https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2025

Lecture 13: OS I

Co-Instructor: **Nikos Triandopoulos**

March 11, 2025

BROWN

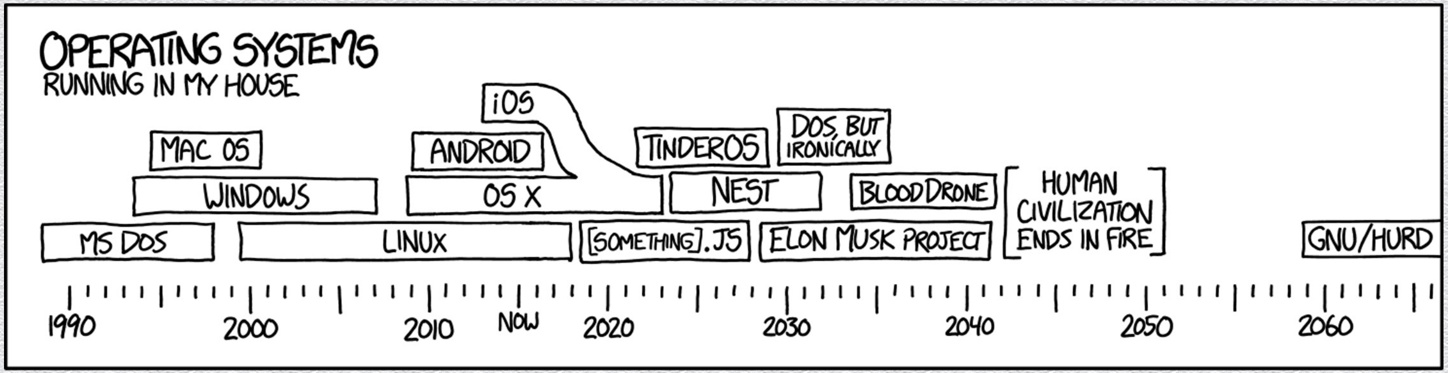
CS1660: Announcements

- Course updates
  - Project 2 is due Thursday, March 13
  - Homework 2 is now out and due Tuesday, March 18
  - Where we are
    - ✓ **Part I: Crypto**
    - ✓ **Part II: Web** (with demos coming soon)
    - ⇨ **Part III: OS**
    - Part IV: Network
    - Part V: Extras

Today

- OS security

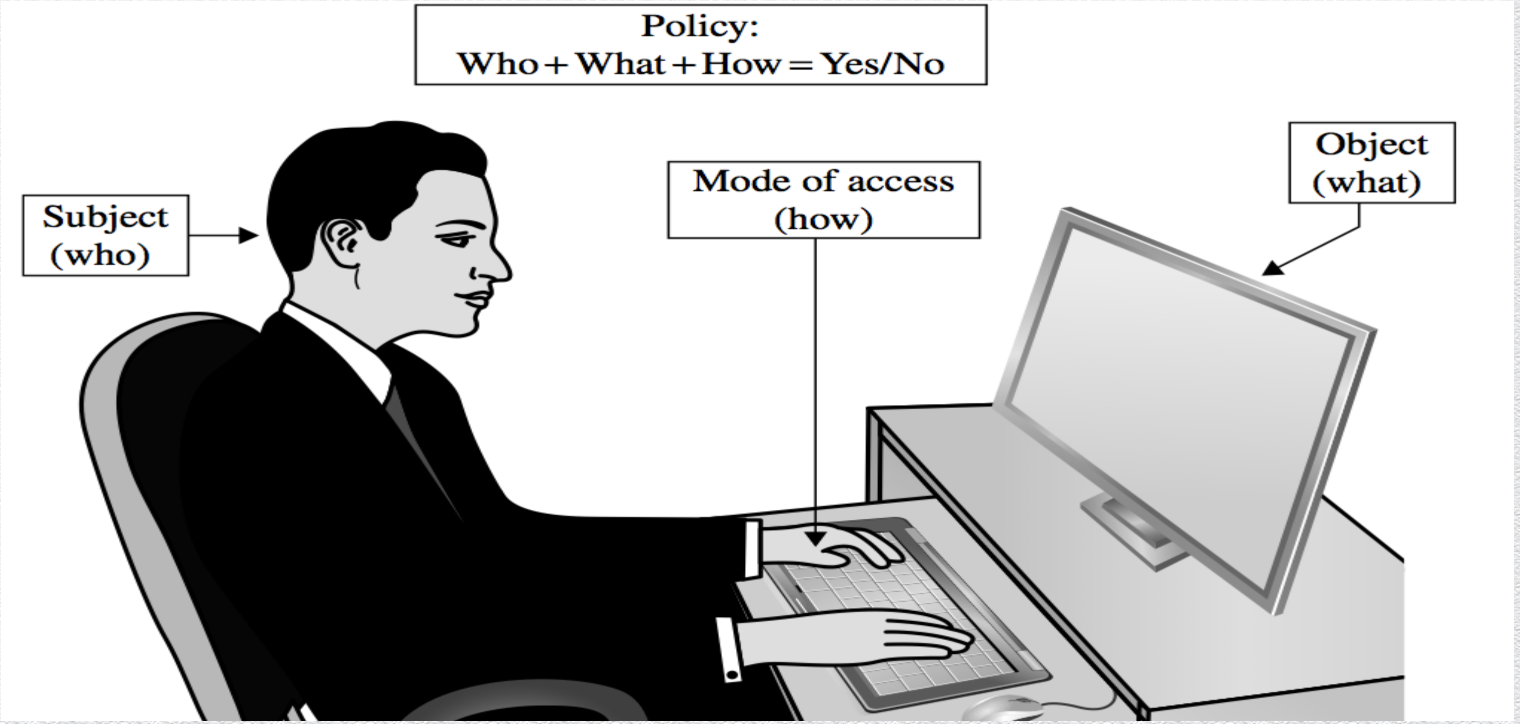
Operating Systems Security



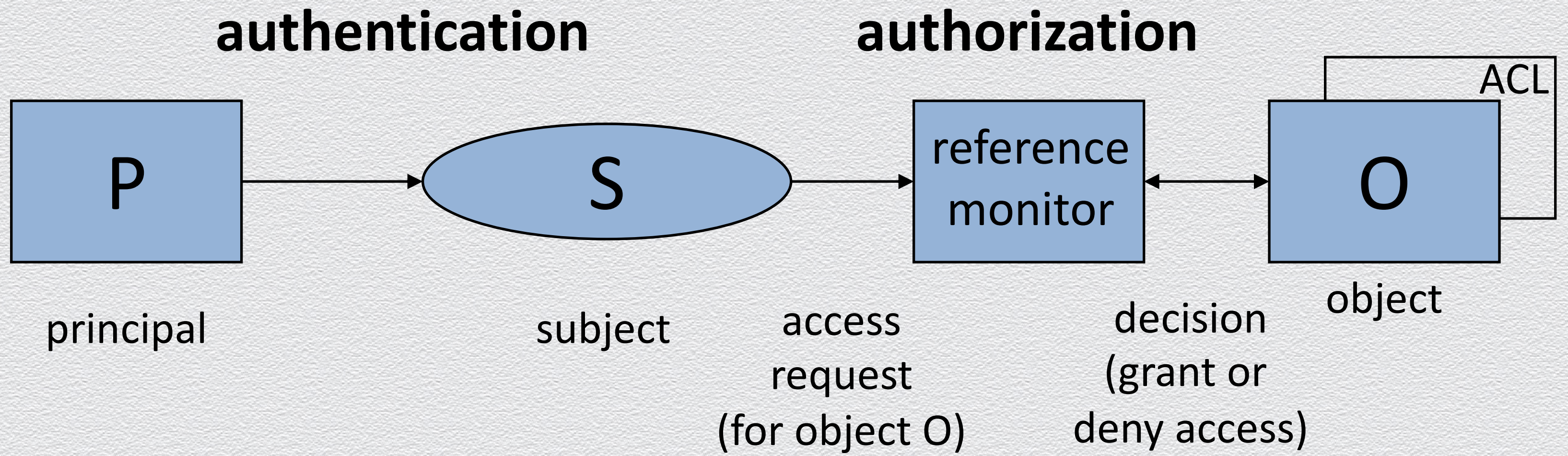
Source: XKCD

Access control

Access control (AC)



General structure of access control mechanism



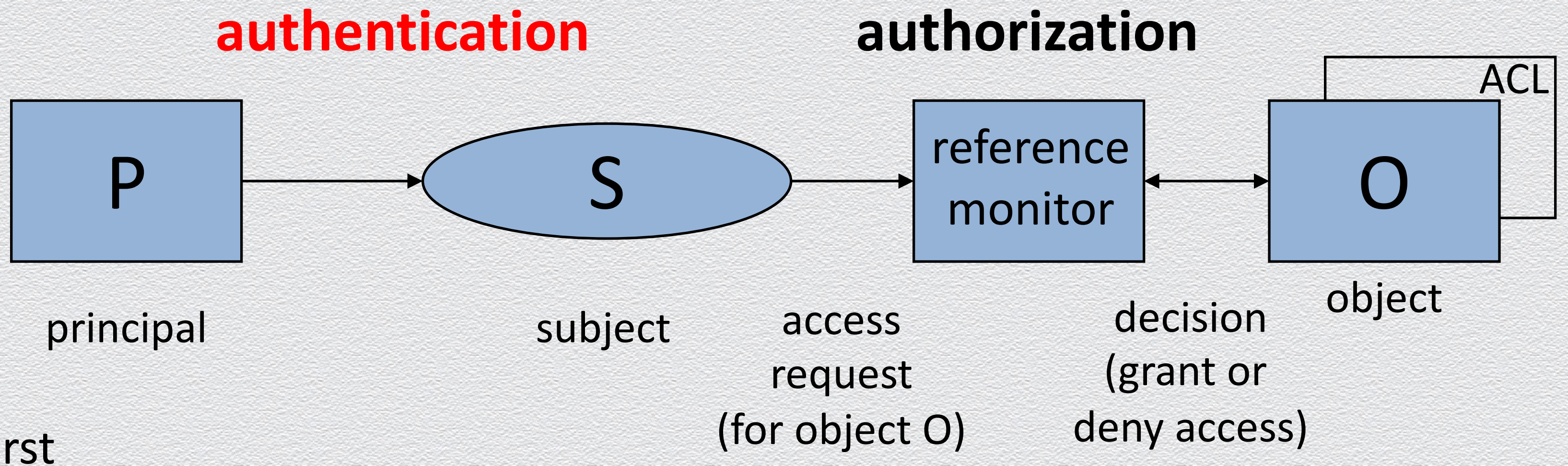
Basic terminology

- Subject/Principal
  - active entity – user or process
- Object
  - passive entity – file or resource
- Access operations
  - vary from basic memory access (read, write) to method calls in object-oriented systems
  - comparable systems may use different access operations or attach different meanings to operations which appear to be the same

Access operation

- Access right
  - right to perform an (access) operation
- Permission
  - typically a synonym for access right
- Privilege
  - typically a set of access rights given directly to roles like administrator, operator, ...

