems

- ✓ The operating system is the fundamental controller of all system resources
  - It is a primary target of attack

History of Operating System

- ✓ Single-user systems, no OS

Users responsible for loading program, libraries of and “cleaning up” after use

- ✓ Multiprogrammed OS, aka monitors
  - Multiple users
  - Multiple programs
  - Scheduling, sharing, concurrent use
- ✓ Personal computers

Protected Objects

The rise of multiprogramming meant that several aspects of a computing system require protection

1. Memory
2. Sharable I/O devices such as disks
3. Serially reusable I/O devices, such as printers
4. Sharable programs and sub-procedures
5. Networks
6. Sharable data

OS Layered Design

OS implements several levels of functionality and protection

The functions are grouped in three categories:

- ✓ Security Kernel

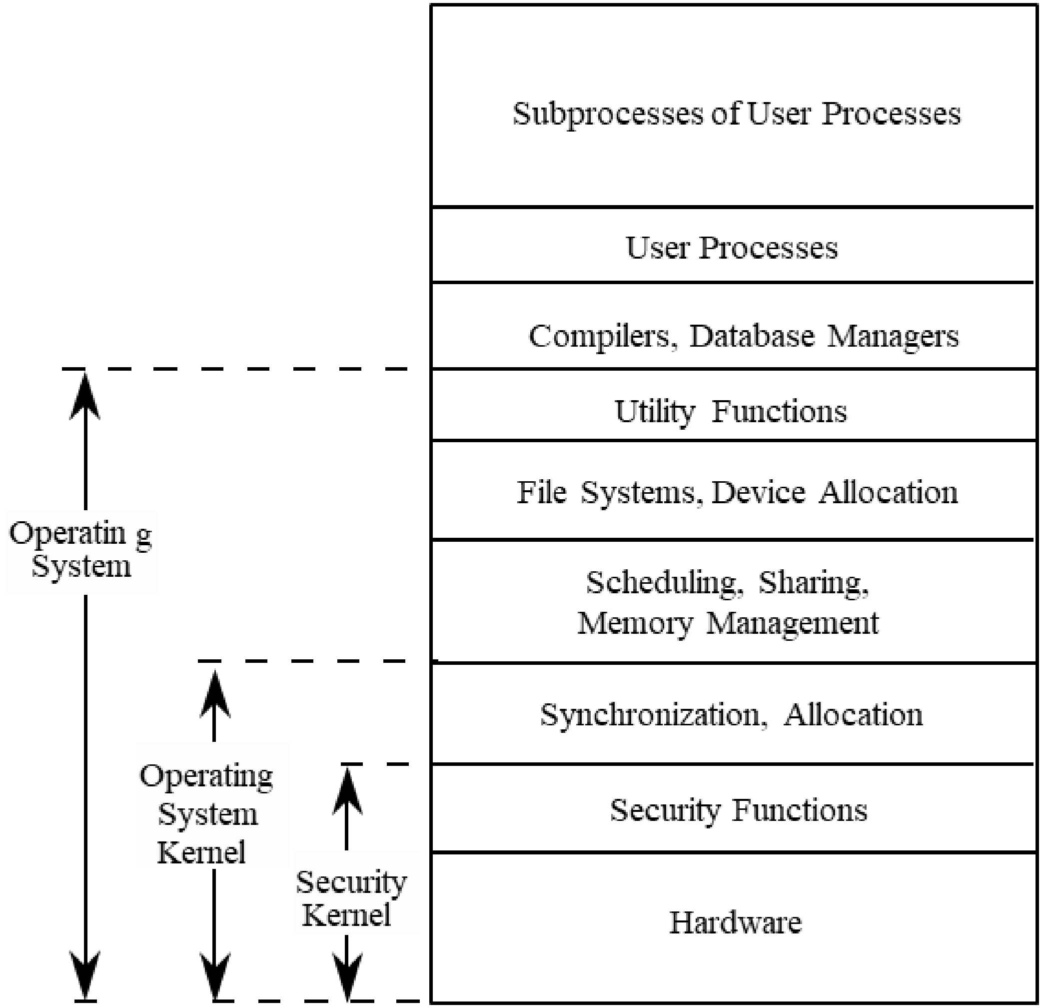
  Enforces security

- ✓ Operating System Kernel

  Allocates resources such as time or access to hw devices

- ✓ Other Operating System Functions

  Implement user-hw interface

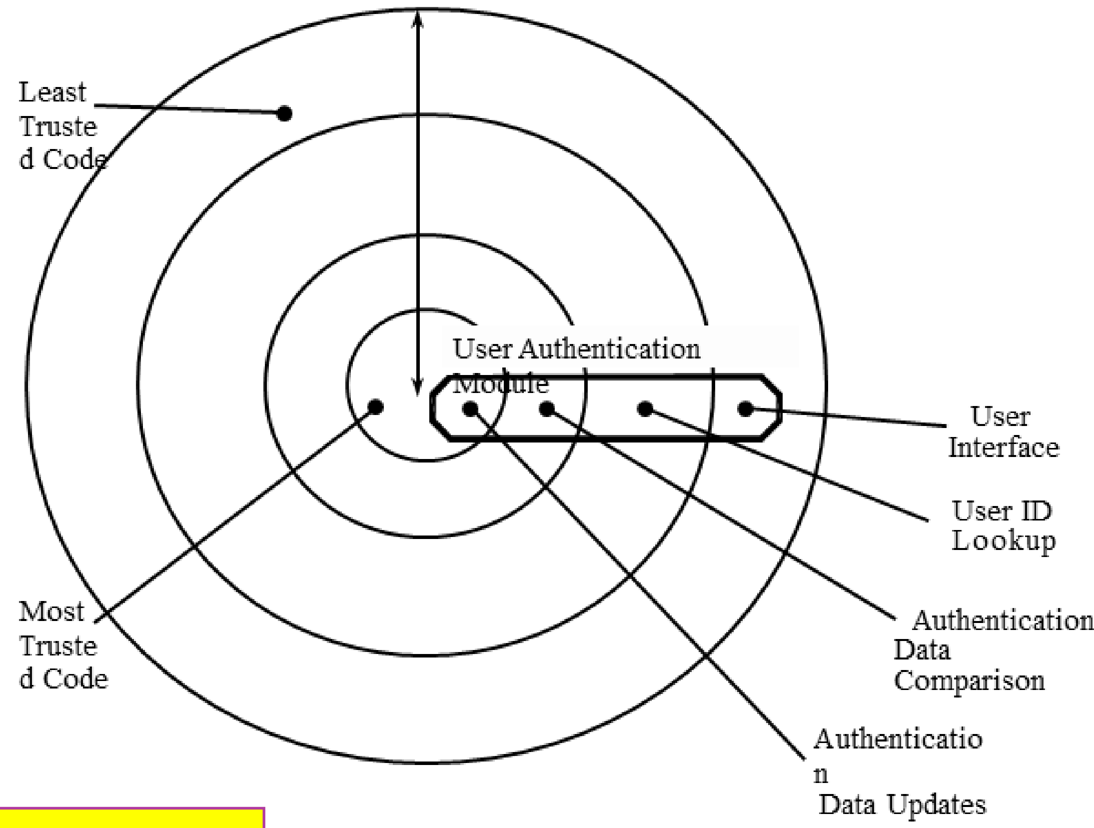


Functions Spanning Layers

Example. Password Authentication

User authentication functions implemented in several layers

- ✓ UI: displaying/receiving password and echoing every character as *
- ✓ User ID lookup and comparison
- ✓ Authentication update: checking user's identity has been authenticated, or request to change user's password in the system table.

