# OS Security

What is the role of Operating Systems in security?

By Matthew Tancreti
For COM S 352
Iowa State University

## OS Security

“The operating system is the fundamental controller of all system resources which makes it a primary target of attack, as well.”

Reading Security in Computing by Charles P. Pfleeger, Shari Lawrence Pfleeger, and Jonathan Margulies, 2007.

What is the role of Operating Systems in security?

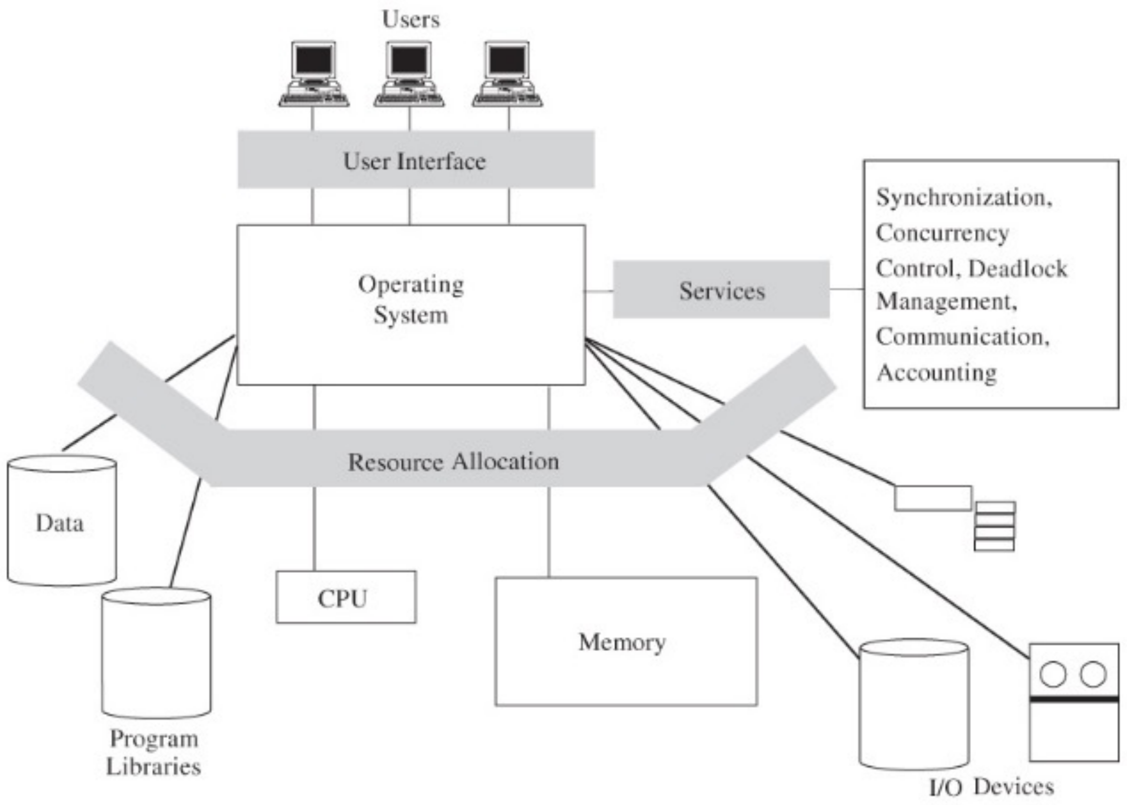
*Padlock* [source]

# Operating System Viewed as Resource Allocator

Job of OS is to provide access to shared **resources**

OS security is concerned about:

- **Access control** – who should be allowed to access a resource?
- **Integrity** – how to prevent corruption of resources (e.g., data and communications)?
- **Availability** – how to ensure resources are available?