Authentication



First

- reference monitor verifies the identity of the principal making the request
  - a user identity is one example for a principal
  - cf. authentication Vs. identification

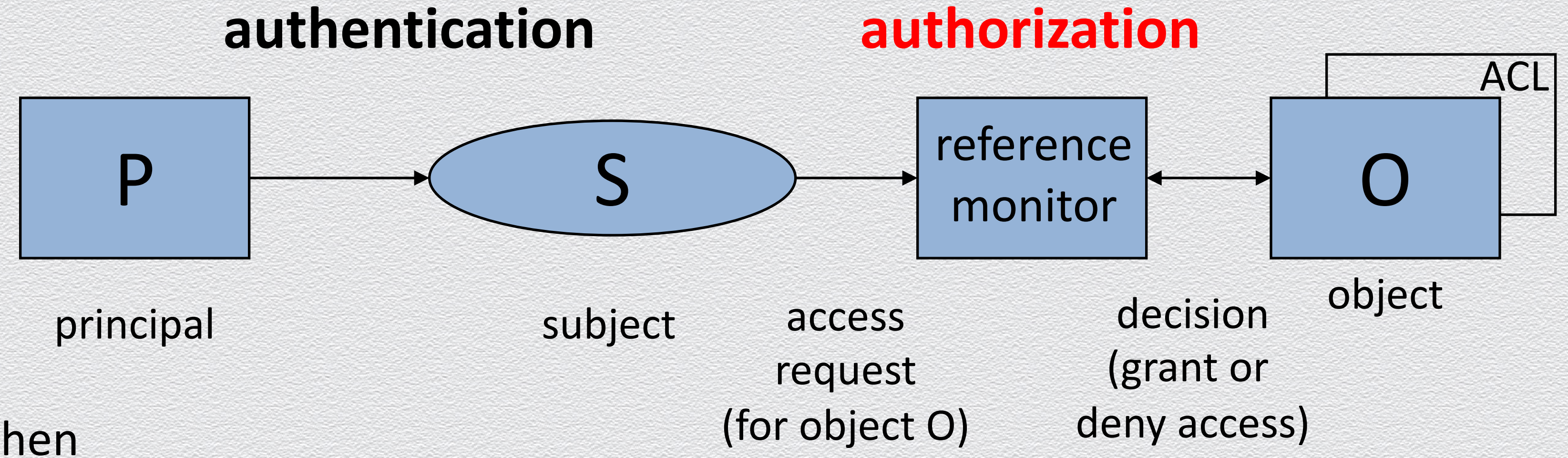
Authentication

- user enters username and password
- if the values entered are correct, the user is "authenticated"
- we could say: "The machine now runs on behalf of the user"
  - this might be intuitive, but it is imprecise
- log on creates a process that "runs with access rights" assigned to the user
  - the process runs under the user identity of the user who has logged on

Users & user identities

- requests to reference monitor do not come directly from a user or a user identity, but from a process
- in the language of access control, the process “speaks for” the user (identity)
- the active entity making a request within the system is called the subject
- must distinguish between three concepts
  - user: person
  - principal: identity (e.g., user name) used in the system, possibly associated with a user
  - subject: process running under a given user identity

Authorization



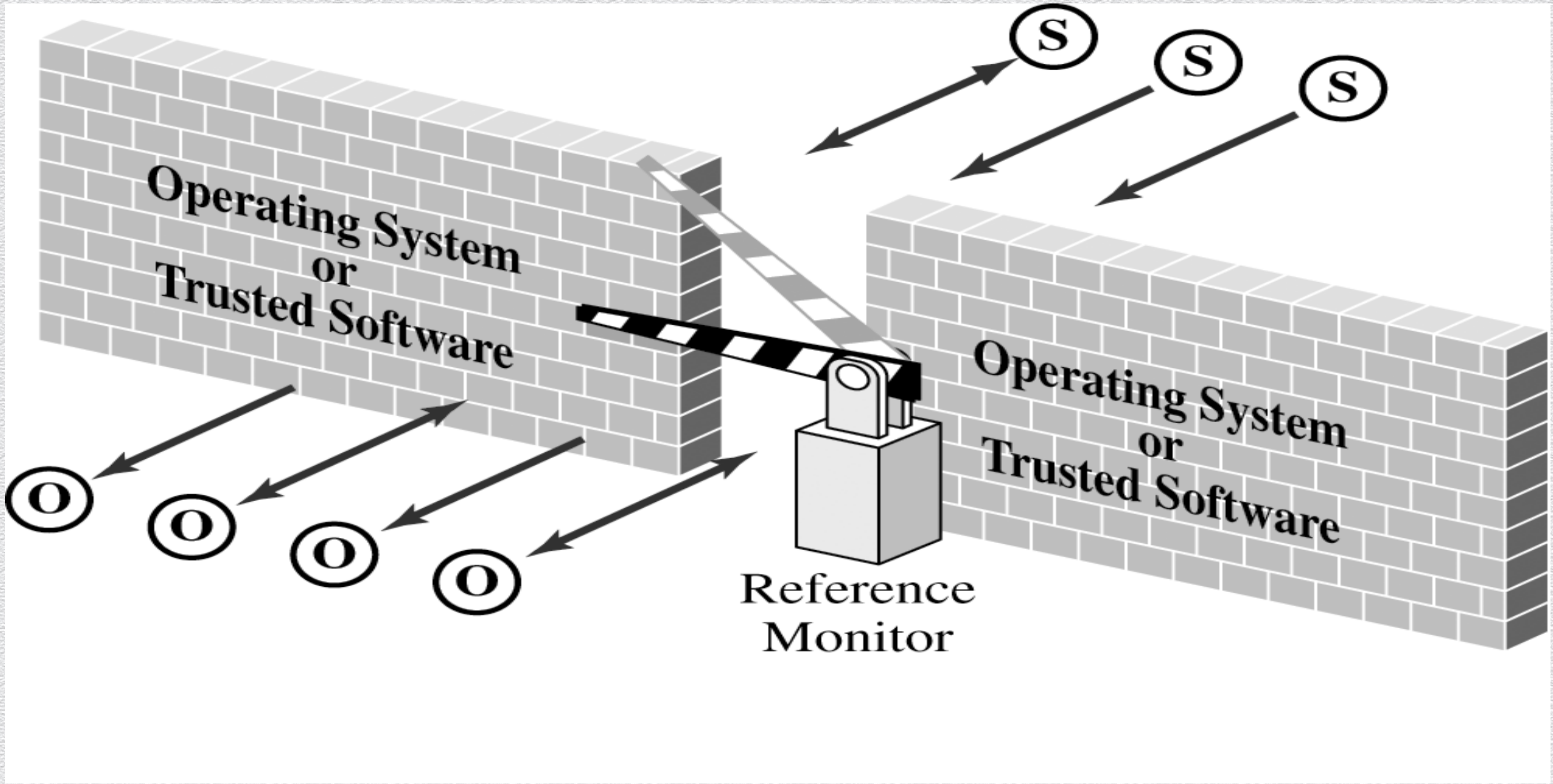
Then

- reference monitor decides whether access is granted or denied
- has to find and evaluate the security policy relevant for the given request
- "easy" in centralized systems; in distributed systems,
  - how to find all relevant policies? how to make decisions if policies may be missing?

Principals & subjects

- a principal is an entity that can be granted access to objects or can make statements affecting access control decisions
  - example: user ID
- subjects operate on behalf of (human users we call) principals
- access is based on the principal's name bound to the subject in some unforgeable manner at authentication time
  - example: process (running under a user ID)