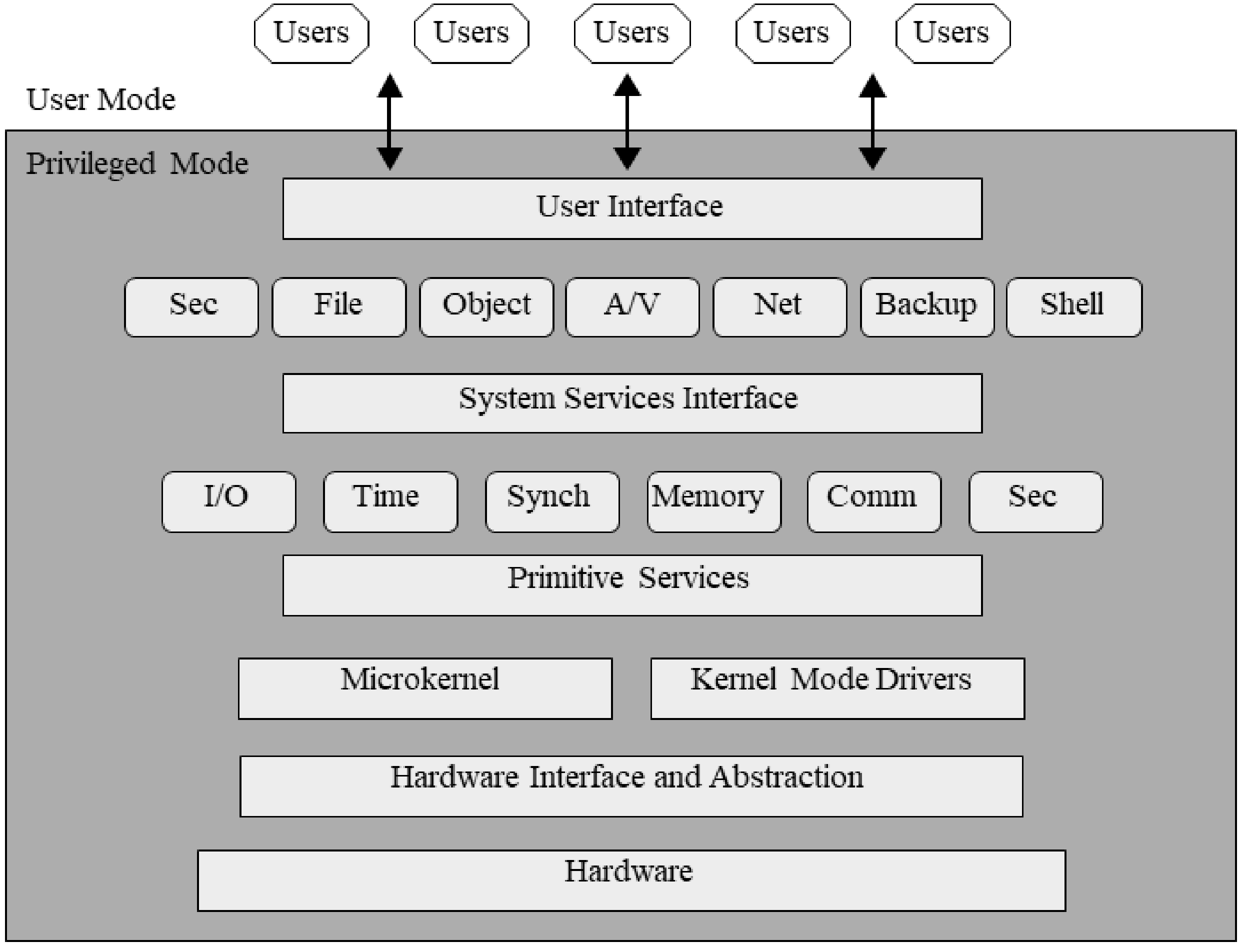


Changing the system password table is certainly more critical than displaying a box for password entry

Modular OS Design

- ✓ A modern OS has different **modules**
- ✓ Modules come from several sources
  - Not all trustworthy
  - Must all integrate successfully

The OS must protect itself in order to protect its users and resources.



OS Security Tools

Virtualization

- ✓ Presenting a user the appearance of a system with only the resources the user is entitled to use.
- ✓ The user has access to a virtual machine (VM)
  - The user cannot access resources available to the OS but exist outside the VM
- ✓ A hypervisor, or VM monitor - software that implements a VM
  - Translates access requests between the VM and the OS
  - Can support multiple OSs in VMs simultaneously
  - ✓ Example. Honeypot.
  - A VM meant to lure an attacker into an environment that can be both controlled and monitored

Sandbox

An environment from which a process can have only limited, controlled impact on outside resources.

