

Attacks on the OS

ICT 3156

Security in Operating Systems: Introduction

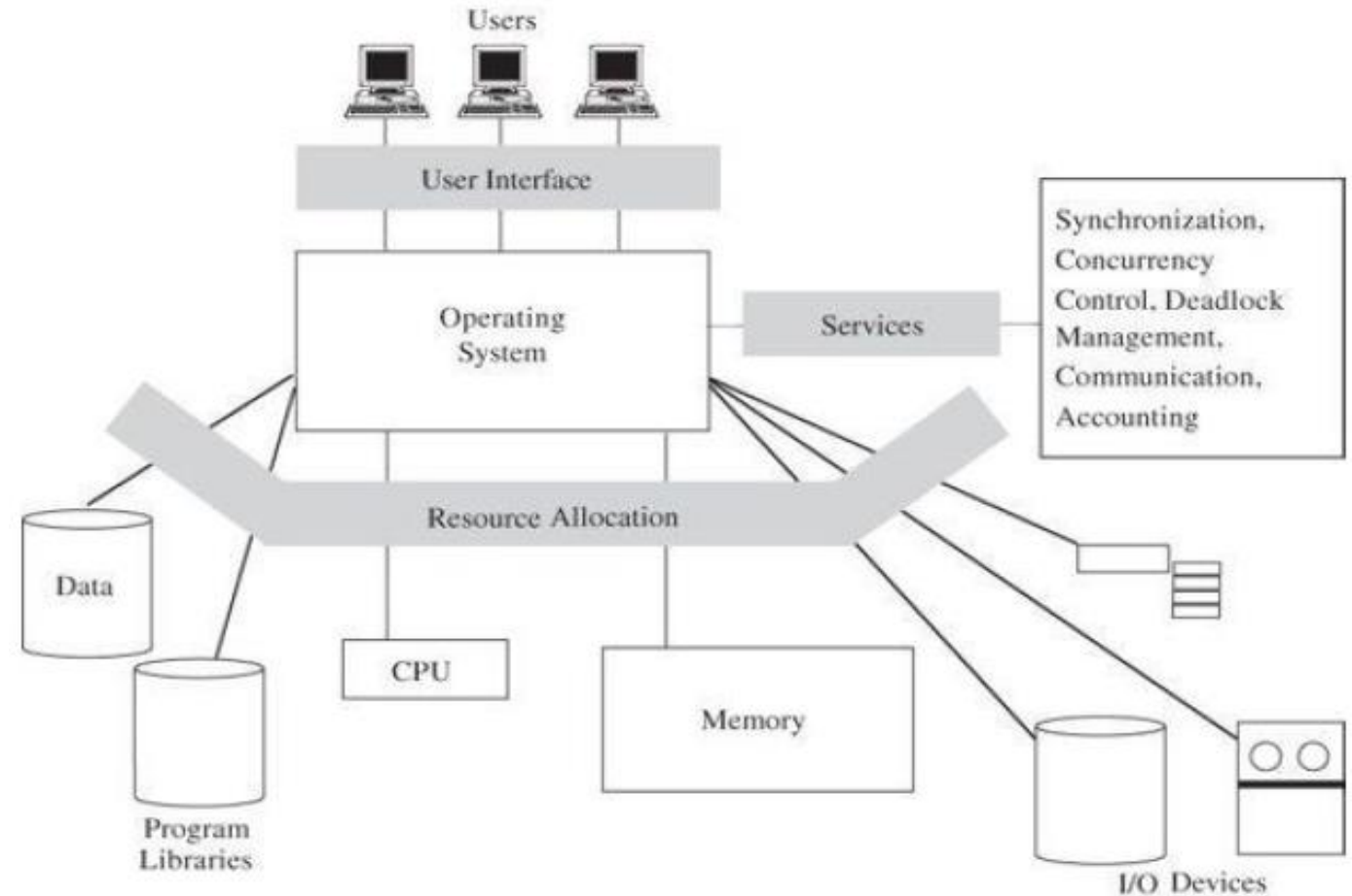
- The operating system is the fundamental controller of all system resources—which makes it a primary target of attack, as well.
- The malicious code files are stored somewhere, usually on disk or in memory.
- The operating system is the first line of defense against all sorts of unwanted behavior.
- Tasks:
 - It protects one user from another.
 - Ensures that critical areas of memory or storage are not overwritten by unauthorized processes.
 - Performs identification and authentication of people and remote operations.
 - Ensures fair sharing of critical hardware resources.

Security in Operating Systems: Introduction

- When the operating system initializes at system boot time, it initiates tasks in an orderly sequence:
 - Primitive functions and device drivers
 - Process controllers
 - File and memory management routines
 - the user interface
- To establish security, early tasks establish a firm defense to constrain later tasks.
- Antivirus applications are usually initiated late because they are add-ons to the operating system. Prevention software can protect only if it is active before the malicious code.
- What if the malware embeds itself in the operating system, such that it is active before operating system components that might detect or block it? Or what if the malware can circumvent or take over other parts of the operating system?

