THE UNIVERSITY of NEW ORLEANS

CSCI 4621/5621 Intro to CyberSecurity

02: OS Security & Access Control

Vassil Roussev

vassil@cs.uno.edu

READING: Oorschot [ch5]

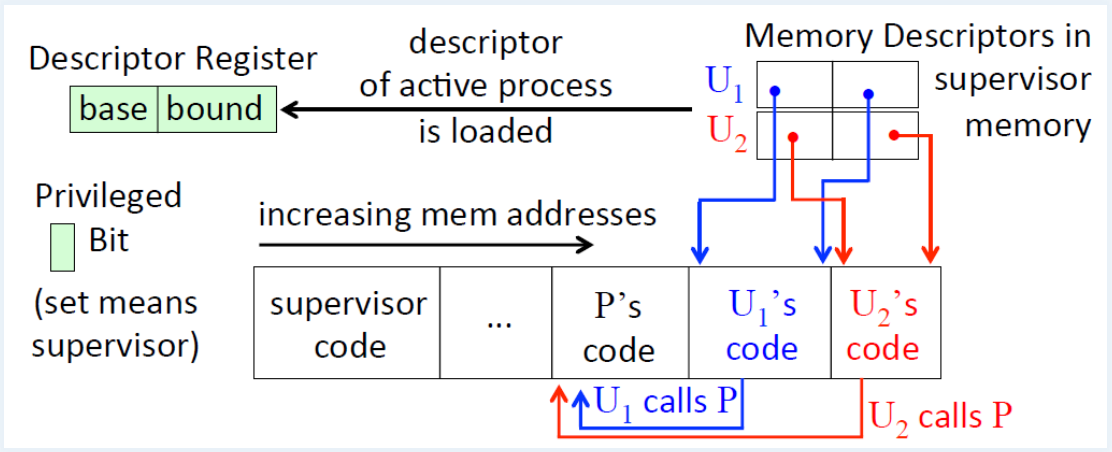
One-slide history

- 1950s: batch-processing systems → one task/user at a time
  - *simple but inconvenient for users/programmers;*
- 1960s: first multi-user time-sharing systems
  - *bring up the need for mechanisms for isolating users from each other*
  - *… and the OS kernel*
- In the early days,

  protection == control to memory access
  - *it is still a fundamental part of access control*
- Most modern ideas go back to 1965-75
  - *especially,* ***Multics*** *and* ***Unix***

MEMORY PROTECTION

ISOLATION: NEED & IMPLEMENTATION



[CREDIT: Oorschot]

- Multiple concurrent tasks
  - » *need to be isolated from each other to avoid interference*
    - both malicious and unintended
  - » *need to be isolated from the kernel*
    - to prevent whole system failures
  - » *also, need to provide legal means for inter-process communication (IPC)*
    - mediated by the *supervisor* (OS kernel)
- Early implementation
  - » *descriptor register <base, bound> + privileged bit + supervisor*

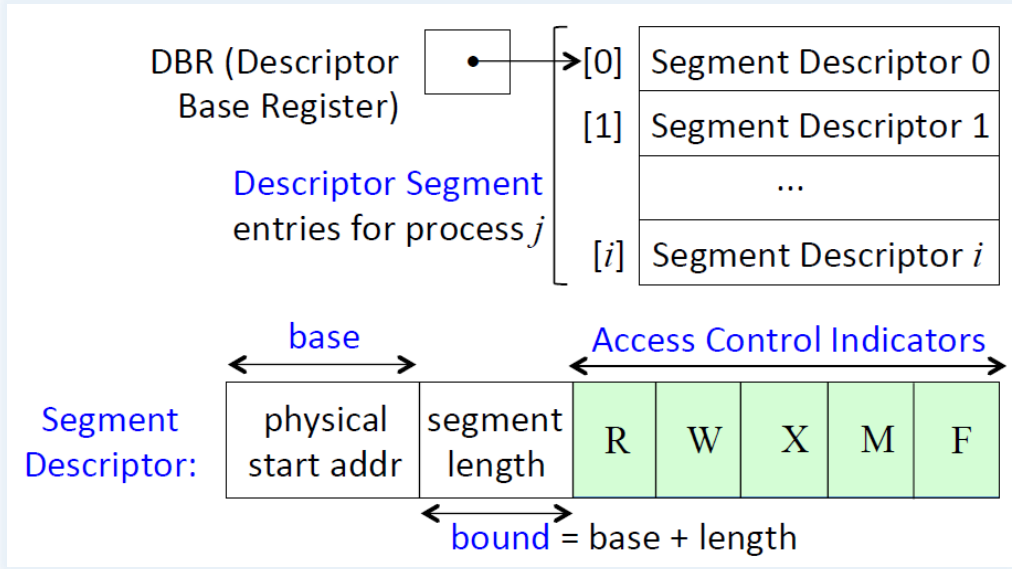
ISOLATION (2)