## Common OS Security Features

**Enforced sharing** – resources should be shared only as appropriate

- Access control tables is common enforcement mechanism
- Table lookup is performed on every access

**Interposes communication and synchronization** – OS mediates communication between processes

- Pipes or shared memory need to restrict which processes can read or write
- Ends of a pipe are file descriptors which can only be shared by forking a child process

**Protection of critical OS data** – the OS must protect its own secret data (e.g., keys) from users

**Guaranteed fair service** – a process should not be able to "game the system" to get extra resources

- Recall a simple MLFQ allows a process to starve others
- Common solution is to add randomization
- Lottery scheduler is good example of guaranteeing fairness

## Common OS Security Features (cont.)

**Interface to hardware** – processes need access to hardware resources, the access should be restricted to appropriate use

**User authentication** – need method to identify users and verify they are who they purport to be

**Memory protection** – processes should be limited in what memory they can access

- Must prevent processes for accessing each others or the kernels memory inappropriately
- Paging and segmentation enforced by the hardware MMU provide protections

**File and I/O** – need to protect files from unauthorized users

- Common mechanism is access control matrix

**Allocation and access control to general objects** – other features that need to be protected include concurrency and synchronization

## Principles for Secure System Design

**Economy of mechanism** – keep systems as small and simple as possible, the more complex a system becomes the more likely it is to have vulnerabilities

**Fail-safe defaults** – default to the secure option over the insecure

**Complete mediation** – check if action to be performed meets security policy *every single time* action is to be taken, for example, access control

**Open design** – Assume the adversary knows every detail of the system's design, for example, encryption should not rely on the secrecy of the algorithm, only the key

"The Protection of Information in Computer Systems" by Jerome Saltzer and Michael Schroeder. Proceedings of the IEEE, Vol. 63, No. 9, September 1975. A highly influential paper, particularly their codification of principles for secure system design.

## Principles for Secure System Design (cont.)

**Separation of privilege** – require separate parties or credentials to perform critical actions, for example, two-factor authentication

**Least privilege** – give user or process the minimum privileges required to perform action

**Least common mechanism** – for different users or processes, use separate data structures or mechanisms to handle them, for example, each process has its own page table

**Acceptability** – if a security mechanism is so burdensome that users bypass or don't use it, then it is worthless

"The Protection of Information in Computer Systems" by Jerome Saltzer and Michael Schroeder. Proceedings of the IEEE, Vol. 63, No. 9, September 1975. A highly influential paper, particularly their codification of principles for secure system design.

## Defense in Depth

A basic principle of security design is **defense in depth**, multiple layers of security controls (defenses) protect the system

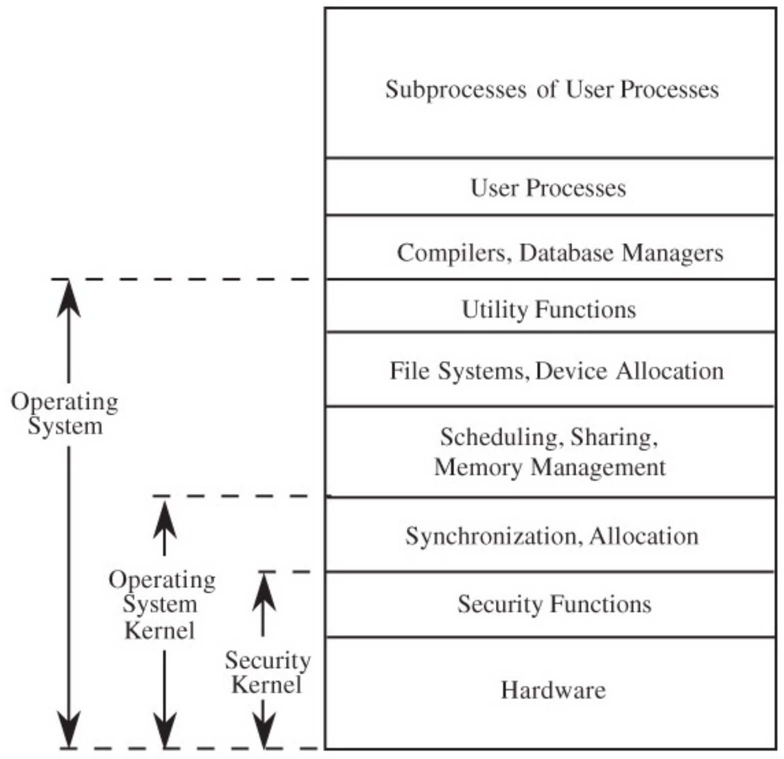
Protect the system using several independent methods, reduces chance that one one vulnerability can compromise entire system