Reference monitor



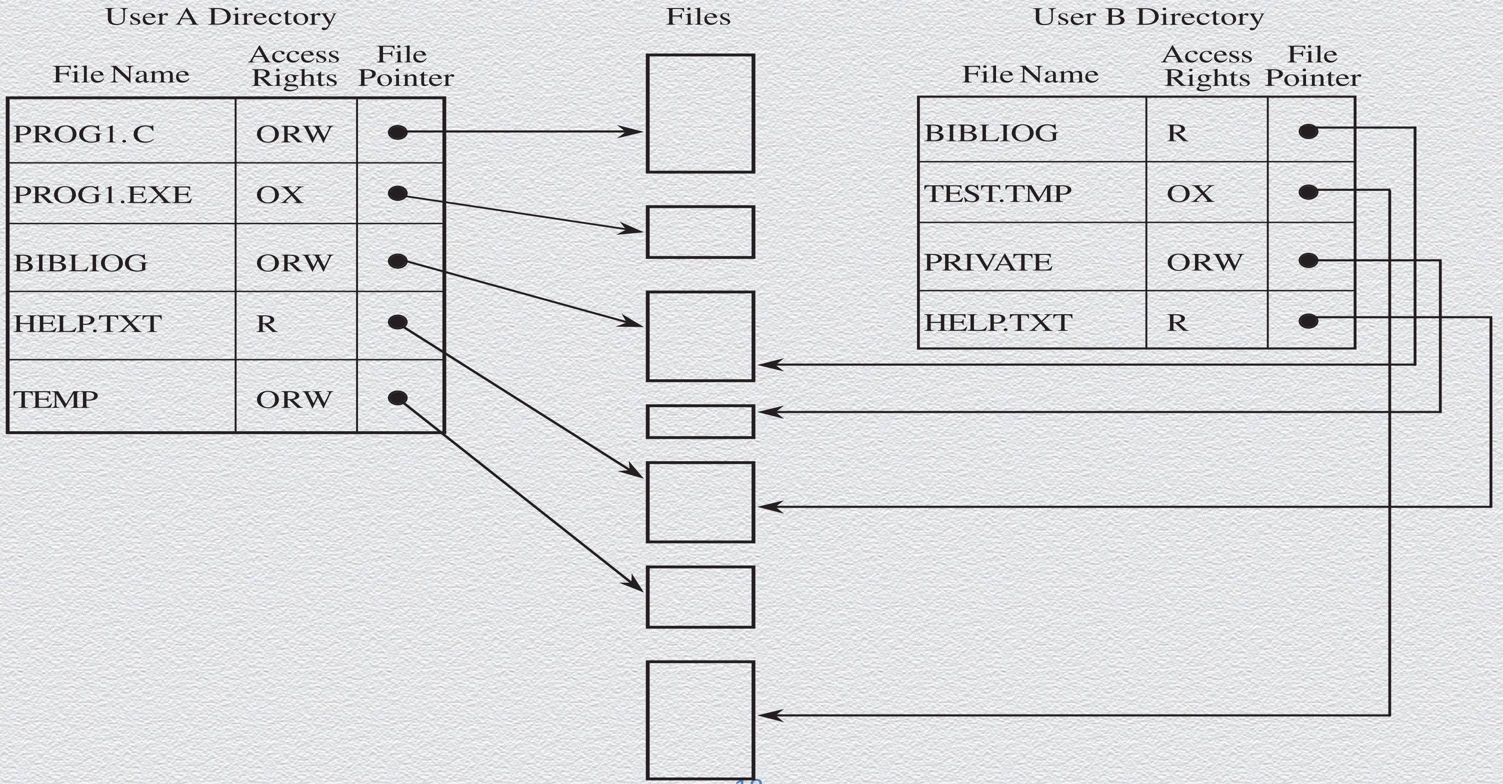
AC policies

- Goals
  - Check every access
  - Enforce least privilege
  - Verify acceptable usage
- Track users' access
- Enforce at appropriate granularity
- Use audit logging to track accesses

Implementing AC policies

- Reference monitor
- Access control directory
- Access control matrix
- Access control list
- Privilege list
- Capability
- Procedure-oriented access control
- Role-based access control

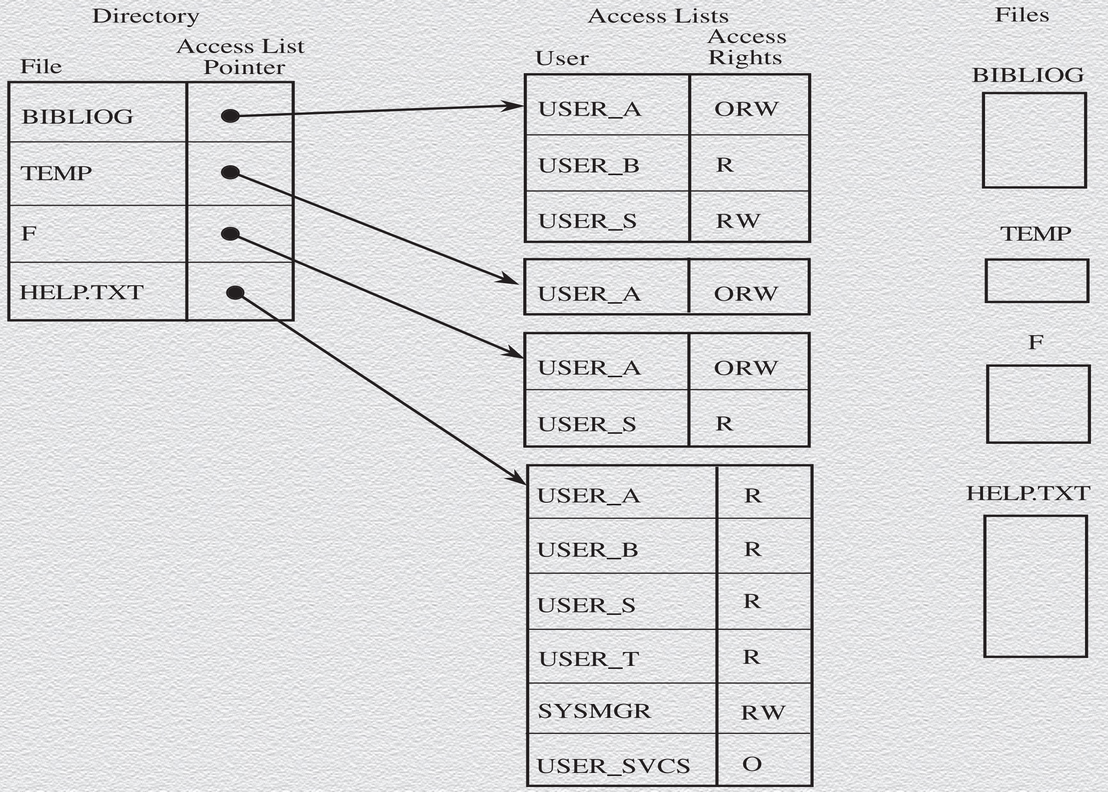
Access control directory



Access control matrix

	BIBLIOG	TEMP	F	HELP.TXT	C_COMP	LINKER	SYS_CLOCK	PRINTER
USER A	ORW	ORW	ORW	R	X	X	R	W
USER B	R	-	-	R	X	X	R	W
USER S	RW	-	R	R	X	X	R	W
USER T	-	-	-	R	X	X	R	W
SYS_MGR	-	-	-	RW	OX	OX	ORW	O
USER_SVCS	-	-	-	O	X	X	R	W

Access control list



Basic access control and information flow models

- Discretionary access control (DAC)
  - owner determines access rights
  - typically identity-based access control: access rights are assigned to users based on their identity
  - e.g., ACM
- Mandatory access control (MAC)
  - system enforce system-wide rules for access control
  - e.g., law allows a court to access driving records without the owners’ permission

DAC

- In DAC the user (e.g., owner of resources/files) is responsible for deciding how information is accessed
- Local access decisions of users might conflict with each other
- Basic terms
  - Access control matrix
  - Security policy (specifying who has the access rights to what)
  - Security mechanism (enforce security policies)

DAC and MAC

- When is DAC insufficient?
  - when owner cannot be trusted for the discretion of the data and external protection of the data is necessary
  - e.g., DAC has the danger of right propagation
    - A can read X and write Y
    - B can read Y, but no access to X
    - A reads X, write the content of X to Y, B got access to X
- MAC
  - non-discretionary
  - labels are assigned to subjects and objects
  - owner has no special privileges
  - e.g., Bell-LaPadula, lattices models, SELinux by NSA

Traditional models for MAC

- Bell-LaPadula (BLP)
  - About confidentiality
- Biba
  - About integrity with static/dynamic levels

Bell-LaPadula security model

- The Bell-LaPadula (BLP) model is about information confidentiality
- It was developed to formalize the US Department of Defense multilevel security policy

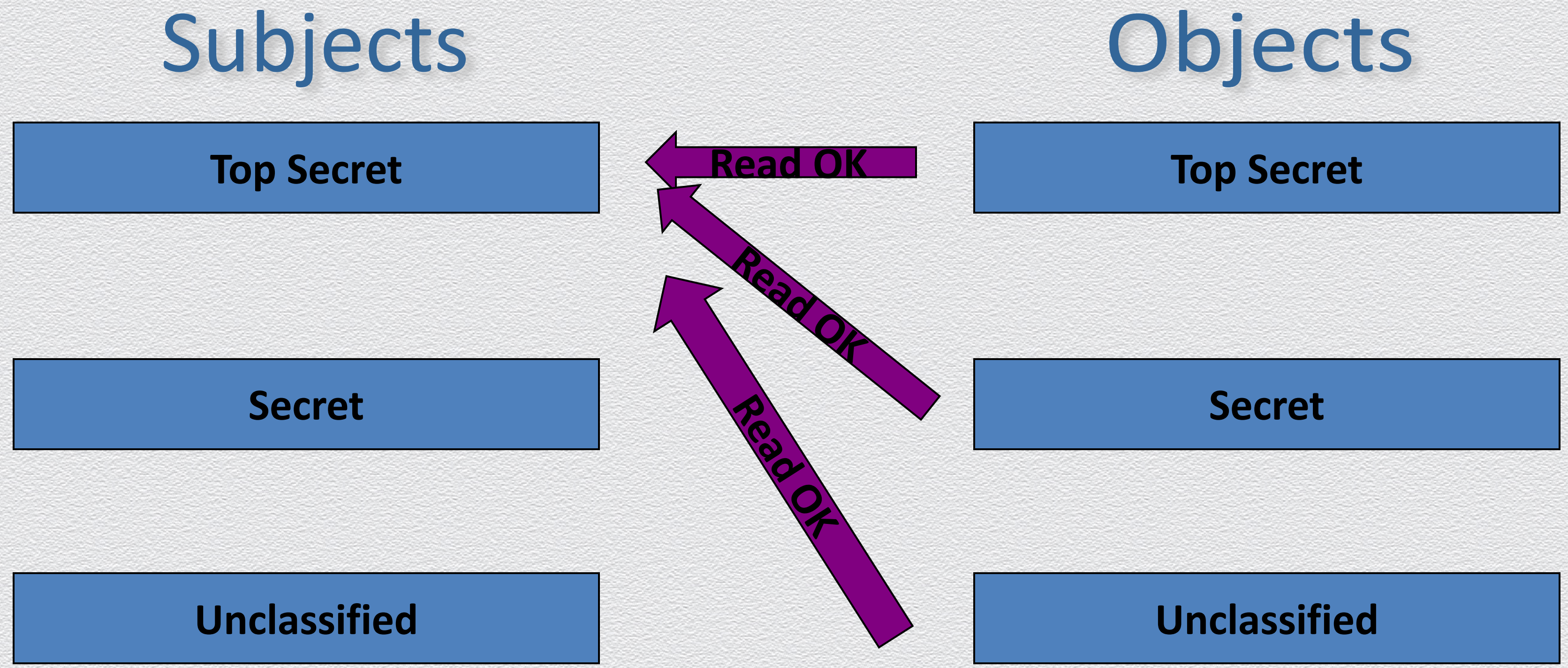
Bell – LaPadula - details

- Each user subject and information object has a fixed security class – labels
- Use the notation **≤** to indicate **dominance**
- Simple Security (ss) property:

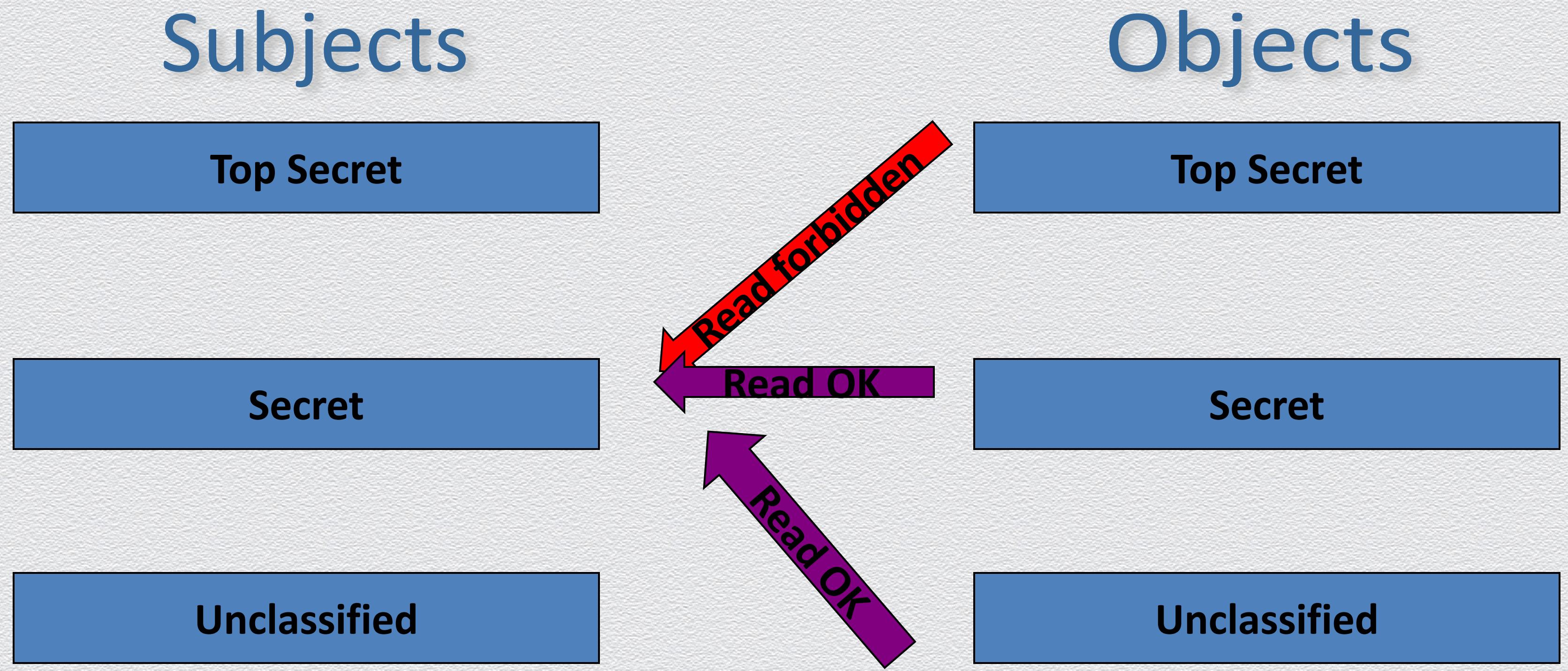
  no read-up property

  - a subject s has read access to an object o iff the class of the subject C(s) is greater than or equal to the class of the object C(o)
  - i.e. subjects s can read objects o iff $C(o) \leq C(s)$