- Problem: descriptor register approach allows only

  all-or-nothing access

- Solution: segment addressing (Multics)
  - » *address space split into segments*
    - per-process segment descriptor table
  - » *allows finer grain memory access control*
    - per-segment

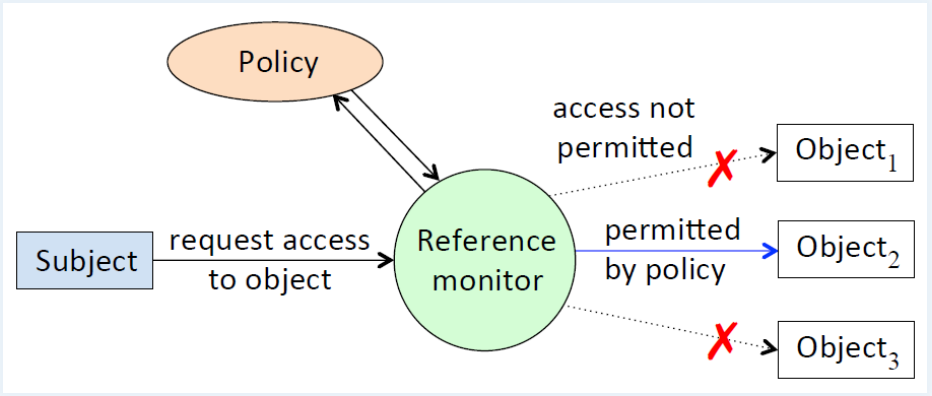
      Read / **W**rite / e**X**ecute / **M**ode / **F**ault access
  - » *allows for sharing of segments b/w processes*



[CREDIT: Oorschot]

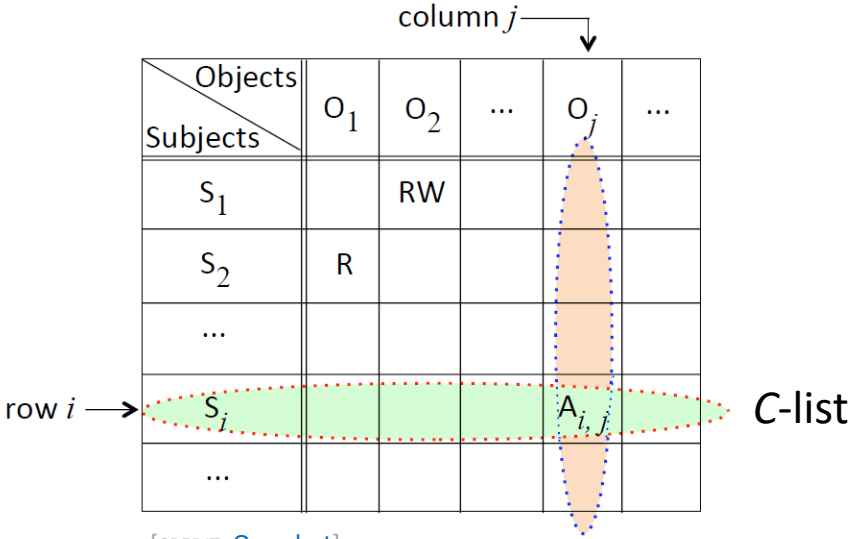
REFERENCE MONITOR

REFERENCE MONITOR



[CREDIT: Oorschot]