Often conceived of as rings of security, the inner most core of the system is protected by the most layers

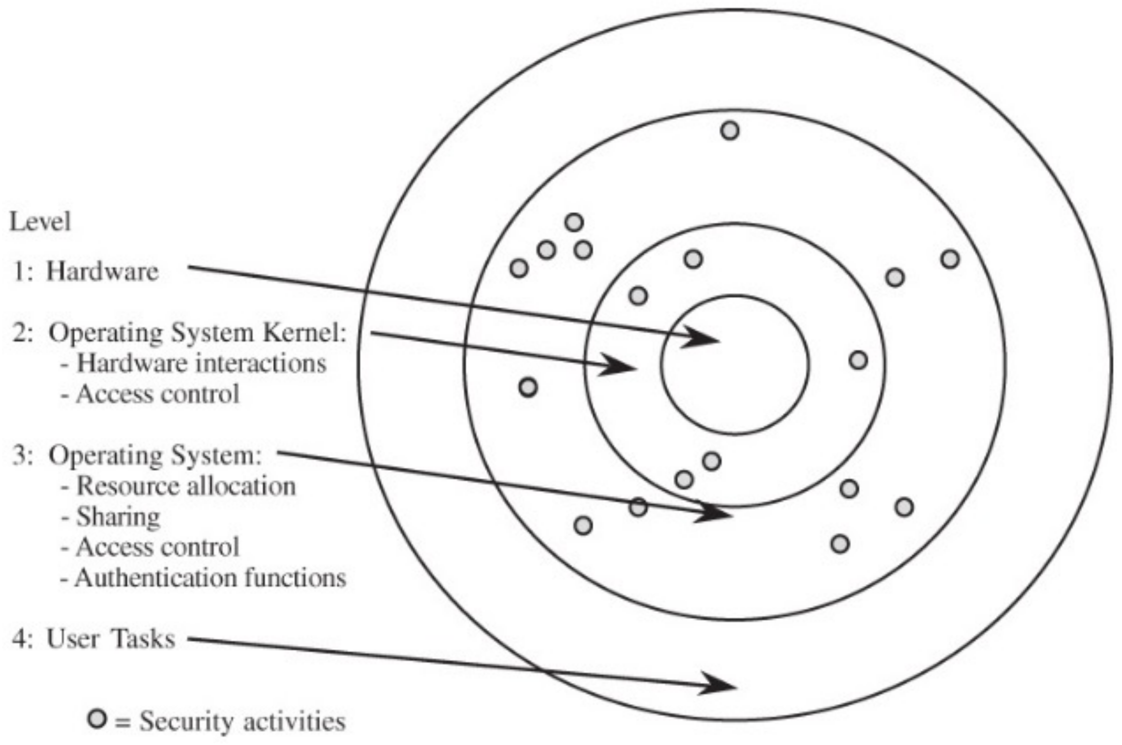
# Layered Architecture

**Layered architecture** follows the principle of defense in depth

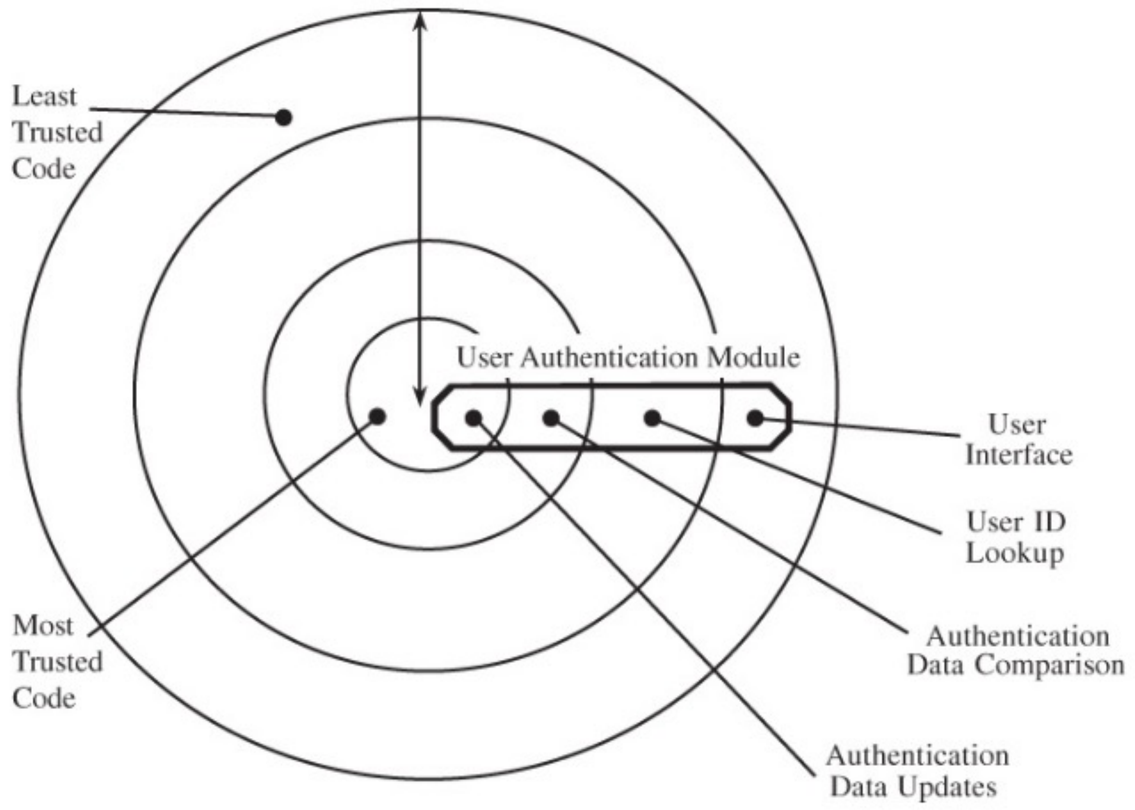
Often relies on hardware support to protect inner layers, e.g., kernel mode and machine mode



Subprocesses of User Processes
User Processes
Compilers, Database Managers
Utility Functions
File Systems, Device Allocation
Scheduling, Sharing, Memory Management
Synchronization, Allocation
Security Functions
Hardware
Operating System
Operating System Kernel
Security Kernel