Background: Operating System Structure

- Operating systems are not just for conventional computers.
- Security Features of Ordinary Operating Systems.



Security Features of Ordinary Operating Systems

Enforced sharing (integrity, consistency)	Table lookup, combined with integrity controls
Inter-process communication and synchronization	Access control tables
Protection of critical operating system data	Encryption, hardware control, isolation, and others.
Guaranteed fair service.	Hardware clocks combine with scheduling disciplines to provide fairness. Hardware facilities and data tables combine to provide control.
Memory protection	Hardware mechanisms, such as paging or segmentation
File and I/O device access control	Table lookup
Allocation and access control to general objects	Table lookup
User authentication	Password Comparison or other such methods.

Protected Objects

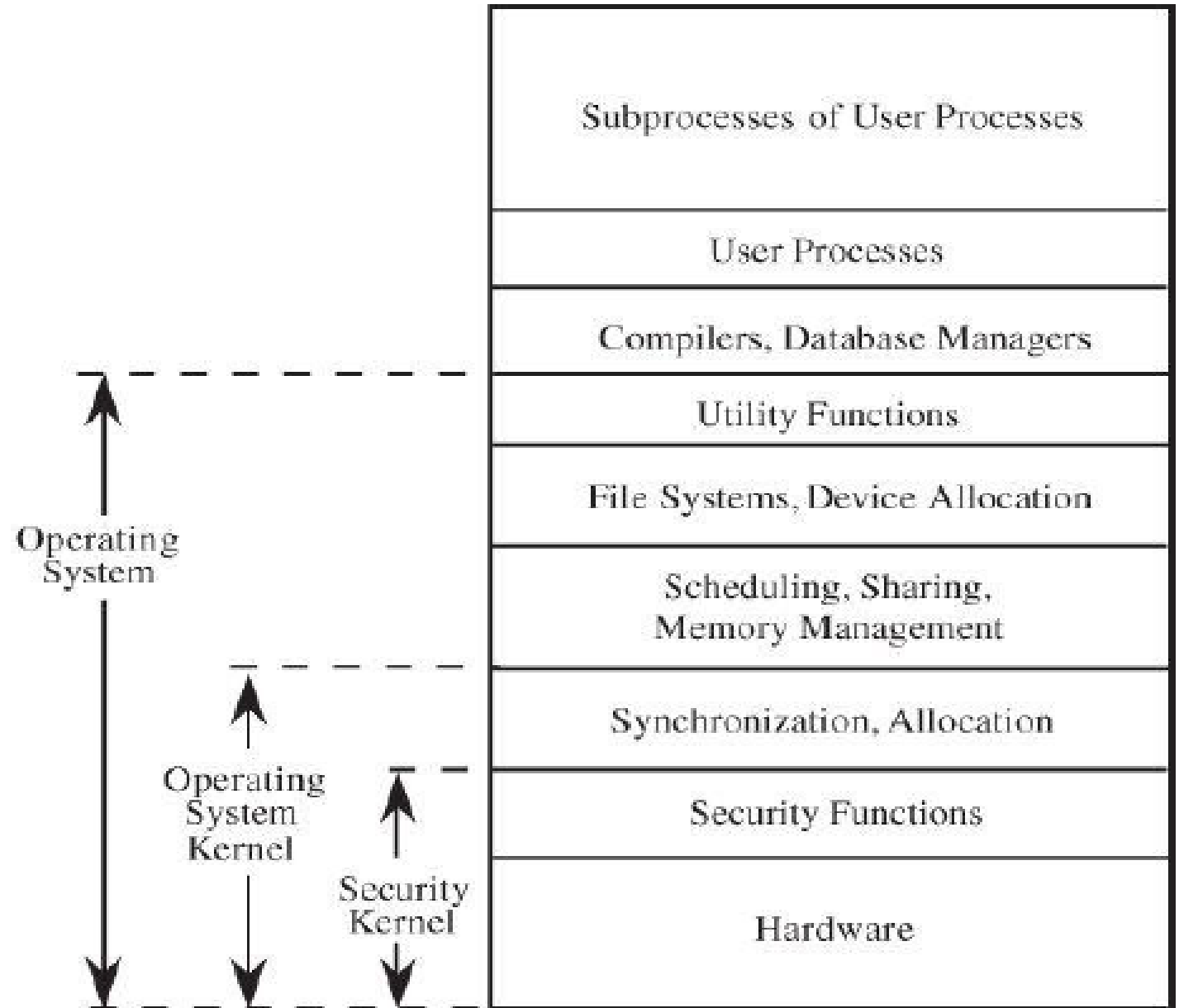
- Memory
- Sharable I/O devices, such as disks
- Serially reusable I/O devices, such as printers and tape drives
- Sharable programs and sub-procedures
- Networks
- Sharable data

Operating System Design to Protect Objects

Functions arranged from most critical (at the bottom) to least critical (at the top).