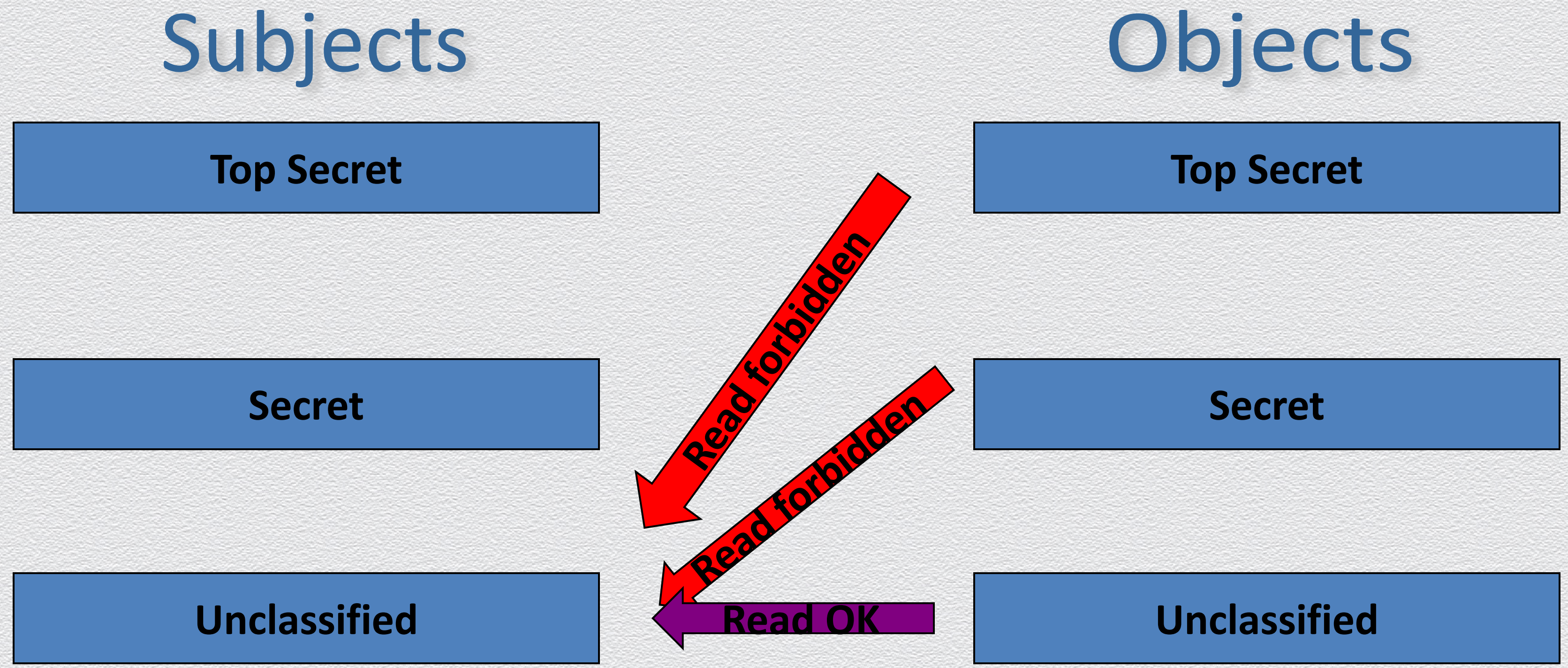
Access control: Bell-LaPadula



Access control: Bell-LaPadula



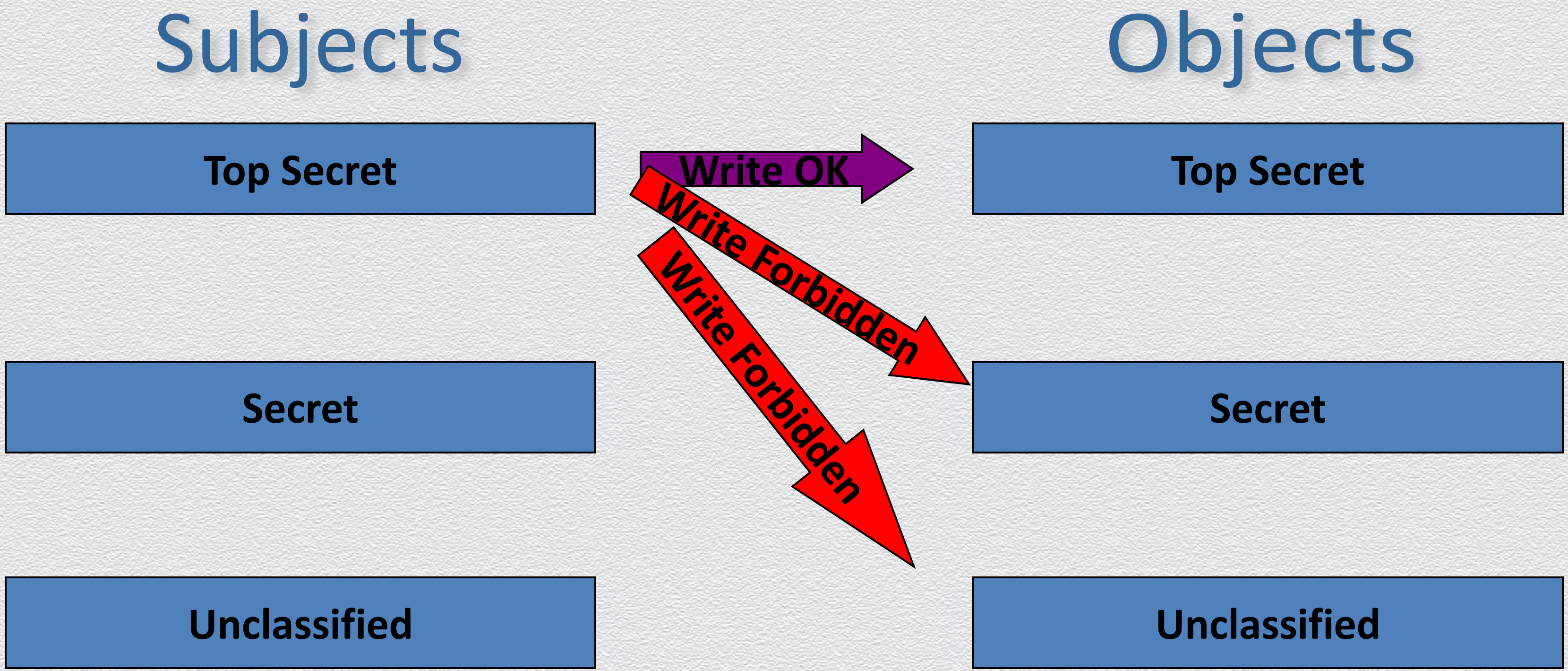
Access control: Bell-LaPadula



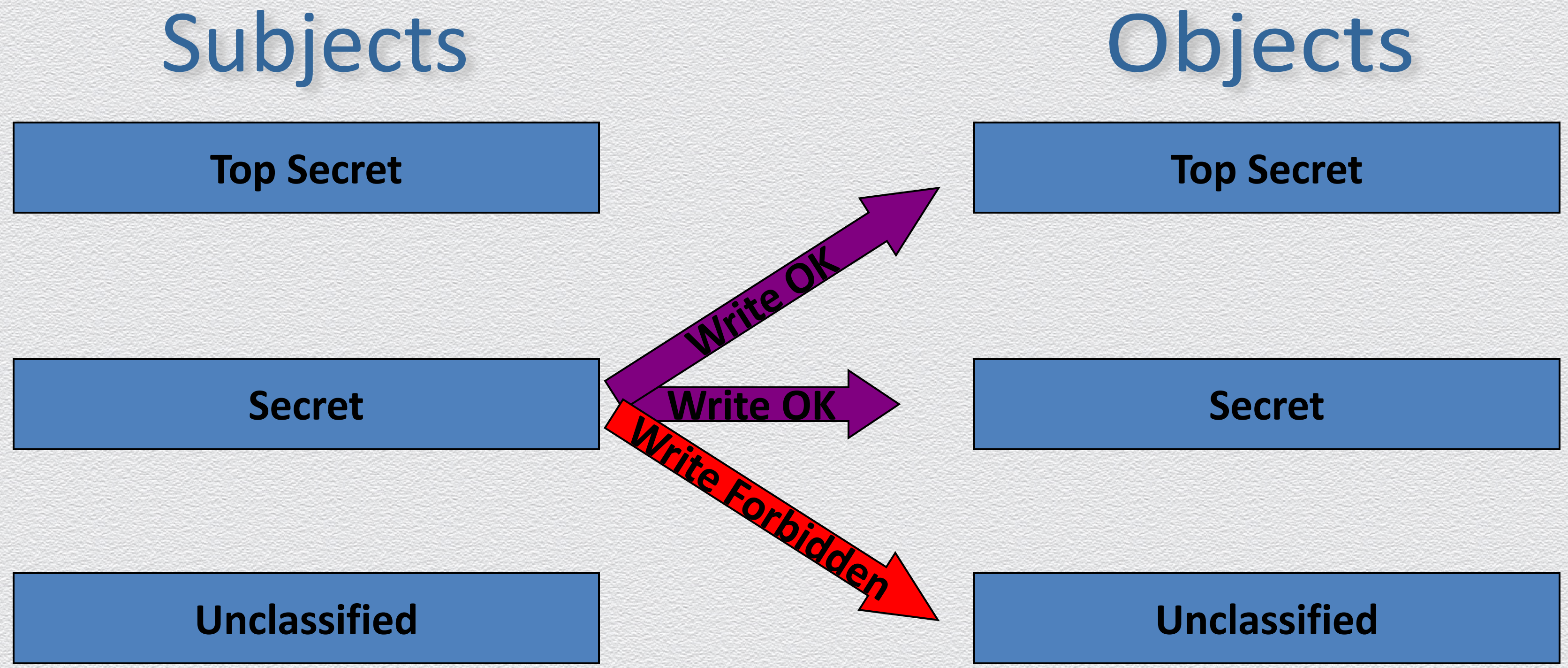
Bell - LaPadula (2)

- * property (star):
  the no write-down property
  - A subject s can write to object *p* if $C(s) \leq C(p)$

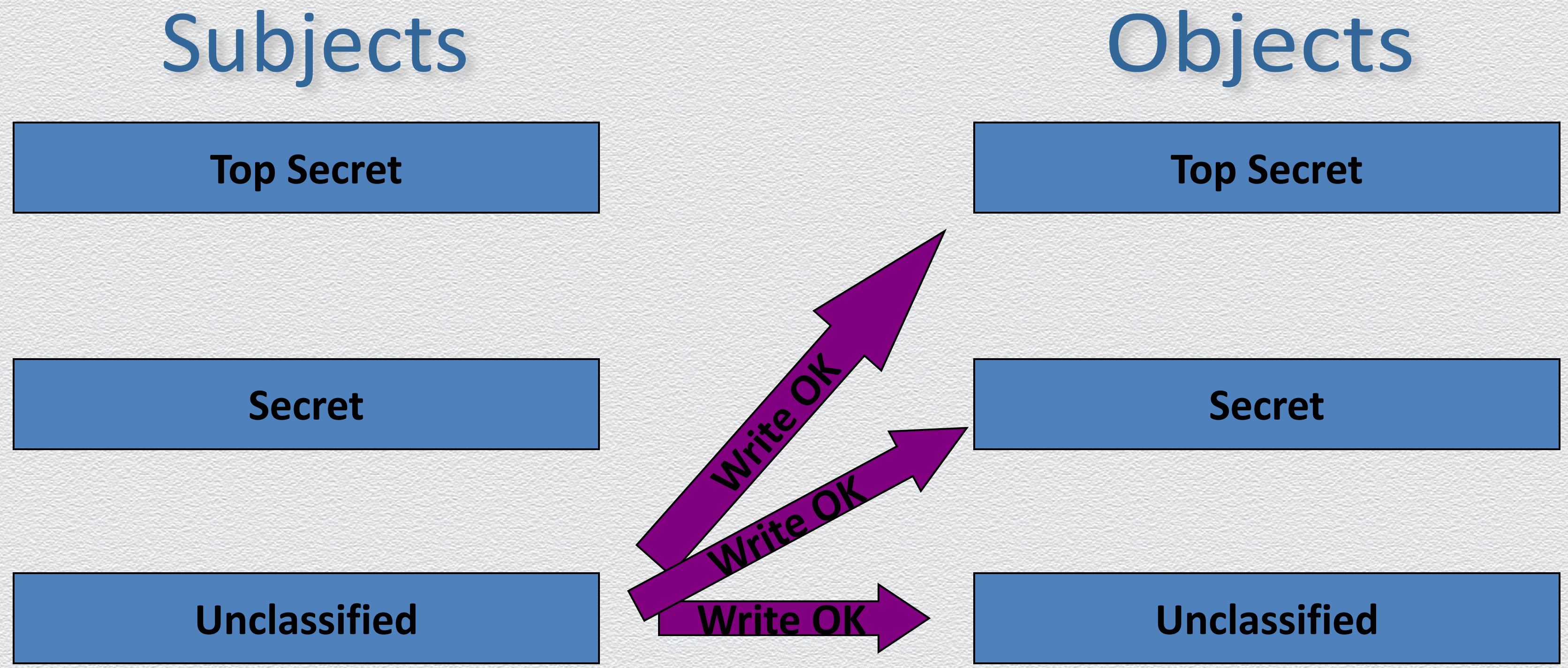
Access control: Bell-LaPadula



Access control: Bell-LaPadula



Access control: Bell-LaPadula



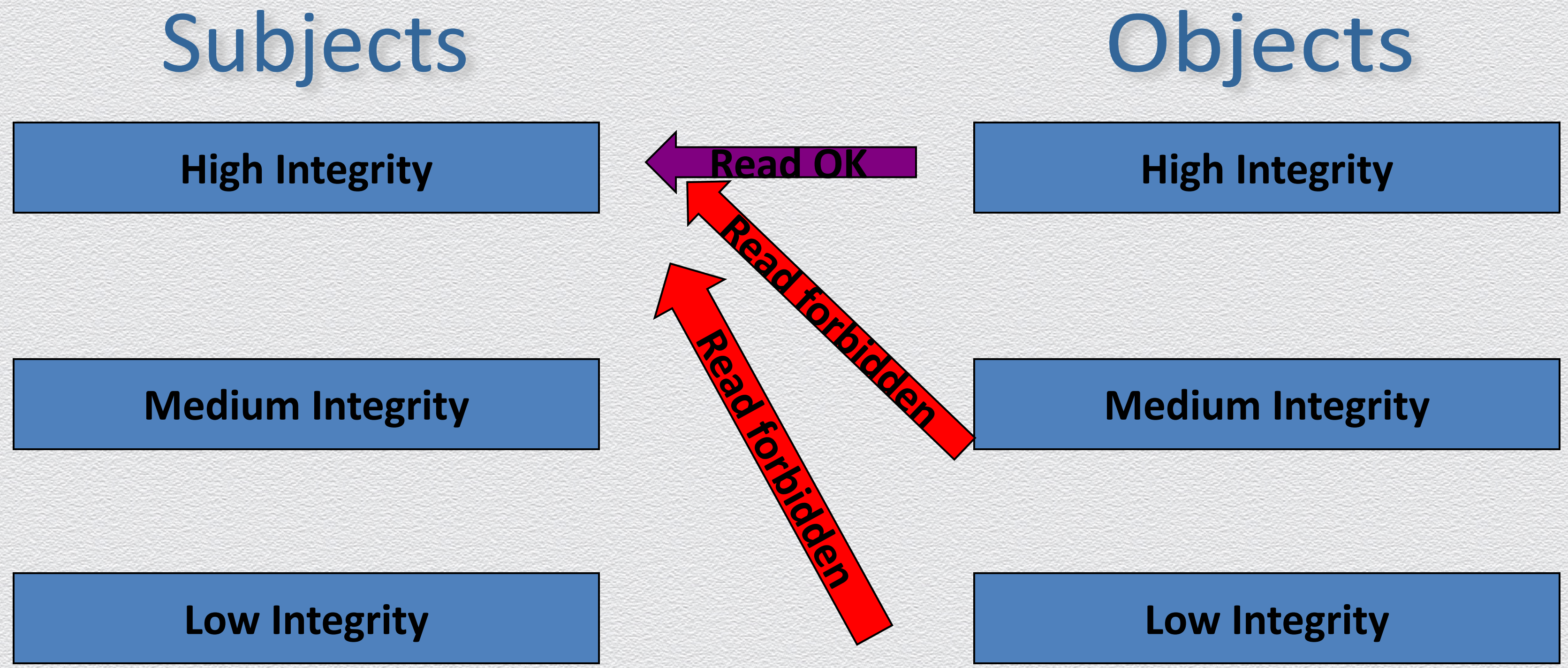
Security models - Biba

- Based on the Cold War experiences, information *integrity* is also important, and the Biba model, complementary to Bell-LaPadula, is based on the flow of information where preserving integrity is critical.
- The “dual” of Bell-LaPadula