OS Security Tools

Separation and Sharing

Methods of separation

Separation occurs by space, time, access control, or cryptography

- Physical
- Temporal
- Logical
- Cryptographic

OS Security Tools

Separation and Sharing

Methods of supporting separation/sharing

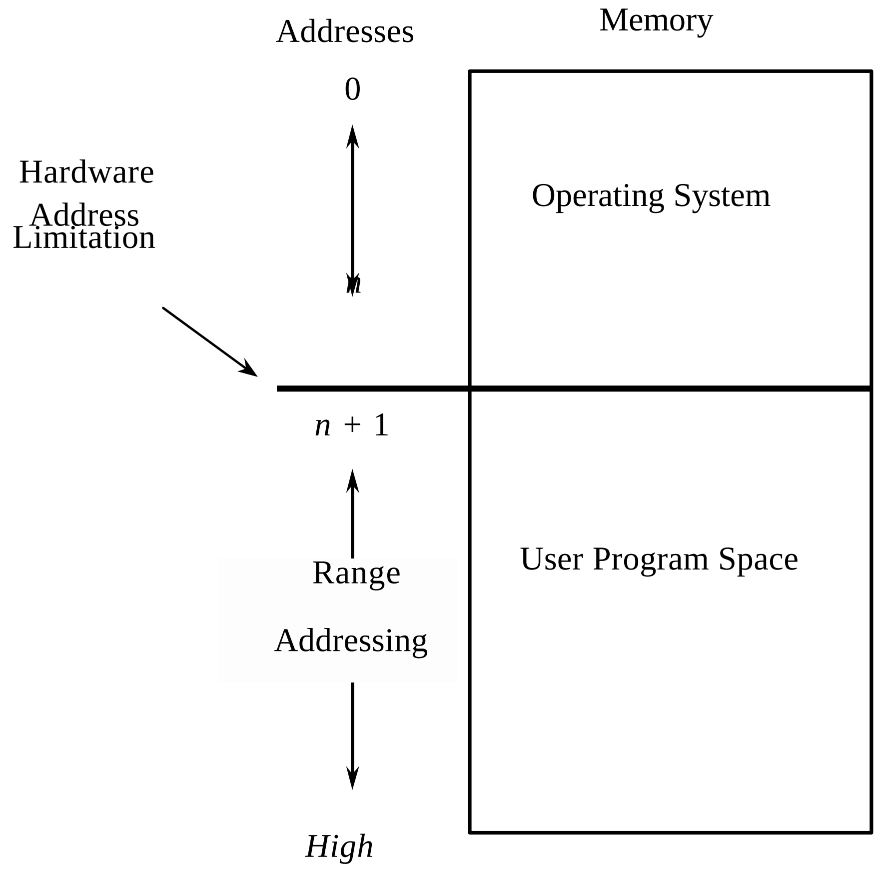
1. Do not protect: designed for one user/process; sensitive procedures are being run at separate times
2. Isolate: different processes run concurrently but unaware of each other
3. Share all or share nothing: an object is either public or private
4. Share but limit access: access control is implemented for a specific user and a specific object
5. Limit use of an object: the use of object after access is limited; e.g. read but not print

Hardware Protection of Memory

Memory protection implements both separation and sharing

1 Fence

- ✓ Fence is a method to confine users to one side of a boundary
- ✓ **Implementation**
  1. Fixed fence-
  2. Fence register



1. Fixed fence

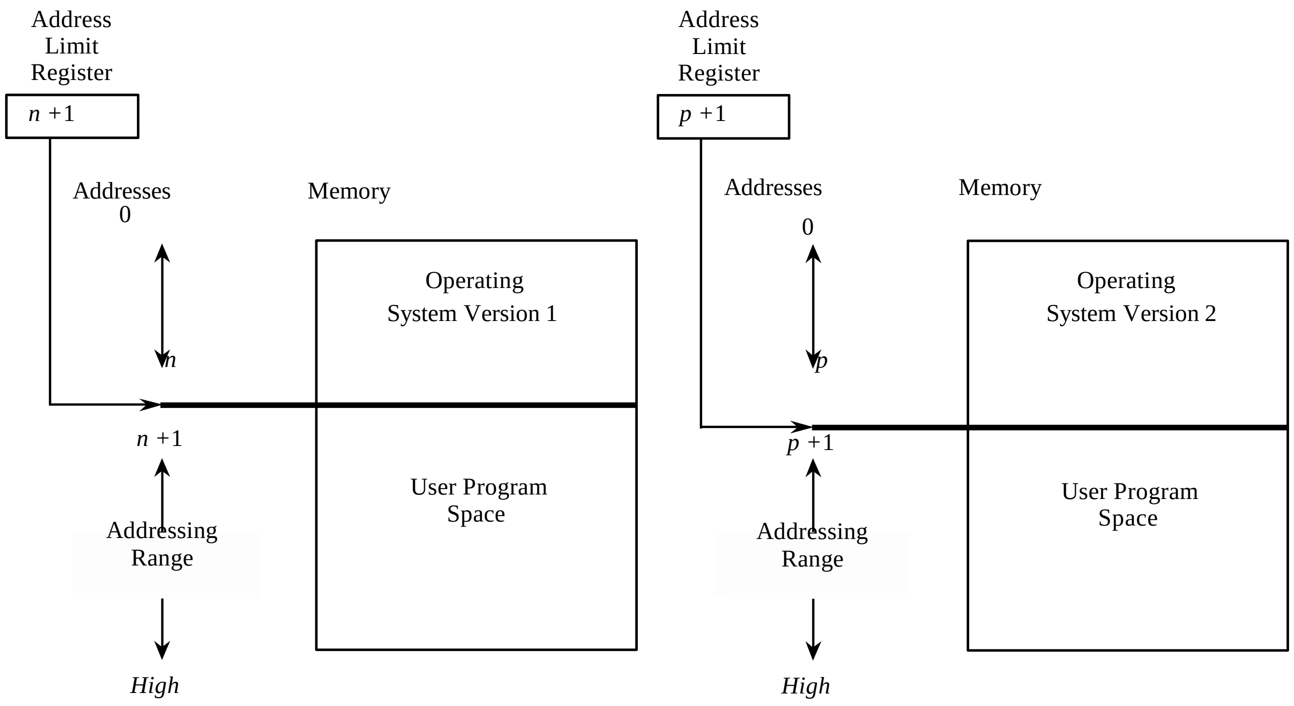
- ✓ A predefined memory address
- ✓ Cons. Very restrictive

OS Security Tools

1 Fence

2. Fence register

- ✓ Hardware register contains the end address of the OS
- ✓ Pros. Location of the fence could be changed
- ✓ Cons. A fence register protects in only one direction.
  - OS is protected from a single user, users are not protected from one another
  - A user cannot identify certain areas of the program as inviolable