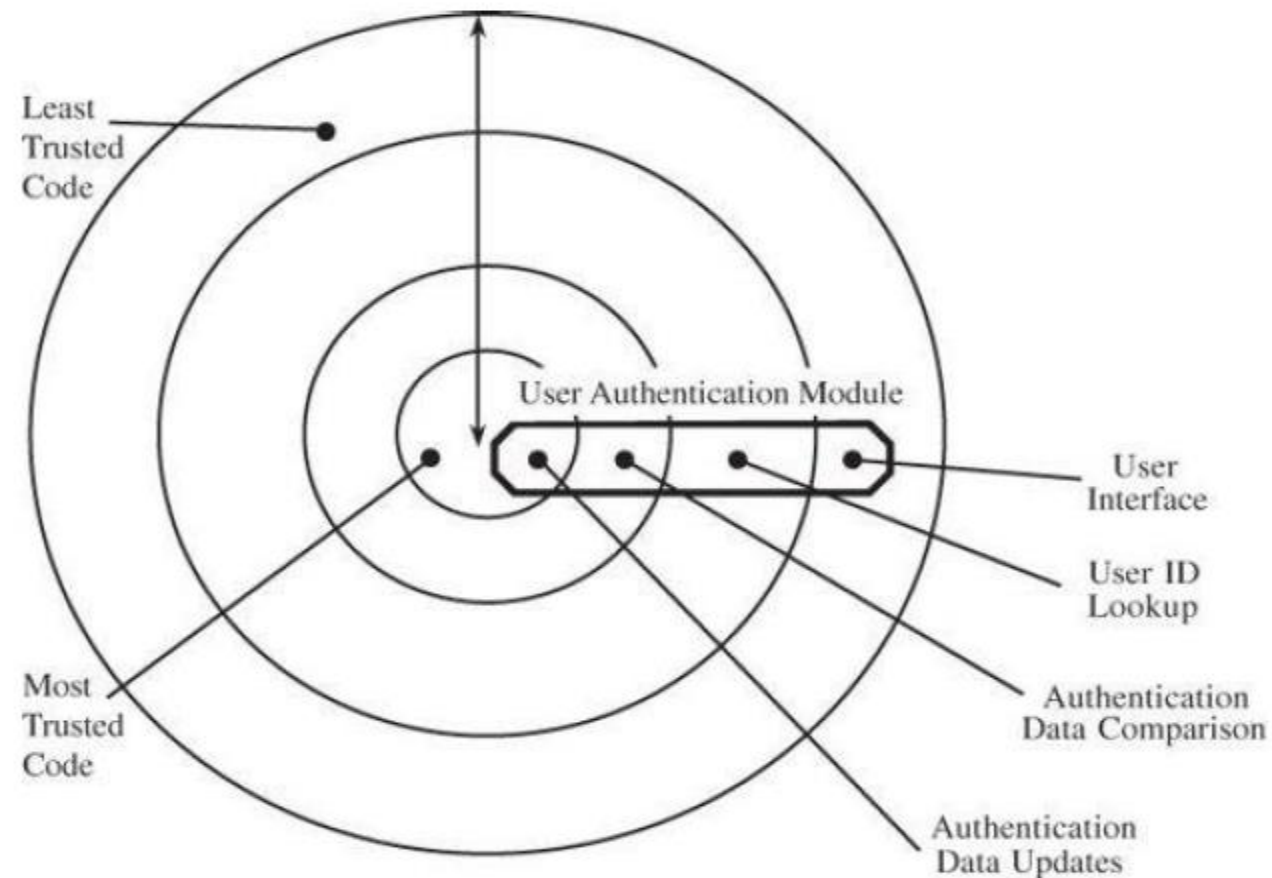
The functions are grouped in three categories:

1. Security kernel
2. Operating system kernel
3. Other operating system functions



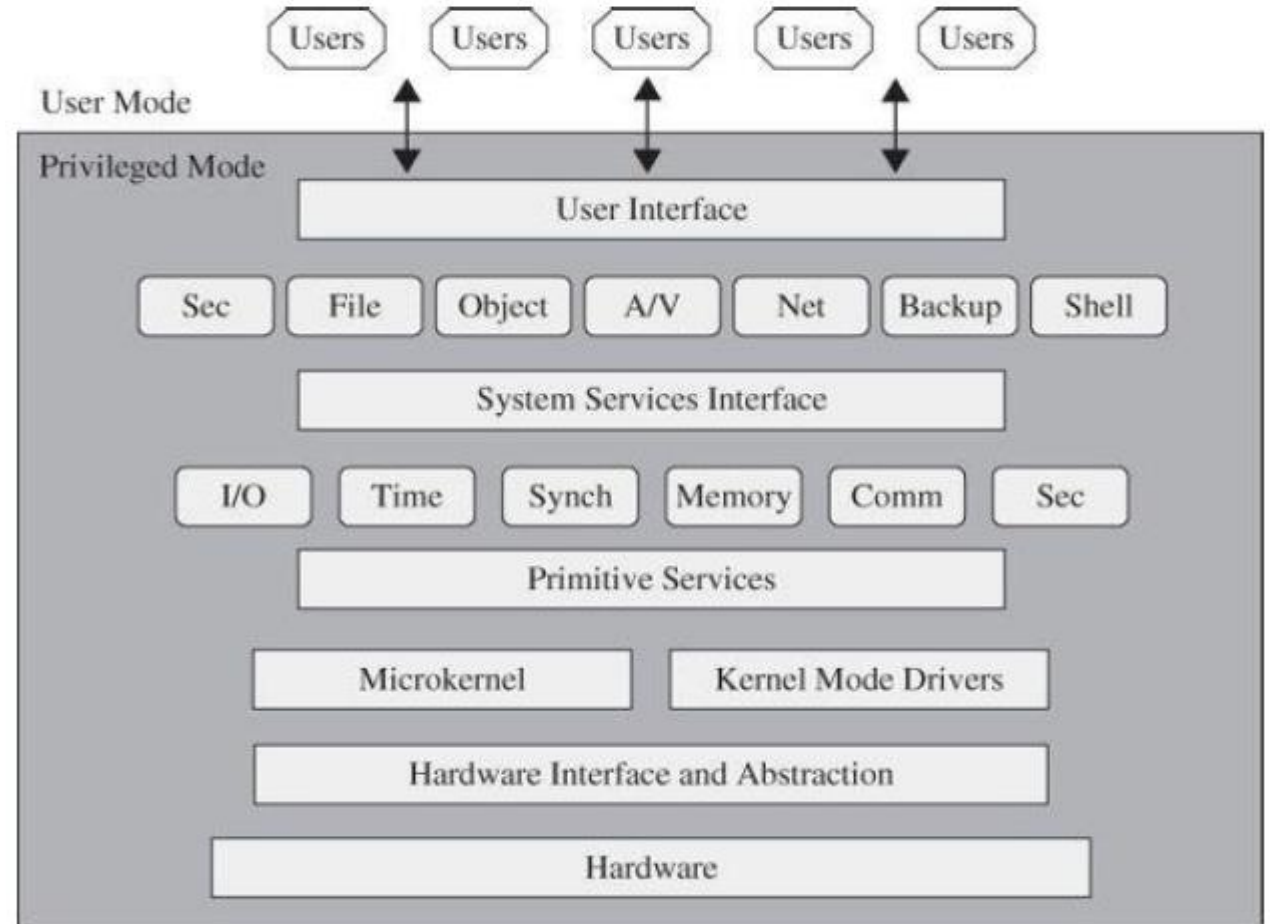
Operating System Design to Protect Objects

- The critical functions of controlling hardware and enforcing security are said to be in lower or inner layers, and the less critical functions in the upper or outer layers.