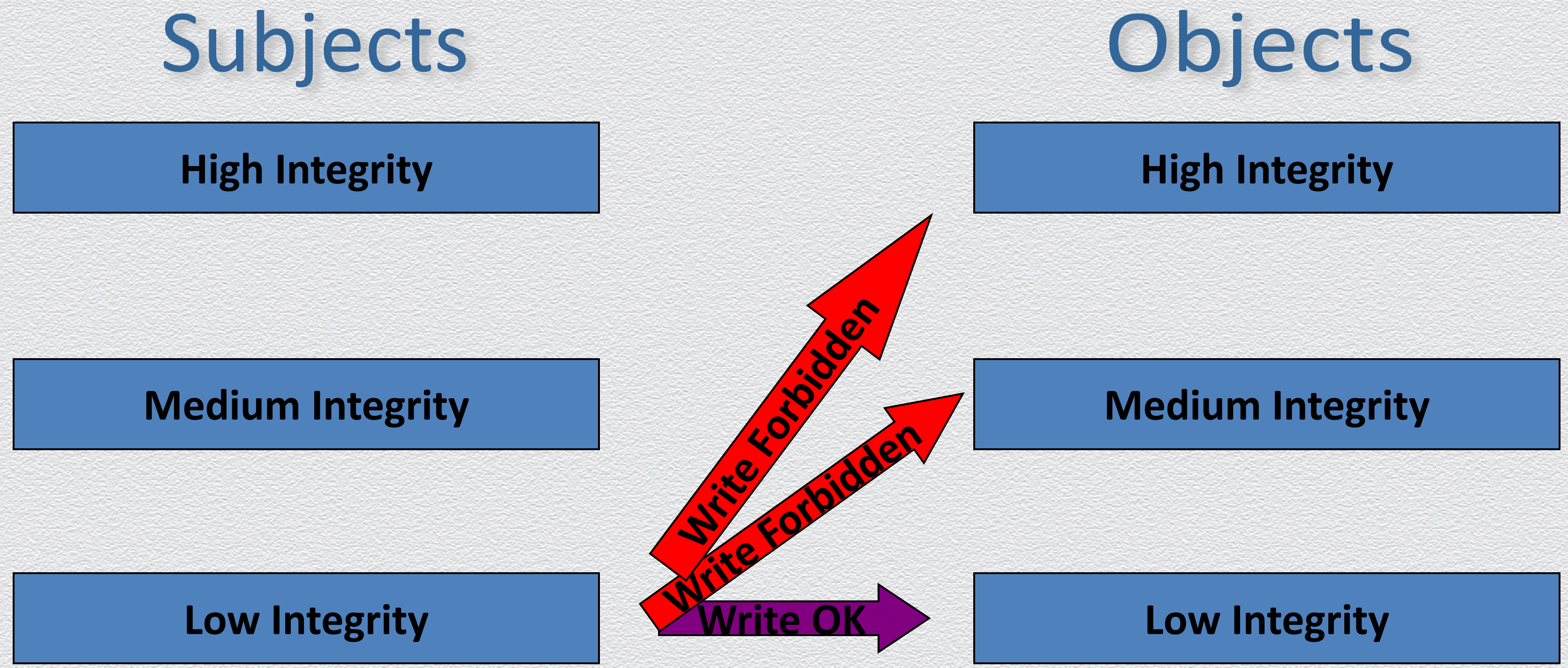
Integrity control: Biba

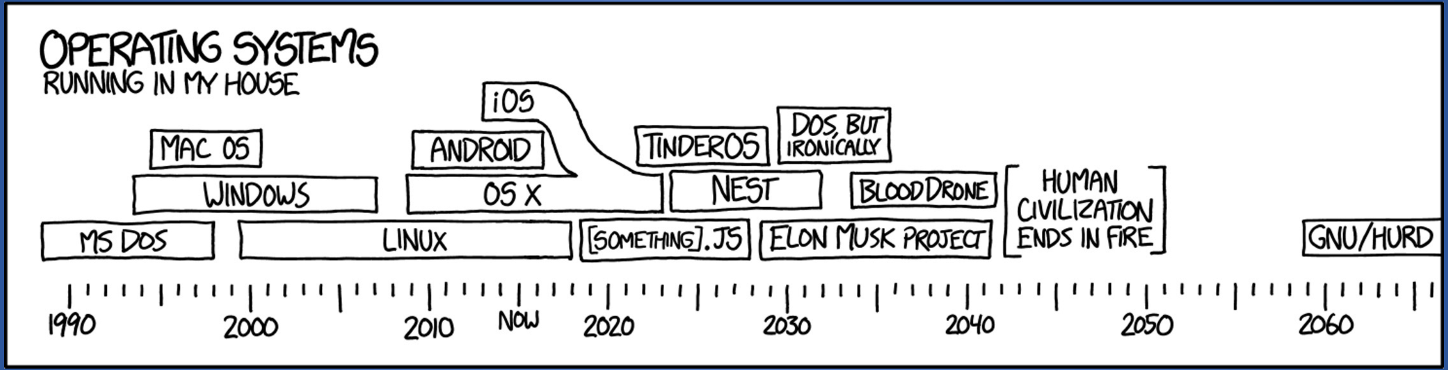
- Designed to preserve integrity, not limit access
- Three fundamental concepts:
  - Simple Integrity Property – no read down
  - Star Integrity Property (*) – no write up
  - No execute up

Integrity control: Biba



Integrity control: Biba



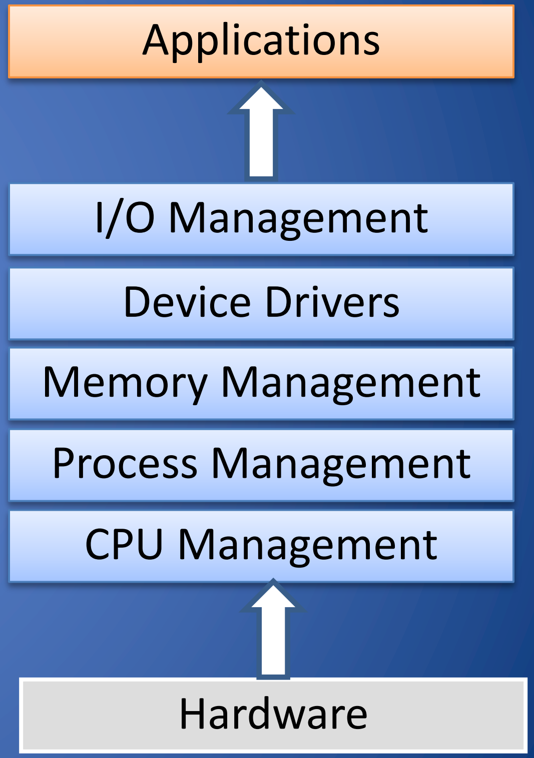

OPERATING SYSTEMS
RUNNING IN MY HOUSE
iOS
MAC OS
ANDROID
TINDEROS
DOS, BUT IRONICALLY
WINDOWS
OS X
NEST
BLOOD DRONE
HUMAN CIVILIZATION ENDS IN FIRE
MS DOS
LINUX
[SOMETHING].JS
ELON MUSK PROJECT
GNU/HURD
1990
2000
2010
NOW
2020
2030
2040
2050
2060

Source: XKCD

Operating System Layers

Many layers of abstraction:

- Kernel: core of the OS, controls hardware, resource access
  - Various subsystems (memory management, networking, storage, …)
- Execution modes:
  - user mode: access to resources mediated by the kernel
  - kernel mode: full and direct access to resources



Processes

The kernel manages applications as processes (or threads)

Every process has:

- Process ID (PID)
- Virtual memory
- Effective user

Processes

The kernel manages applications as processes (or threads)

Every process has:

- Process ID (PID)
- Virtual memory
- Effective user

Kernel provides

- Separate address space from other process
- Time/resource sharing
- Access control

Processes

```
emplisi@ubuntu:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─VGAuthService
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─anacron───sh───run-parts───mlocate───flock───updated
        ├─avahi-daemon───avahi-daemon
        ├─bluetoothd
        ├─boltd───2*[{boltd}]
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─firefox─┬─3*[Web Content───18*[{Web Content}]]
        │         ├─Web Content───19*[{Web Content}]
        │         ├─WebExtensions───18*[{WebExtensions}]
        │         ├─file:// Content───18*[{file:// Content}]
        │         └─59*[{firefox}]
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-ses
        │      │                 │                 │
```

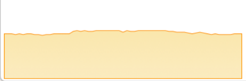
Activity Monitor
My Processes

CPU | Memory | Energy | Disk | Network

Search

Process Name	Mem...	Threads	Ports	PID	User
Firefox	10.09 GB	162	1,387	3561	deemer
Virtual Machine Service	7.92 GB	24	83	35580	deemer
FirefoxCP Isolated Web Content	5.19 GB	39	158	3569	deemer
Docker	4.10 GB	42	333	32597	deemer
Microsoft PowerPoint	3.59 GB	42	13,784	3516	deemer
FirefoxCP Isolated Web Content	1.91 GB	33	140	3568	deemer
FirefoxCP Isolated Web Content	1.30 GB	32	136	3567	deemer
FirefoxCP Isolated Web Content	1.30 GB	33	139	3566	deemer
FirefoxCP Isolated Web Content	830.4 MB	29	114	6432	deemer
Preview	773.0 MB	6	1,713	11577	deemer
FirefoxCP Isolated Web Content	659.7 MB	29	115	66660	deemer
Open and Save Panel Service (Preview)	526.2 MB	4	2,596	11578	deemer
Terminal	519.3 MB	7	459	3476	deemer
Finder	457.7 MB	8	1,456	3292	deemer
Discord Helper (Renderer)	420.0 MB	40	805	11208	deemer
Docker Desktop	414.3 MB	29	7,939	32617	deemer
FirefoxCP Isolated Web Content	407.9 MB	29	113	63607	deemer
FirefoxCP Isolated Web Content	405.5 MB	32	137	3576	deemer