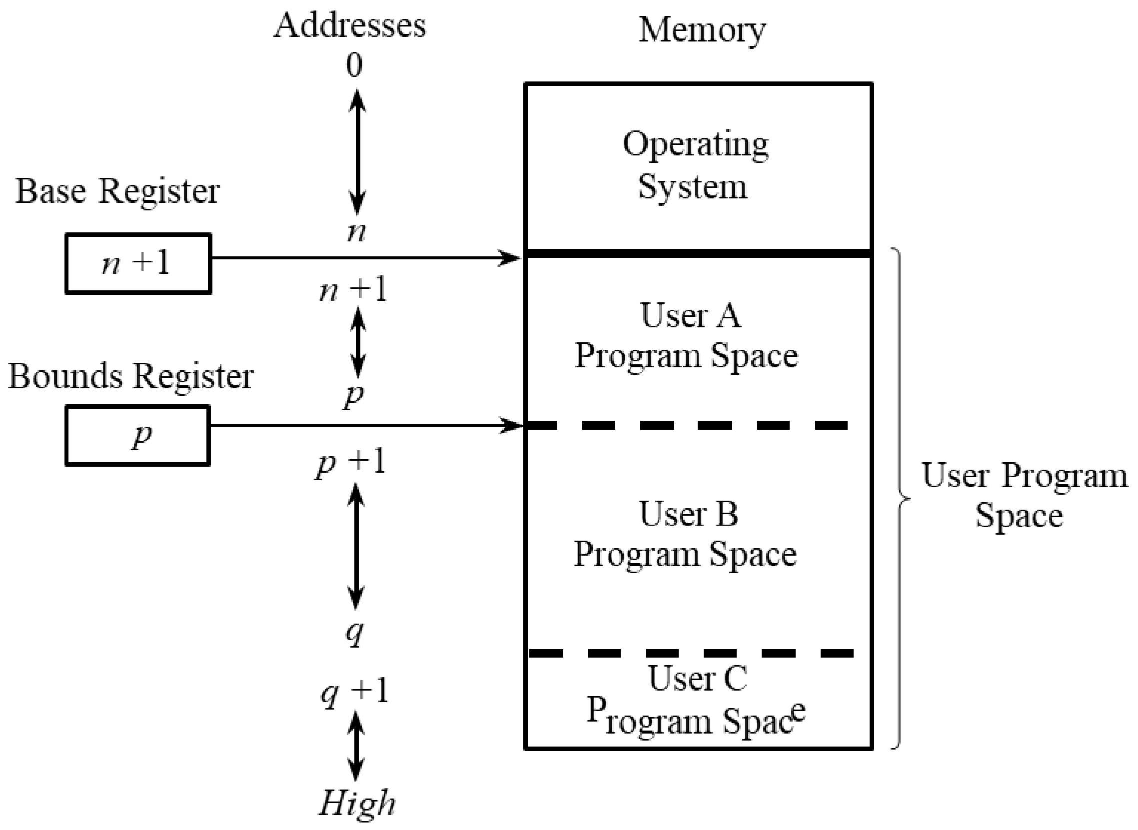


OS Security Tools

2 Base/Bounds Registers

Base/bounds registers surround a program, data area, or domain

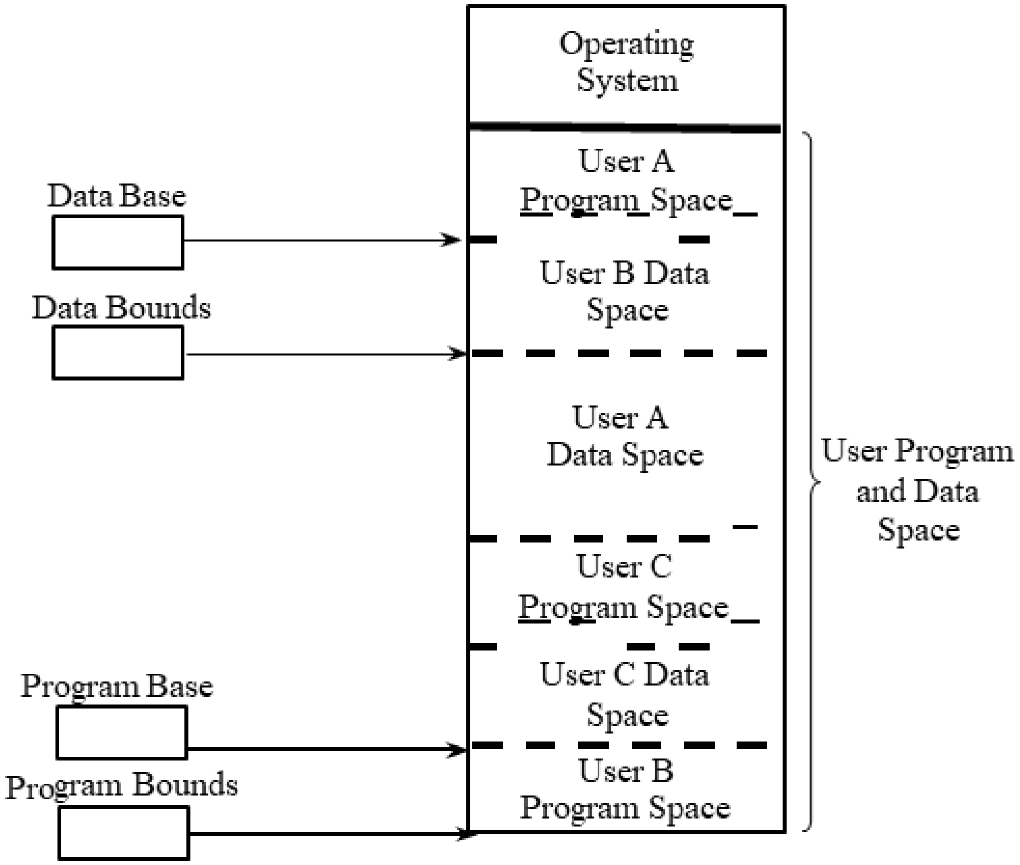
- ✓ Base register
  - AKA **variable fence register**
  - A relocation register is used to provide a base or starting address.
- ✓ Bound register
  - An upper address limit
  - Protects users from other users, ***but not within users designated space***
  - Solution. Two Pairs of Base/Bounds Registers



OS Security Tools

3 Two Pairs of Base/Bounds Registers

- ✓ A user can accidentally store data on top of instructions
- ✓ **Solutions.** Use another pair of base/bounds registers
  - One for the instructions (code)
  - One for the data space
- ✓ Do not prevent all program errors, just limiting the effect of overwriting important parts of data/code
- ✓ The ability to relocate user space separately
- ✓ A problem with previous approaches is the contiguous allocation of memory!
  - Protection is all or none
    - Certain data values written upon initializing and must not change



OS Security Tools

4 Tagged Architecture

- ✓ Every word of memory has extra bits to indicate the access rights
- ✓ The bits are tested every time an instruction accesses that word