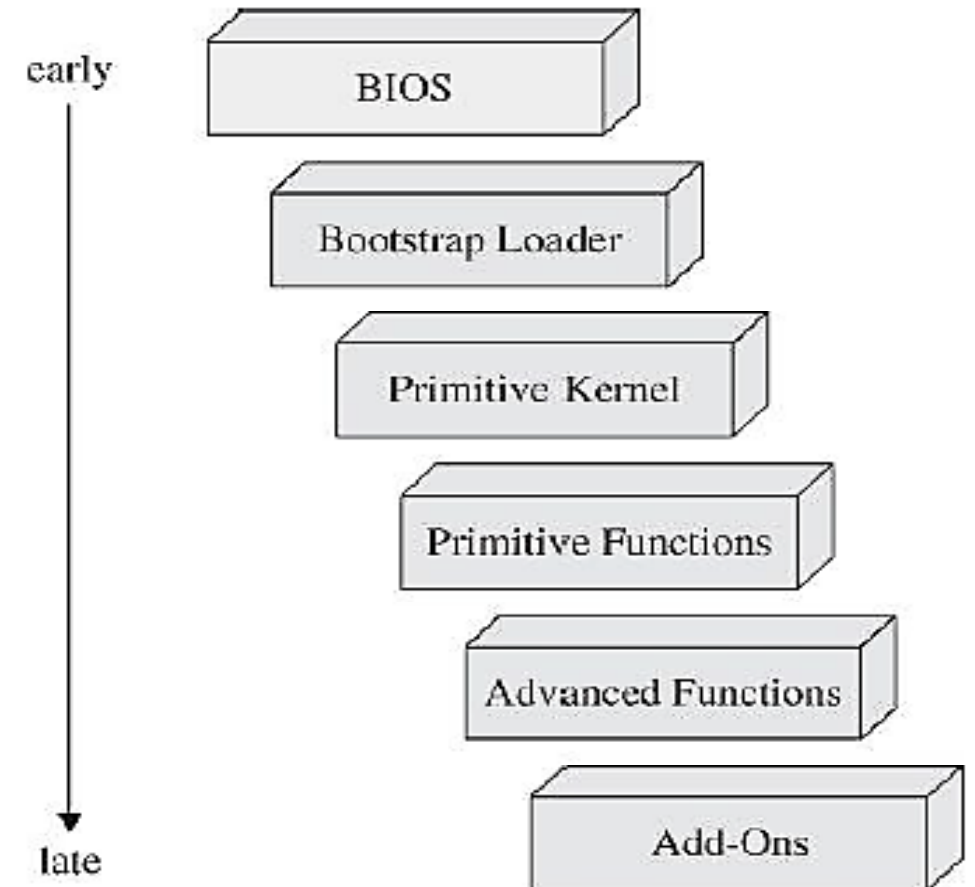
Operating System Design to Protect Objects

- From a security standpoint these modules come from different sources, not all trustworthy, and must all integrate successfully.
- All these pieces are maintained separately, so any module can change at any time, but such changes risk incompatibility.



Operating System Design for Self-Protection

- OS must protect itself not just from errant or malicious user programs but also from harm from incorporated modules, drivers, and add-ons, and with limited knowledge of which ones to trust and for what capabilities.
- The complexity of timing, coordination, and hand-offs in operating system design and **activation** is enormous.



OS Tools to Implement Security Functions

- ACL
- Audit Logs
- Virtualization
- Hypervisor
- Sand-box
- Honey pot
- Separation and Sharing
- Hardware Protection of Memory

A subject is permitted to access an object in a particular mode, and only such authorized accesses are allowed.

A log of which subject accessed which object when and in what manner. Auditing is a tool for **reacting after a security breach**, not for preventing one.

Presenting a user the appearance of a system with only the resources the user is entitled to use.

OS Tools to Implement Security Functions

- ACL
- Audit Logs
- Virtualization
- Hypervisor
- Sand-box
- Honey pot
- Separation and Sharing
- Hardware Protection of Memory

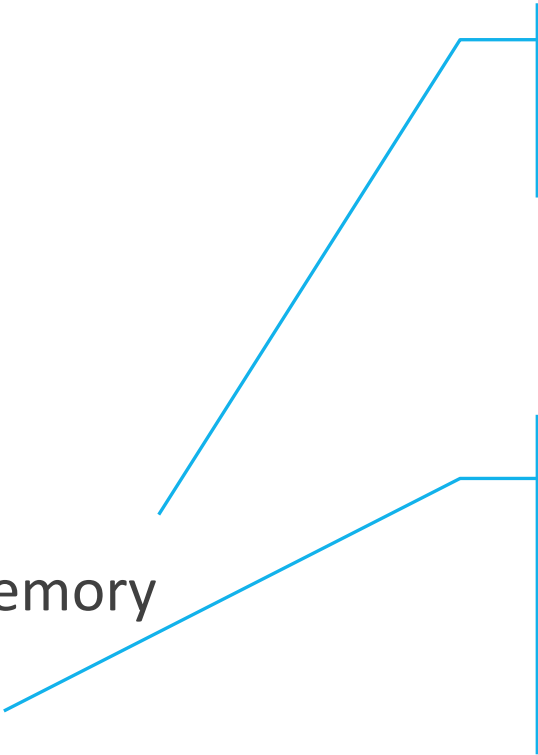
It receives all user access requests, directly passes along those that apply to real resources the user is allowed to access, and redirects other requests to the virtualized resources.

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

System to lure an attacker into a faux environment that can be both controlled and monitored.

OS Tools to Implement Security Functions

- ACL
- Audit Logs
- Virtualization
- Hypervisor
- Sand-box
- Honey pot
- Hardware Protection of Memory
- Separation and Sharing



Memory protection implements both separation and sharing.

Keeping one user's objects separate from other users.
Separation occurs by space (physical), time (temporal), access control (logical), or cryptography.