# Example of Authentication in Layers

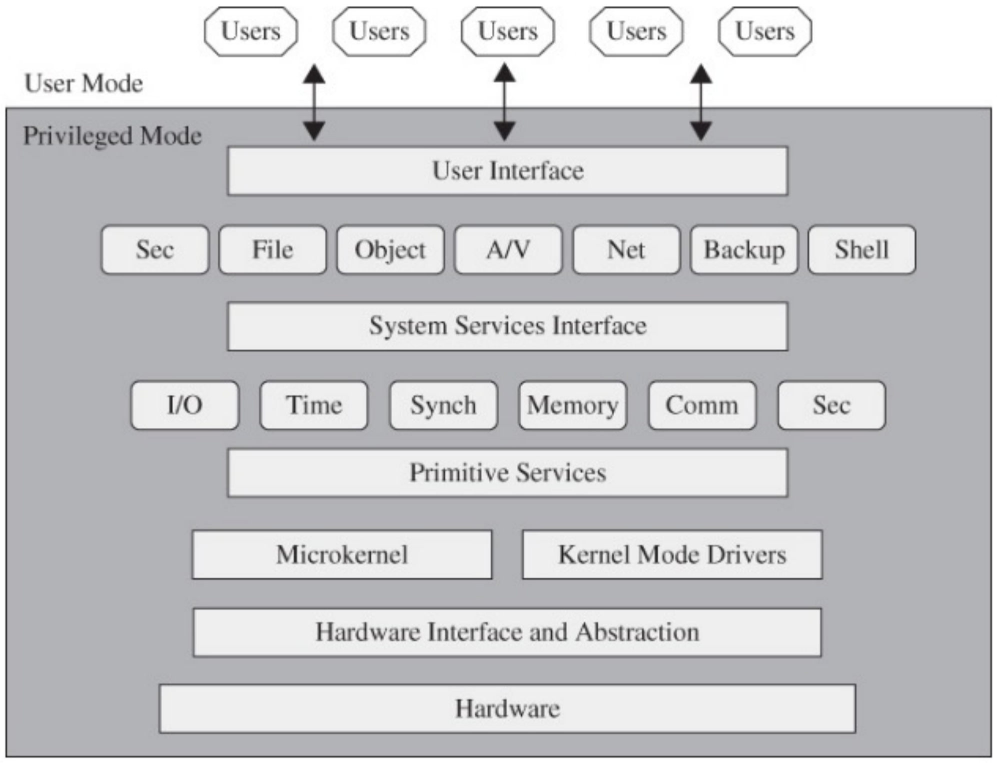

Least Trusted Code
User Authentication Module
User Interface
User ID Lookup
Authentication Data Comparison
Authentication Data Updates
Most Trusted Code

# Trust: Motivation for Layered Architecture

A strong motivation for defense in depth in OSes is **trust**

An OS consists of many modules: utilities, drivers, services (e.g., antivirus) and a kernel

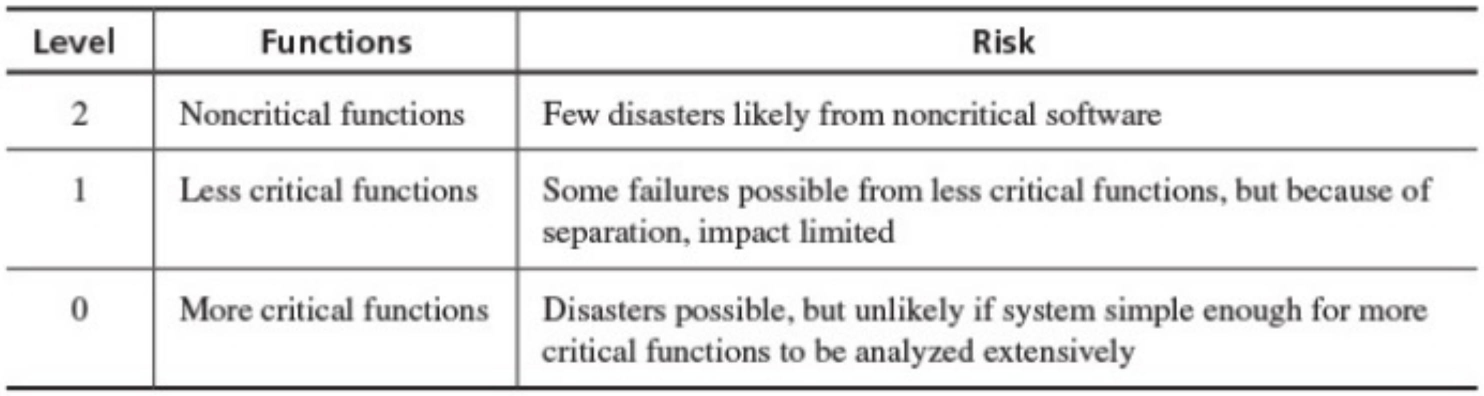
The modules are made by different vendors, do not want to trust all at the same level



## Design Principle: Layered Trust

A hierarchically designed system has layers of trust

Layers are isolated to limit effects of problems in one layer



| Level | Functions | Risk |
|---|---|---|
| 2 | Noncritical functions | Few disasters likely from noncritical software |
| 1 | Less critical functions | Some failures possible from less critical functions, but because of separation, impact limited |
| 0 | More critical functions | Disasters possible, but unlikely if system simple enough for more critical functions to be analyzed extensively |