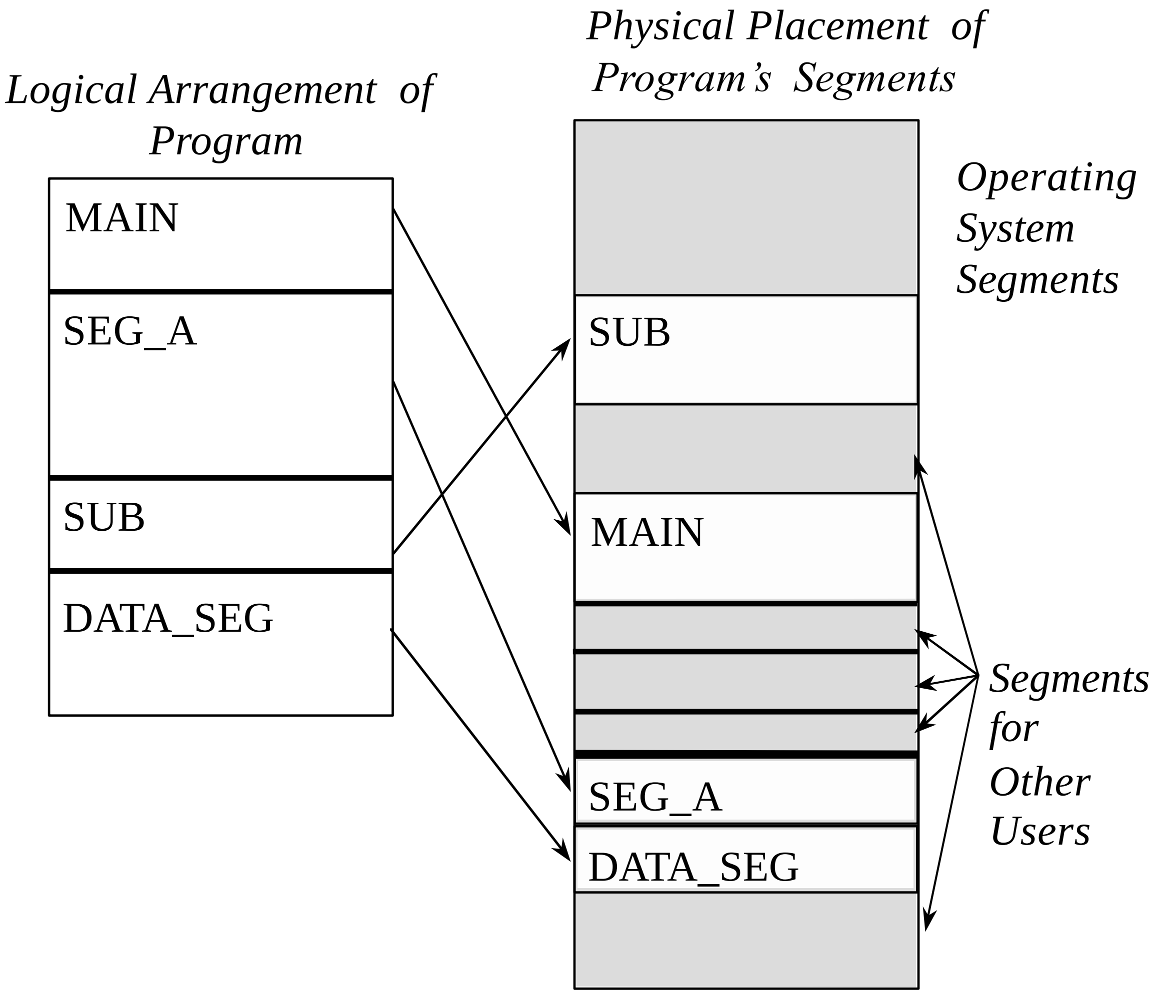
Tag	Memory Word
R	0001
RW	0137
R	0099
X	
X	
X	
X	
X	
X	
R	4091
RW	0002

Code: R = Read-only   RW = Read/Write
X = Execute-only

OS Security Tools

5 Segmentation

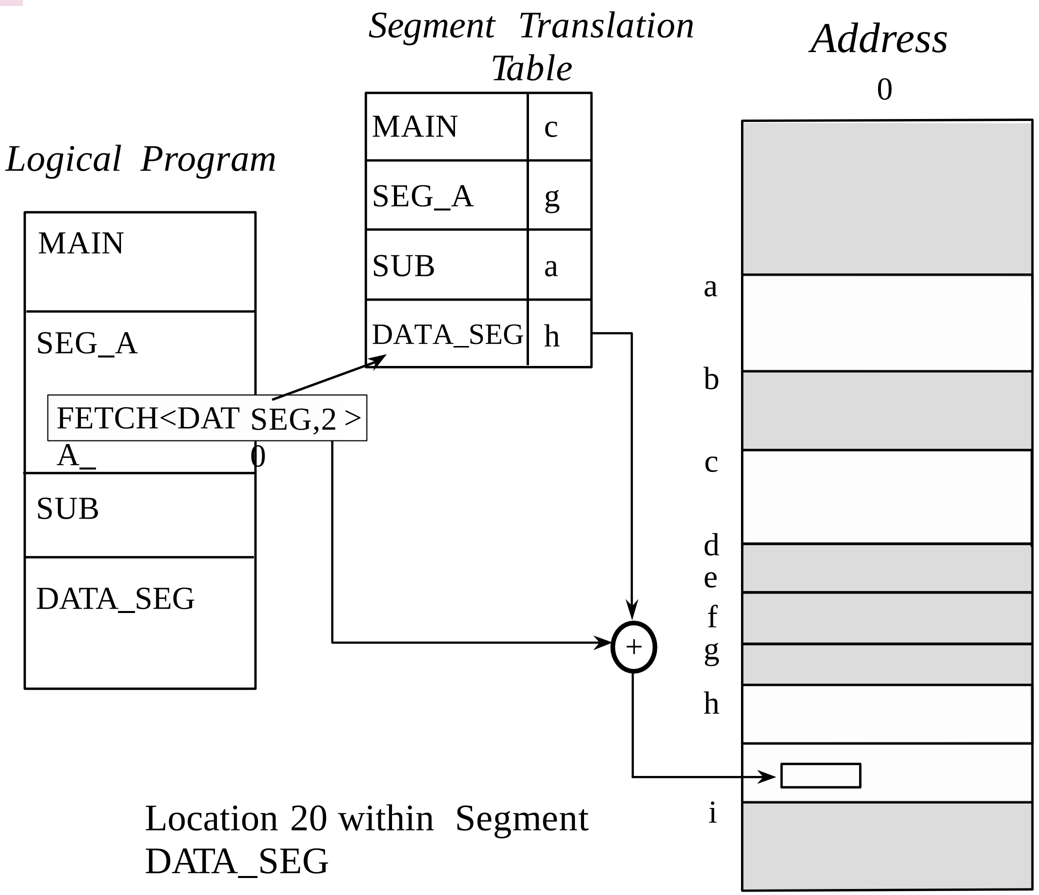
- ✓ Dividing process memory into separate variable-length parts having different access right