Separation In An Operating System

- Separation in an operating system can occur in several ways:
- **Physical separation**, by which different processes use different physical objects, such as separate printers for output requiring different levels of security.
- **Temporal separation**, by which processes having different security requirements are executed at different times.
- **Logical separation**, by which users operate under the illusion that no other processes exist, as when an operating system constrains a program's accesses so that the program cannot access objects outside its permitted domain.
- **Cryptographic separation**, by which processes conceal their data and computations in such a way that they are unintelligible to outside processes.

Security in the Design of Operating Systems

- The operating system opens many points to which code can later attach as pieces are loaded during the boot process; if one of these pieces is not present, the malicious code can attach instead.
- The more complex the software, the more possibilities for unwanted software introduction.
- Simple, modular, loosely coupled designs present fewer opportunities to the attacker.

Security in the Design of Operating Systems

1. Layered Design

- A nontrivial operating system consists of at least four levels:
 - Hardware
 - Kernel
 - Operating System
 - User
- Each of these layers can include sublayers.
- The trustworthiness and access rights of a process can be judged by the process's proximity to the center: The more trusted processes are closer to the center or bottom.

Security in the Design of Operating Systems

1. Layered Design

- Of the four ways to implement separations in OS, logical separation is most applicable to layered design.
- This means a fundamental (inner or lower) part of the OS must control the accesses of all outer or higher layers to enforce separation.
- Some lower-level layers present some or all of their functionality to higher levels, but each layer properly encapsulates those things below itself. **What does this remind you of?**
- One can “peel off” each layer and still have a logically complete system with less functionality.
- Layering presents a good example of how to trade off and balance design characteristics.
- Another justification for layering is damage control.

Security in the Design of Operating Systems

1. Layered Design

- Hierarchical structuring has two benefits:
 - Permits identification of the most critical parts, which can then be analyzed intensely for correctness, so the number of problems should be smaller.
 - Isolation limits effects of problems to the hierarchical levels at and above the point of the problem, so the harmful effects of many problems should be confined.
- **Layering ensures that a security problem affects only less sensitive layers.**

Security in the Design of Operating Systems

2. Kernelized Design

- A kernel is the part of an operating system that performs the lowest-level functions.
- A security kernel is responsible for enforcing the security mechanisms of the entire operating system. Typically, the operating system is designed so that the security kernel is contained within the _____.

Security in the Design of Operating Systems

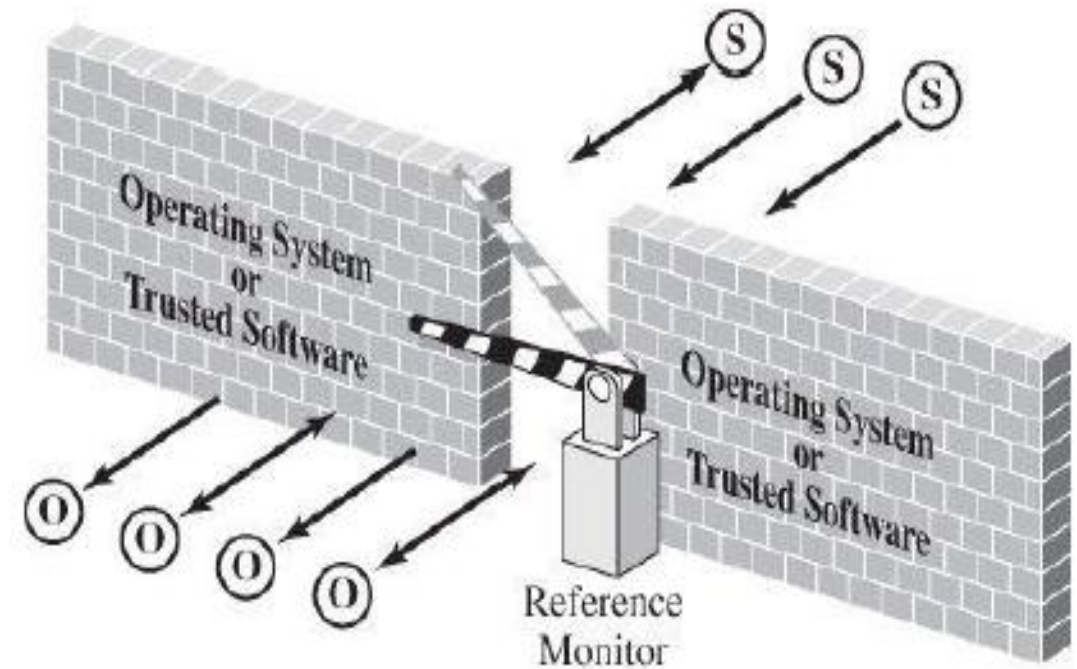
2. Kernelized Design

Good design reasons why security functions may be isolated in a security kernel.