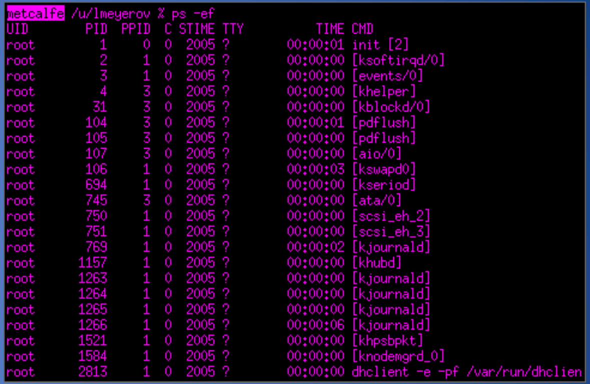
MEMORY PRESSURE

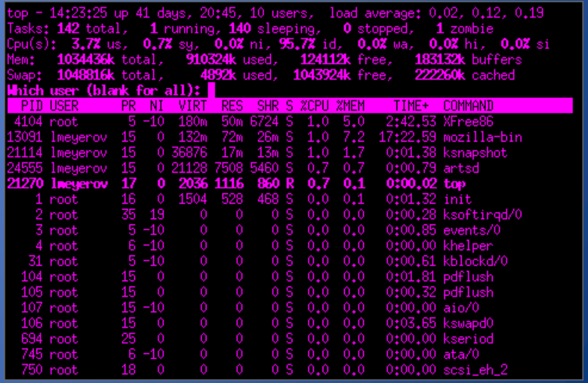


Physical Memory:	32.00 GB
Memory Used:	28.48 GB
Cached Files:	3.46 GB
Swap Used:	8.80 GB
App Memory:	9.95 GB
Wired Memory:	3.00 GB
Compressed:	14.92 GB

View Processes in Linux

- ps: displays snapshot of running processes
  - ps -ef : show all processes
  - ps -u <username>: show processes for a user
- top, htop: fancier list of processes
  - top -u <username>: filter by username
- kill <pid>: terminates a process





Process Management

- Each process has a context, which includes the user, parent process, and address space
- Kernel enforces policies to decide which resources each processes can use

System calls (syscalls)

- Primary way processes interact with kernel
- OS provides a “library” of syscalls for nearly all OS functions
  - Files: read, write, open, close, chmod, …
  - Process management: fork, clone, kill, …
  - Networking: socket, bind, connect

On syscall, process “yields” to kernel, executes in privileged kernel mode

System services (daemons)

- Background process that performs common tasks
- Started at boot time
- Could run with higher permissions than users

Typical services:

- Remote SSH connections
- Web servers
- Logging

Identification and Authentication (recap)

- A subject should provide a unique identifier
- Authentication is the act of confirming the truth of an attribute of a datum or entity
- There are three authentication factors:
  - Knowledge: Something you know
  - Ownership: Something you have
  - Inherence: Something you are

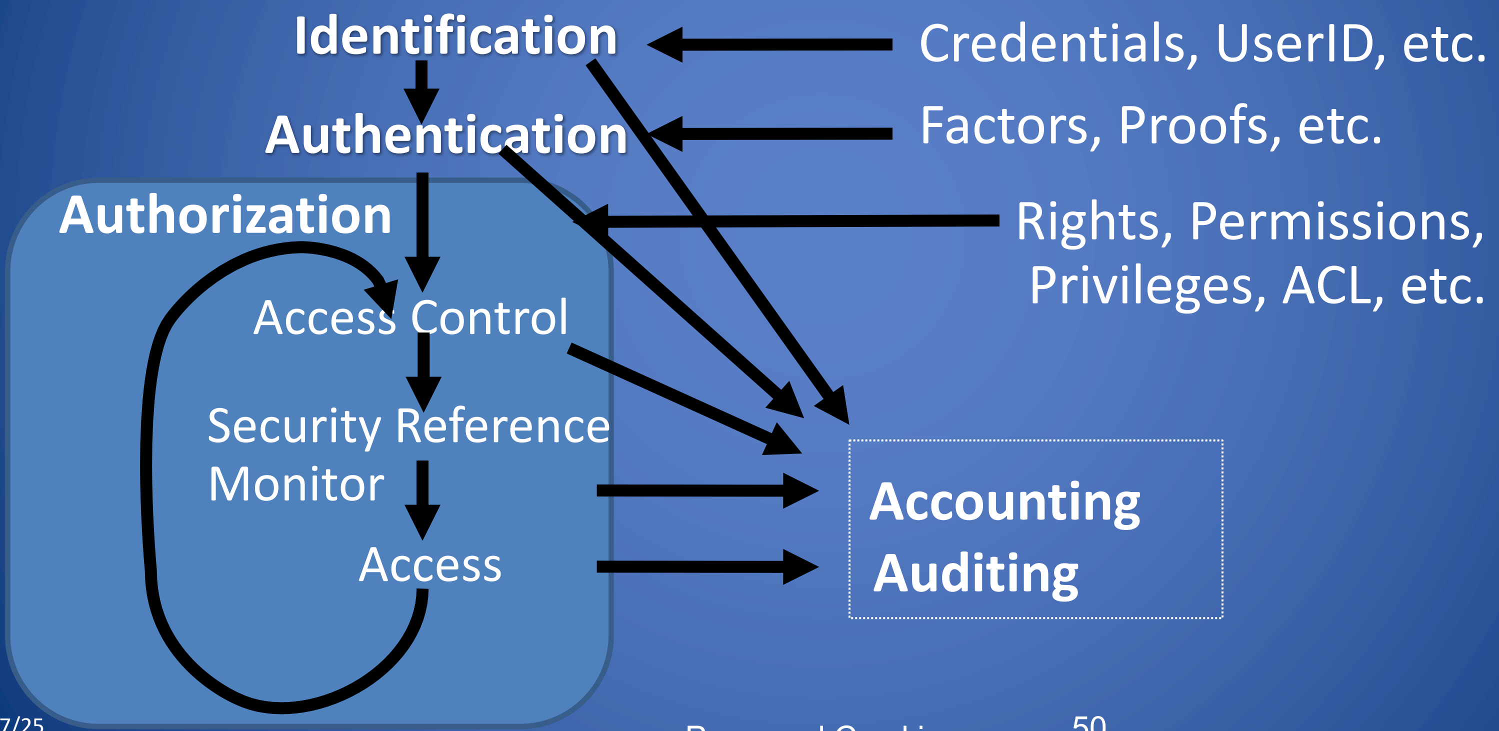
Authorization

- Once a subject is Authenticated, access should be authorized
- Authorization is the function of specifying access rights to resources (access control)
- More formally, "to authorize" is to define access policy: permissions, rights, etc.

AAA and more…

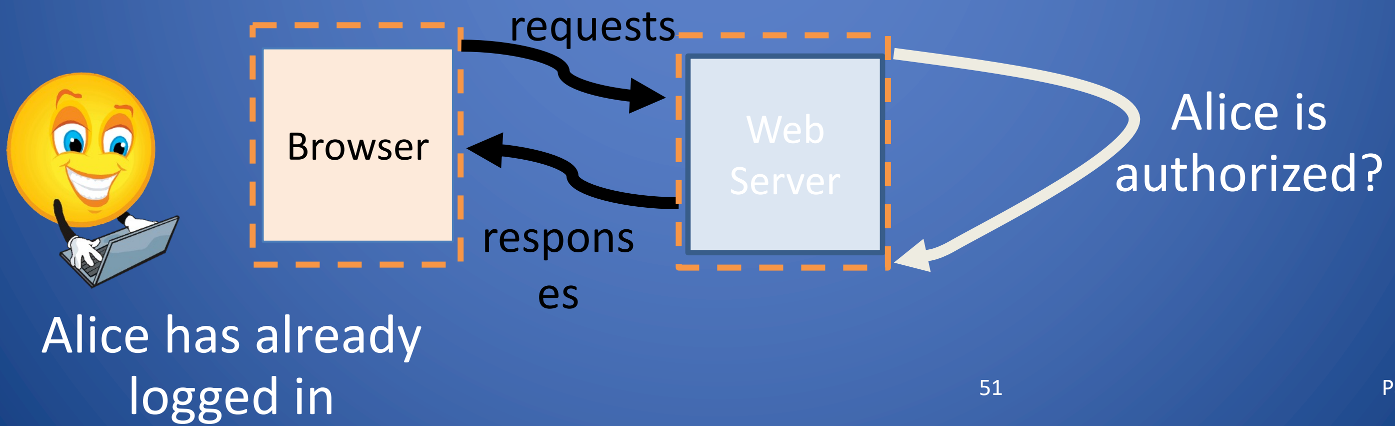
Identification, **Authentication**, **Authorization**, **Accounting**, Auditing

– AAA Working Group, IETF



Authorization on the web

- Alice logs in (i.e. authenticates)
- The web server is now aware of who is logged in
- Alice attempts to access a course
- The application checks to see if Alice has the authorization for the course…
  - If so, Alices receives the requested information
  - If not, Alice has a denied access response
- Authorization could be just for reading or writing or execute (more in the future lectures)



AAA: Authentication, Authorization, Accounting

Authorization: how to specify access rights to resources

- To authorize => to define access policy

Users

- Each process is associated with a user
- Specific users can have more privileges than regular users
  - Install or remove programs
  - Change rights of other users
  - Modify the configuration of the system
- Unix: root is a “super-user” with no restrictions