OS Security Tools

5 Segmentation

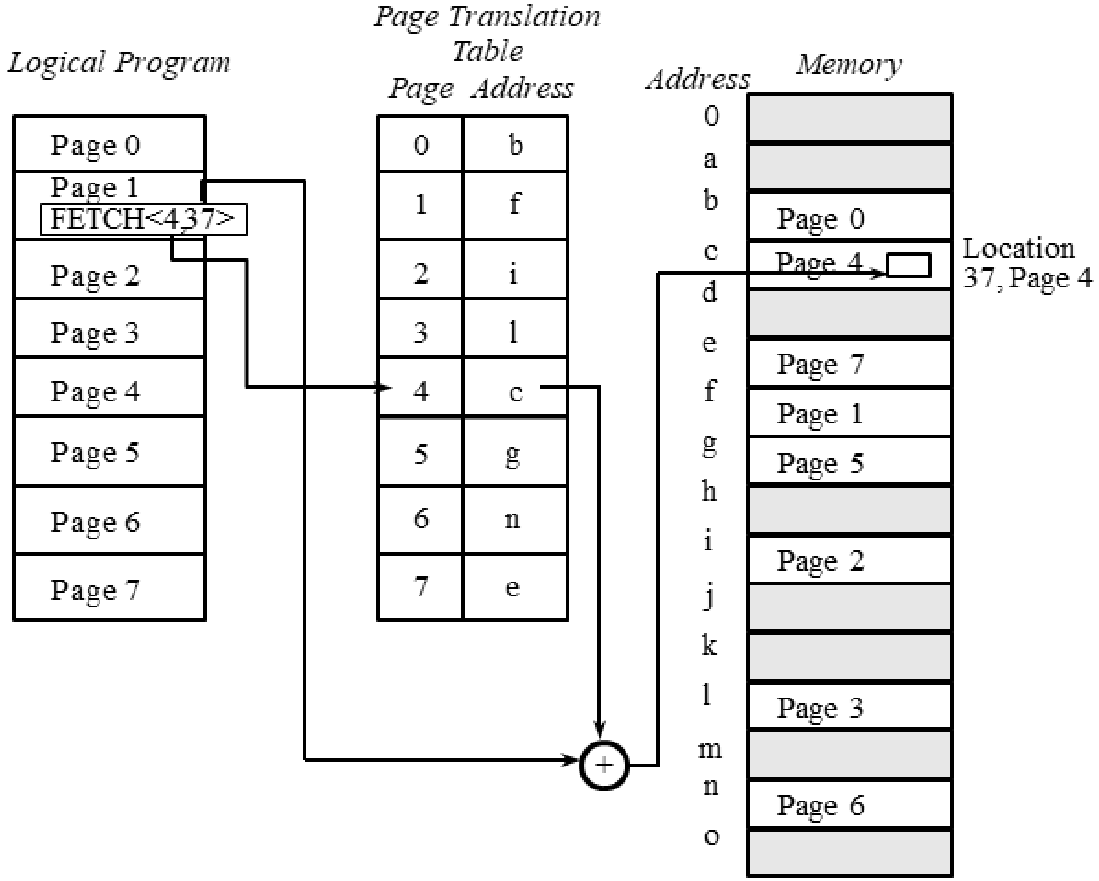
- ✓ The OS maintains a table of segment names and their physical addresses in memory called **segment address table**
  - SAT for each process in execution
  - Shared segments have the same segment name and address in the processes' SATs
- ✓ Access to a location is requested in the form <name, offset>
- ✓ As contiguous memory allocation, segmentation suffers from external fragmentation



OS Security Tools

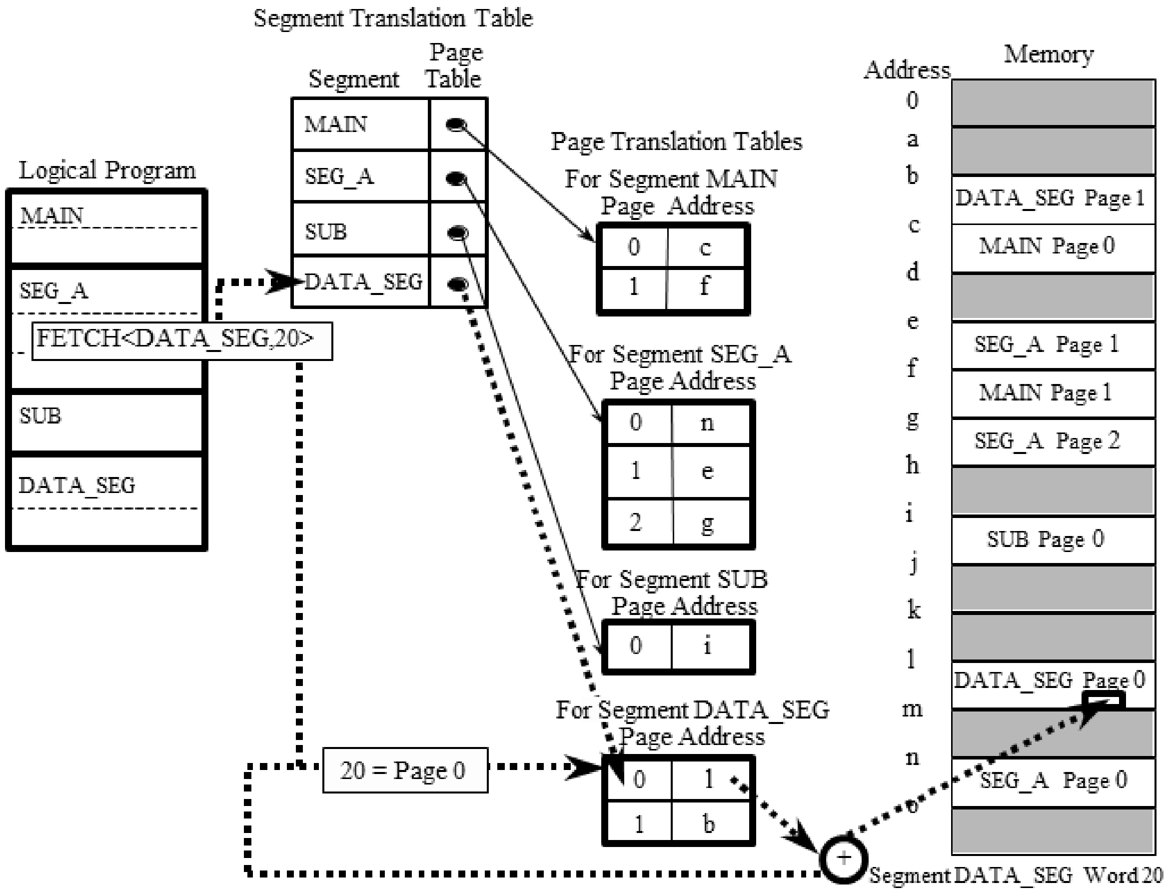
6 Paging

- ✓ Process logical memory is divided into equal-sized pieces called **pages.**
- ✓ Memory is divided into equal-sized units called **page frames.**
- ✓ Each address in a paging scheme is a two- part object <page no., offset>
- ✓ The OS maintains a table of pages numbers and their offsets.
- ✓ External fragmentation is not a problem but the scheme suffers from internal fragmentation.



7 Paged Segmentation

- ✓ Applying paging on top of segmentation
- ✓ The program is divided into logical segments
- ✓ Each segment is broken into fixed-size pages



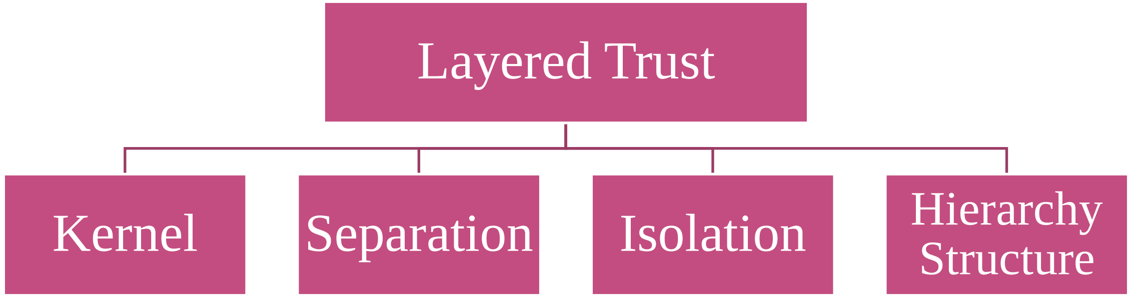
Principles of Secure OS Design

Simplicity of design

- ✓ OSs are inherently complex, and any unnecessary complexity only makes them harder to understand and secure.