- **Coverage**. Every access to a protected object must pass through the security kernel.
- **Separation**.
- **Unity**. All security functions are performed by a single set of code, so it is easier to trace the cause of any problems that arise with these functions.
- **Modifiability**. Changes to the security mechanisms are easier to make and easier to test
- **Compactness**. Because it performs only security functions, the security kernel is likely to be relatively small.
- **Verifiability**. Being relatively small, the security kernel can be analyzed rigorously.

Reference Monitor

- The most important part of a security kernel. Controls accesses to objects.
- Separates subjects and objects, enforcing that a subject can access only those objects expressly allowed by security policy.
- A reference monitor is not necessarily a single piece of code; rather, it is the collection of access controls for devices, files, memory, inter-process communication, and other kinds of objects.



Correctness and Completeness

- **Correctness** implies that because an operating system controls the interaction between subjects and objects, security must be considered in **every aspect of its design**.
 - The OS design must include definitions of which objects will be protected in what ways, what subjects will have access and at what levels, and so on.
 - There must be a clear mapping from the security requirements to the design so that all developers can see how the two relate.
-
- **Completeness** requires that security functionality be included in all places necessary.
 - Security seldom succeeds as an add-on; it must be part of the initial philosophy, requirements, design, and implementation.
 - **Security enforcement must be correct and complete.**

Secure Design Principles

- Least privilege
- Economy of mechanism
- Open design
- Complete mediation
- Permission based
- Separation of privilege
- Least common mechanism
- Ease of use

These principles are articulated well by
Jerome Saltzer and Michael Schroeder.

Computer Security: Arts and Science by Matt Bishop

Rootkit

- **Root**: most privileged subject (in a Unix system).
- It is the name of the entity (subject) established to own and run all primitive system tasks.
- **Rootkit**: Tool or script that obtains privileges of root.
- A rootkit is a piece of malicious code that goes to great lengths not to be discovered or, if discovered and removed, to reestablish itself whenever possible.
- Two conditions that help avoid discovery:
 - Rootkit code executing before other programs that might block rootkit execution.
 - The rootkit not being detected as a file or process.
- Being in control early in the system boot cycle would allow you to control the other system defenses instead of their controlling you.

Phone Rootkit

- The OS of smartphones have rich functionality.
- The complexity of the operating system led to more opportunities for attack and, ultimately, a rootkit.
- What a phone rootkit could do?
 - Could turn on a phone's microphone without the owner's knowing it happened.
 - Respond to a text query by relaying the phone's location as furnished by the GPS receiver.
 - Could turn on power-hungry capabilities—such as the Bluetooth radio and GPS receiver—to quickly drain the battery.
- The worst part of these three attacks is that they are effectively undetectable.