Users

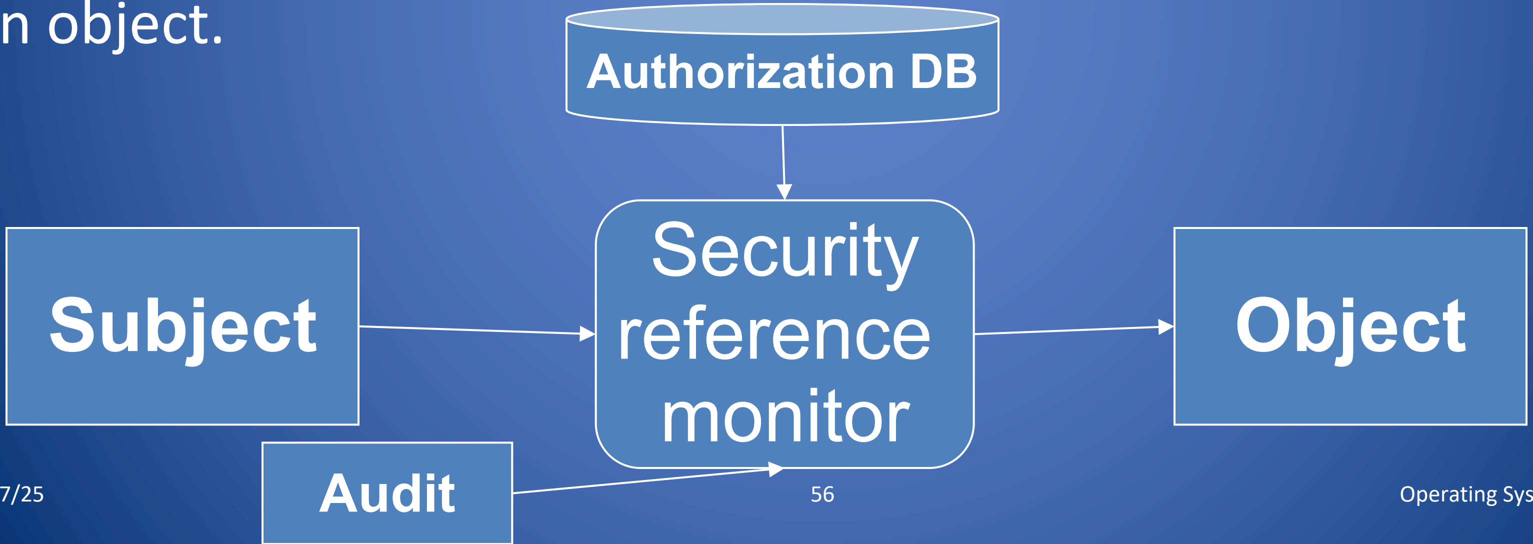
- Each process is associated with a user
- Specific users can have more privileges than regular users

- Unix: root is a "super-user" with no restrictions

How to we manage users?

Security Reference Monitor (SRM) (recap)

- Checks for proper authorization before granting access to objects.
- Object manager asks SRM if a Subject has the proper rights to execute a certain type of action on an Object.
- Implements auditing functions to keep track of attempts to access an object.



Discretionary Access Control (DAC)

- Users can protect what they own
  - The owner may grant access to others
  - The owner may define the type of access (read/write/execute) given to others
- DAC is the standard model used in operating systems
- Mandatory Access Control (MAC)
  - Multiple levels of security for users and documents (i.e. confidential, restricted, secret, top secret)
  - A user can create documents with just his level of security

General Principles

- Files and folders are managed by the operating system
- Applications, including shells, access files through an API
- Access control entry (ACE)
  - Allow/deny a certain type of access to a file/folder by user/group
- Access control list (ACL)
  - Collection of ACEs for a file/folder
- A file handle provides an opaque identifier for a file/folder
- File operations
  - Open file: returns file handle
  - Read/write/execute file
  - Close file: invalidates file handle
- Hierarchical file organization
  - Tree (Windows)
  - DAG (Linux)

Access Control Entries and Lists

- An Access Control List (ACL) for a resource (e.g., a file or folder) is a sorted list of zero or more Access Control Entries (ACEs)
- An ACE refers specifies that a certain set of accesses (e.g., read, execute and write) to the resources is allowed or denied for a user or group
- Examples of ACEs for folder “Bob’s CS166 Grades”
  - Bob; Read; Allow
  - TAs; Read; Allow
  - TWD; Read, Write; Allow
  - Bob; Write; Deny
  - TAs; Write; Allow

Closed vs. Open Policy

Closed policy

- Also called "default secure"

- Give Tom read access to "foo"
- Give Bob r/w access to "bar
- Tom: I would like to read "foo"
  - Access allowed
- Tom: I would like to read "bar"
  - Access denied

Open Policy

- Deny Tom read access to "foo"
- Deny Bob r/w access to "bar"
- Tom: I would like to read "foo"
  - Access denied
- Tom: I would like to read "bar"
  - Access allowed

Question (1)

An ACL with no entries on a file?

A. Access Allowed to all with Open Policy
   Access Allowed to all with Closed Policy
B. Access Denied to all with Open Policy
   Access Allowed to all with Closed Policy
C. Access Allowed to all with Open Policy
   Access Denied to all with Closed Policy
D. Access Denied to all Open Policy
   Access Denied to all Closed Policy
E. It is not possible to realize

Question (1) - Answer

An ACL with no entries on a file?

A. Access Allowed to all with Open Policy
   Access Allowed to all with Closed Policy
B. Access Denied to all with Open Policy
   Access Allowed to all with Closed Policy
C. Access Allowed to all with Open Policy
   Access Denied to all with Closed Policy
D. Access Denied to all Open Policy
   Access Denied to all Closed Policy
E. It is not possible to realize

Closed Policy with Negative Authorizations and Deny Priority

- Give Tom r/w access to “bar”
- Deny Tom write access to “bar”
- Tom: I would like to read “bar”
  - Access allowed
- Tom: I would like to write “bar”
  - Access denied
- Policy is used by Windows to manage access control to the file system

Role-Based Access Control

- Within an organization roles are created for various job functions
- The permissions to perform certain operations are assigned to specific roles
- Users are assigned particular role, with which they acquire the computer authorizations
- Users are not assigned permissions directly, but only acquire them through their role

U.S. Navy image in the public domain.

Access Control: File System

Linux vs. Windows

- Linux
  - Allow-only ACEs
  - Access to file depends on ACL of file and of all its ancestor folders
  - Start at root of file system
  - Traverse path of folders
  - Each folder must have execute (cd) permission
  - Different paths to same file not equivalent
  - File's ACL must allow requested access

- Windows
  - Allow and deny ACEs
  - By default, deny ACEs precede allow ones
  - Access to file depends only on file's ACL
  - ACLs of ancestors ignored when access is requested
  - Permissions set on a folder usually propagated to descendants (inheritance)
  - System keeps track of inherited ACE's

Linux File Access Control

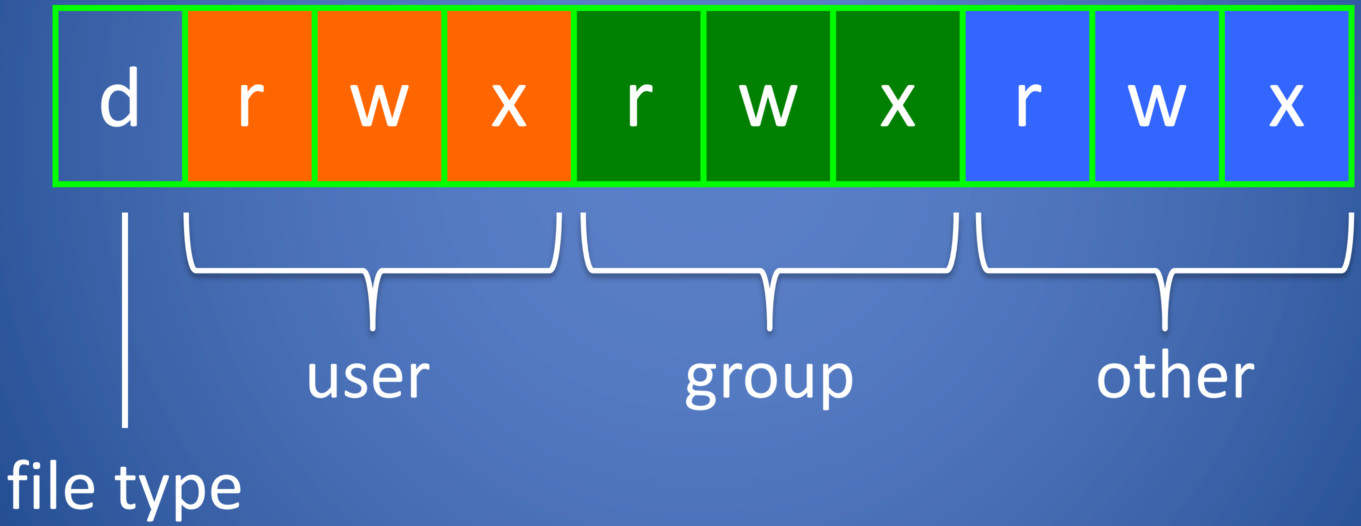
- File Access Control for:
  - Files
  - Directories
  - Therefore…
    - \dev\ : *devices*
    - \mnt\ : *mounted file systems*
    - What else? *Sockets, pipes, symbolic links…*

Unix Permissions

- Standard for all UNIXes
- Every file is owned by a user and has an associated group
- Permissions often displayed in compact 10-character notation
- To see permissions, use `ls -l`

```
jk@sphere:~/test$ ls -l
total 0
-rw-r-----  1 jk ugrad 0 2005-10-13 07:18 file1
-rwxrwxrwx  1 jk ugrad 0 2005-10-13 07:18 file2
```

Unix File Types and Basic Permissions



Permissions Examples (Regular Files)

-rw-r—r--	read/write for owner, read-only for everyone else
-rw-r-----	read/write for owner, read-only for group, forbidden to others
-rwx------	read/write/execute for owner, forbidden to everyone else
-r--r--r--	read-only to everyone, including owner
-rwxrwxrwx	read/write/execute to everyone

Permissions for Directories

- Permissions bits interpreted differently for directories
- *Read* bit allows listing names of files in directory, but not their properties like size and permissions
- *Write* bit allows creating and deleting files within the directory
- *Execute* bit allows entering the directory and getting properties of files in the directory
- Lines for directories in `ls -l` output begin with d, as below:

```
jk@sphere:~/test$ ls -l
Total 4
drwxr-xr-x  2 jk ugrad 4096 2005-10-13 07:37 dir1
-rw-r--r--  1 jk ugrad    0 2005-10-13 07:18 file1
```

Permissions Examples (Directories)

drwxr-xr-x	all can enter and list the directory, only owner can add/delete files
drwxrwx---	full access to owner and group, forbidden to others
drwx--x---	full access to owner, group can access known filenames in directory, forbidden to others
-rwxrwxrwx	full access to everyone