Layered design

- ✓ Keeps a design logical and understandable
- ✓ A way to limit risk
- ✓ Enables layered trust
  - Encapsulation
  - Damage control



Example: very tight access controls on critical OS functions, fewer access controls on important noncritical functions, and few if any access controls on functions that aren’t important to the OS.

Principles of Secure OS Design

Kernelized Design

- ✓ A kernel is the part of the OS that performs the lowest-level functions
  - Synchronization
  - Interprocess communication
  - Message passing
  - Interrupt handling

A security kernel

- ✓ Is responsible for enforcing the security mechanisms of the entire OS
- ✓ Typically contained within the kernel
- ✓ The most important part of a security kernel is the reference monitor (RM)
  - Controls accesses to objects
  - Consists of a collection of access controls for devices, files, memory, Interprocess communication, …
  - RM is tamperproof, unbypassable, and analyzable.

Principles of Secure OS Design

Trusted Systems

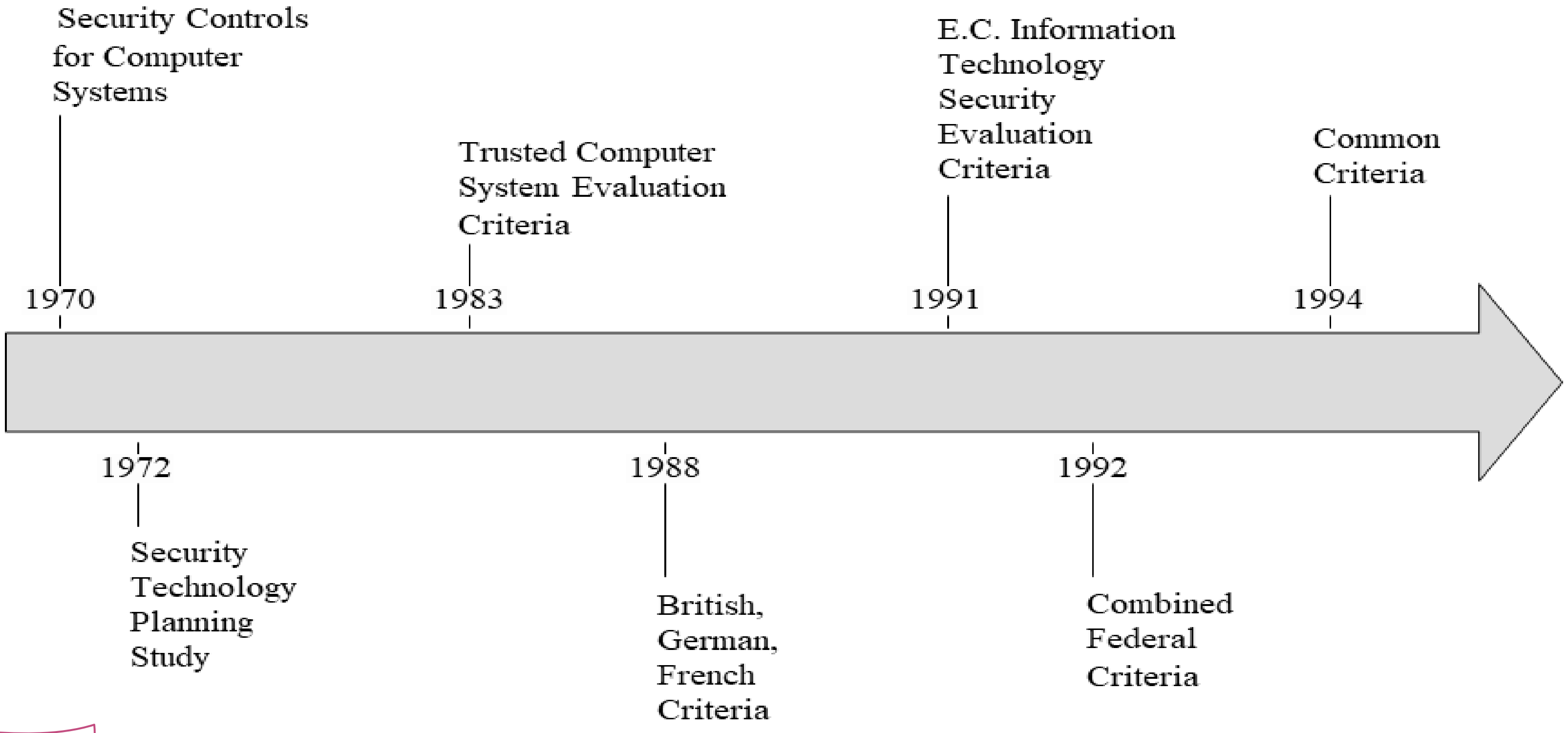
- ✓ A trusted system is one that has been shown to warrant some degree of trust that it will perform certain activities faithfully

Characteristics of a trusted system:

- ✓ **A defined policy** that details what security qualities it enforces
- ✓ Appropriate **measures** and **mechanisms** by which it can enforce security adequately
- ✓ Independent **scrutiny** or **evaluation** to ensure that the mechanisms have been selected and implemented properly

History of Trusted Systems

Started by the U.S. DoD writing the *Trusted Computer System Evaluation Criteria* (TCSEC or Orange Book)