Rootkit Evades Detection

- Malicious code are of a certain name, size, location, or form, but that same predictability makes them targets for tools that search for malicious code.
- Antivirus tools call built-in functions through an API to get **information**.
 - Query the disk, determine the disk format, identify files and where they are stored, find the file names and properties from an index table, or structure the results for use and display.

```
Directory of C:\WINNT\APPS

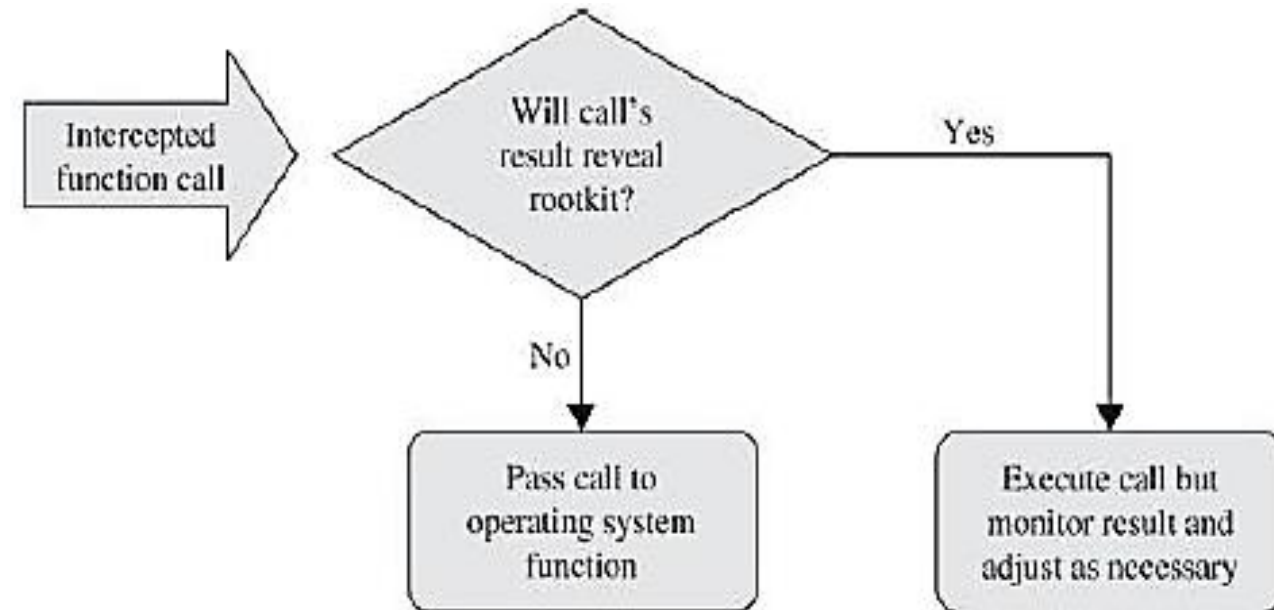
01-09-14  13:34          <DIR>      .
01-09-14  13:34          <DIR>      ..
24-07-12  15:00             82,944 CLOCK.AVI
24-07-12  15:00             17,062 Coffee Bean.bmp
24-07-12  15:00              80 EXPLORER.SCF
06-08-14  15:00          256,192 mal_code.exe
22-08-08  01:00          373,744 PTDOS.EXE
21-02-08  01:00             766 PTDOS.ICO
19-06-10  15:05          73,488 regedit.exe
24-07-12  15:00          35,600 TASKMAN.EXE
14-10-12  17:23          126,976 UNINST32.EXE
          9 File(s)      966,852 bytes
          2 Dir(s)  13,853,132,800 bytes free
```

```
Directory of C:\WINNT\APPS

01-09-14  13:34          <DIR>      .
01-09-14  13:34          <DIR>      ..
24-07-12  15:00             82,944 CLOCK.AVI
24-07-12  15:00             17,062 Coffee Bean.bmp
24-07-12  15:00              80 EXPLORER.SCF
22-08-08  01:00          373,744 PTDOS.EXE
21-02-08  01:00             766 PTDOS.ICO
19-06-10  15:05          73,488 regedit.exe
24-07-12  15:00          35,600 TASKMAN.EXE
14-10-12  17:23          126,976 UNINST32.EXE
          8 File(s)      710,660 bytes
          2 Dir(s)  13,853,472,768 bytes free
```

Rootkit Evades Detection

- The utility to present a file listing uses primitives such as FindNextFile() and NTQueryDirectoryObject.
- To remain invisible, the **rootkit intercepts these calls** so that if the result from FindNextFile() points to mal_code.exe, the rootkit skips that file and executes FindNextFile() again to find the next file after mal_code.exe.
- The higher-level utility to produce the listing keeps the **running total** of file sizes for the files of which it receives information, so the total in the listing correctly reports all files except mal_code.exe.



Rootkit Operates Unchecked

- Because they want to remain undiscovered, rootkits can be difficult to detect and eradicate, or even to count.
- Rootkits can also interfere with computer maintenance because their functionality can become intertwined with other operating system functions being modified.
- Rootkit Revealer.

Rootkit Kills Kernel Modification

- In February 2010, Microsoft issued its usual monthly set of OS updates, including one patch called MS10-015, rated “important.” The patch was to fix one previously publicized vulnerability and one unpublicized one.
- Some users who installed the patch suddenly found that their computers went into an unending loop of rebooting.
- Apparently on system startup the TDL-3 or Alureon rootkit built a table, using the fixed addresses of specific Windows kernel functions.
- In the Microsoft patch, these addresses were changed, so when TDL-3 received control and tried to invoke a (real) kernel function, it transferred to the wrong address and the system shut down with what is known as the “blue screen of death”.

Sony XCP Rootkit

- Installed when a Sony music CD was loaded and played.
- The XCP rootkit was installed from the Sony music CD to prevent a user from copying the tunes, while allowing the CD to be played as audio.
- To do this, it includes its own special music player that is allowed to play the CD.
- It intercepts any functional call to read from the CD drive.
- If the call originated from a music player for a Sony CD, XCP redirects the result to Sony's special music player.
- If the call was from any other application for a Sony CD, the rootkit scrambled the result so that it was meaningless as music and passed that uninterpretable result to the calling application.
- The rootkit has to install itself when the CD is first inserted in the PC's drive. ??

Sony XCP: Patching the Penetration

- Sony decided to release an uninstaller for the XCP rootkit.
- Fixing one fault often causes a failure somewhere else.
- Sony's uninstaller itself opened serious security holes.
- It was presented as a web page that downloaded and executed the uninstaller.
- The web page would run any code from any source, not just the intended uninstaller.
- The code to perform downloads and installations remained on the system even after XCP was uninstalled, meaning that the vulnerability persisted.

Book

- Pfleeger C. P., Pfleeger S. L. and Margulies J., Security in Computing (5e), Prentice Hall, 2015, Chapter 5.