## Trusted Systems

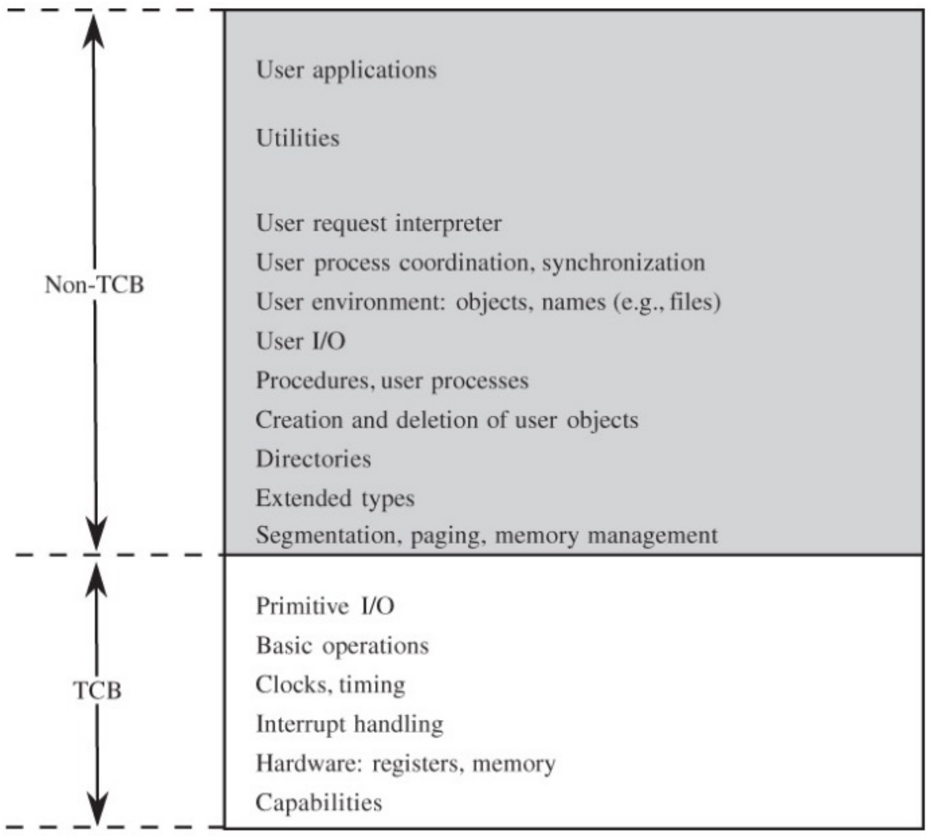
A **trusted system** is one that is relied upon to a specified extent to enforce a specified security policy

Rigorously developed and analyzed, trusted to be tamper-proof, will not execute unauthorized code

Used to act as a kernel or provide critical services (e.g., user authentication, secure boot, digital signature, encryption, etc.)

## Trusted Computing Base (TCB)

A **trusted computing base (TCB)** is the name given to everything in the trusted operating system that is necessary to enforce the security policy


Non-TCB
TCB
User applications
Utilities
User request interpreter
User process coordination, synchronization
User environment: objects, names (e.g., files)
User I/O
Procedures, user processes
Creation and deletion of user objects
Directories
Extended types
Segmentation, paging, memory management
Primitive I/O
Basic operations
Clocks, timing
Interrupt handling
Hardware: registers, memory
Capabilities

## Trusted Platform Module (TPM)

A **Trusted Platform Module (TPM)** is hardware that controls what can be done on the machine, for example, it assures booting into intended operating system

## Other Operating System Tools for Security

**Virtualization** – providing the appearance of one set of resources by using different resources

**Virtual Machine** – present to the users only the resources they need, giving the user the impression their program is running on its own machine

**Hypervisor (virtual machine monitor)** – software that implements a virtual machine

**Sandbox** – similar to virtualization, a protected environment in which a program can run and not endanger anything else on the system

**Honeypot** – a fake environment intended to lure an attacker