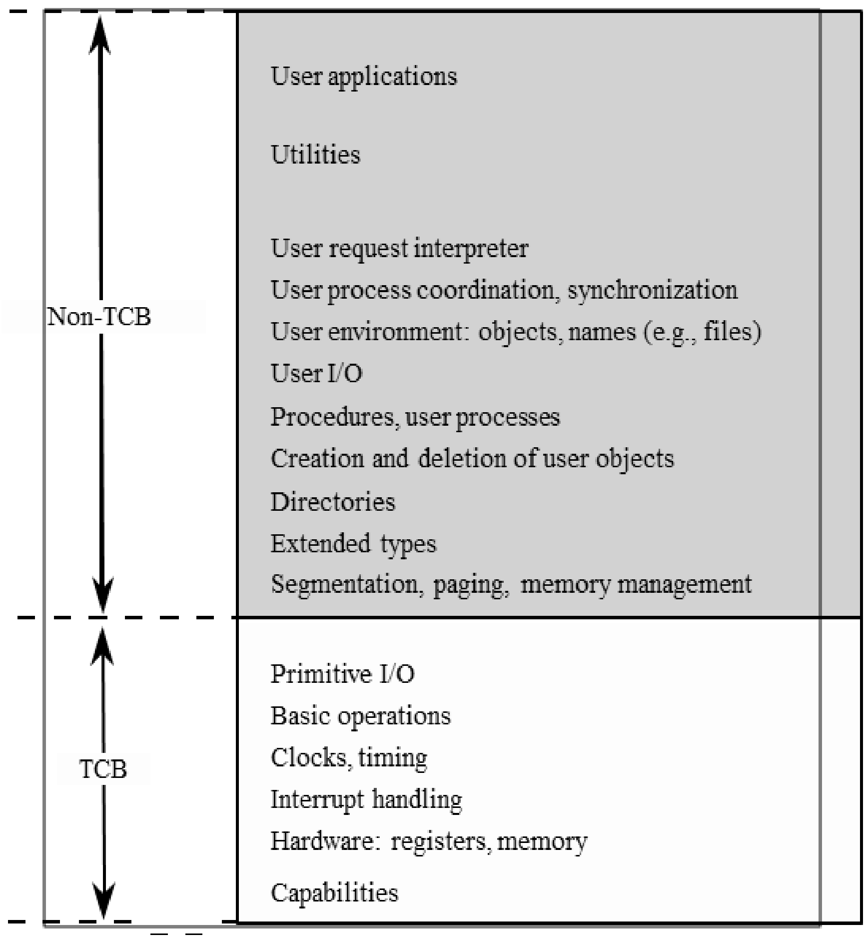
DoD: Department of Defence

Trusted Systems

Trusted computing base (TCB)

- ✓ Everything necessary for a system to enforce its security policy
- ✓ Consists of
  - **hardware**, including processors, memory, registers, a clock, and I/O devices
  - **processes**, so security-critical processes are protected
  - **primitive files**, such as the security access control database and identification and authentication data
  - **protected memory,** so that the reference monitor can be protected against tampering
  - **interprocess communication**, so that different parts of the TCB can pass data to and activate other parts; for example, the reference monitor can invoke and pass data securely to the audit routine

The TCB is separated to achieve self-protection and independence.
The TCB must maintain the secrecy and integrity of each domain.



Trusted Systems

Trusted Systems Characteristics

Secure startup

- A tricky time for security, as most systems load basic I/O functionality before being able to load security functions
- Ensures no malicious code can block or interfere with security enforcement.

Trusted path

- Unforgeable connection by which the user can be confident of communicating directly with the OS
- Precludes interference between a user and the security enforcement mechanisms of the operating system.

Trusted Systems

Trusted Systems Characteristics

Object reuse control

- OS clears memory before reassigning it to ensure that leftover data doesn’t become compromised
- Object sanitization ensures no leakage of data.

Audit

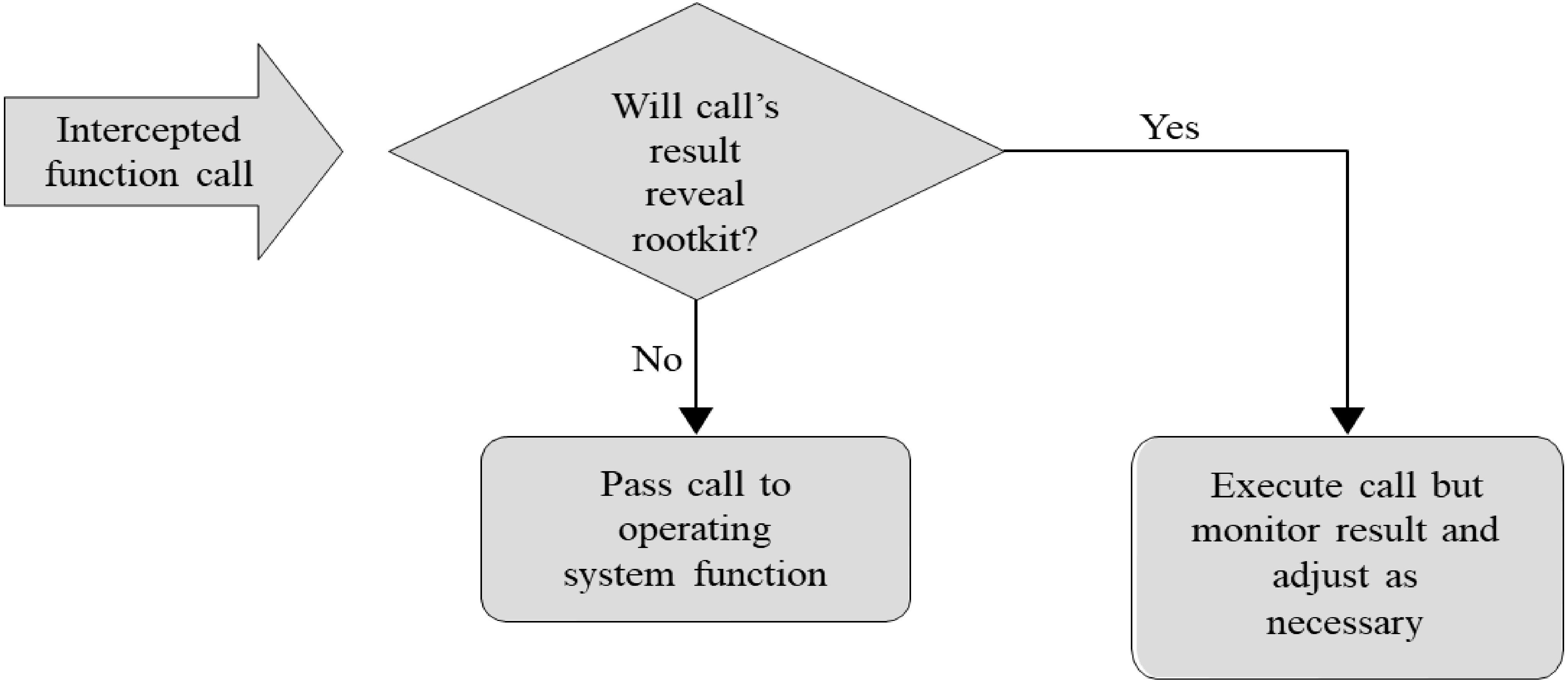
- Trusted systems track security-relevant changes, such as installation of new programs or OS modification
- Audit logs must be protected against tampering and deletion.

Attacks on OS

RootKit

- ✓ Root: most privileged subject (in a Unix system)
- ✓ A **rootkit** is a malicious software package that attains and takes advantage of root status or effectively becomes part of the OS.
- ✓ Rootkits often go to great length to avoid being discovered or, if discovered and partially removed, to re-establish themselves
  - This can include intercepting or modifying basic OS functions

Rootkit Evading Detection



Summary

- OSs have evolved from supporting single users and single programs to many users and programs at once
- Resources that require OS protection: memory, I/O devices, programs, and networks
- OSs use layered and modular designs for simplification and to separate critical functions from noncritical ones
- Resource access control can be enforced in a number of ways, including virtualization, segmentation, hardware memory protection, and reference monitors
- Rootkits are malicious software packages that attain root status or effectively become part of the OS