- Concept
  - *all references **by** any program **to** any program, data or device*
  - *are **validated** against a list of*
  - ***authorized** types of reference based on user and/or program function*
- Access matrix
  - *subject (principal)*
  - *object*
  - *access attributes (permission)*



[CREDIT: Oorschot]

ACL

Reference monitor implementation requirements

- Tamper-proof
- Always invoked
  - *complete mediation/cannot be circumvented*
- Verifiable
  - *must be small enough to be (formally) verifiable*
- Known as a *security kernel*
- Very difficult in practice to accomplish
  - *but quite influential as a design*

Reference monitor implementation [1]

- Dependencies
  - *authentication system*
  - *correct hardware*
  - *trustworthy software*
  - *physical system security*
  - *user I/O security*
  - *…*
- Access matrix is conceptual
  - *(it would be huge, sparse and impractical)*
  - *actual implementations must take efficiency into account*
    - e.g., Unix FS: rwx rwx rwx model (often insufficiently expressive)

Reference monitor implementation [2]

- Capability- vs. ID-based systems
- Capabilities
  - *access token (bearer token)*
    - correct token provides access regardless of identity
- ID
  - *identity check performed before access*
    - authorization list maintained on a per-object basis
- Audit trails
  - *complete mediation provides for detailed logs*
    - must be secured/archives
    - there is a performance cost for very detailed logs

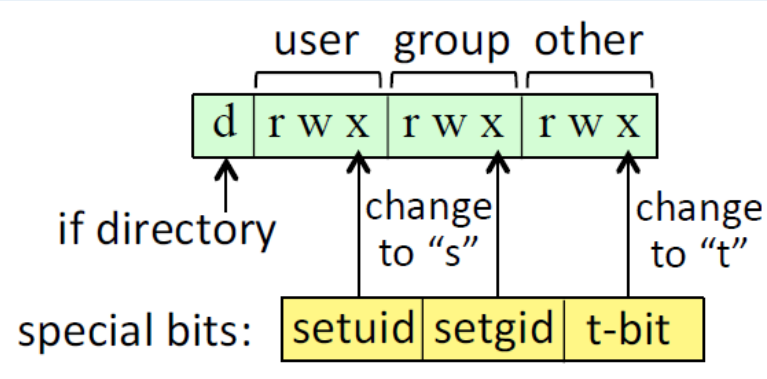
OBJECT PERMISSION & FILE-BASED AC

UNIX: EVERYTHING IS A FILE

- Special files represent various system resources
  - » *processes, network connections, printers, etc.*
- Conceptually, ACLs are very expressive
  - » *but can present efficiency problems*
- Unix model

  owner:group:others → **R**ead, **W**rite, e**X**ecute bit for each
- Superuser → UID=0
- root → byconvention, UID=0
- umask → default permissions

UNIX FILE PERMISSIONS



[CREDIT: Oorschot]

Binary (12 bits)	Octal	Symbolic	Meaning
000 100 000 000	0400	- r-- --- ---	user (owner) has R
000 010 000 000	0200	- -w- --- ---	user (owner) has W
000 001 000 000	0100	- --x --- ---	user (owner) has X
000 000 110 000	0060	- --- rw- ---	group has R, W
000 000 101 000	0050	- --- r-x ---	group has R, X
000 000 011 000	0030	d --- -wx ---	group has W, X; file is a directory file
000 000 000 111	0007	- --- --- rwx	other has R, W, X
000 110 100 100	0644	- rw- r-- r--	user has R, W; group and other have R

SETUID BIT & EUID

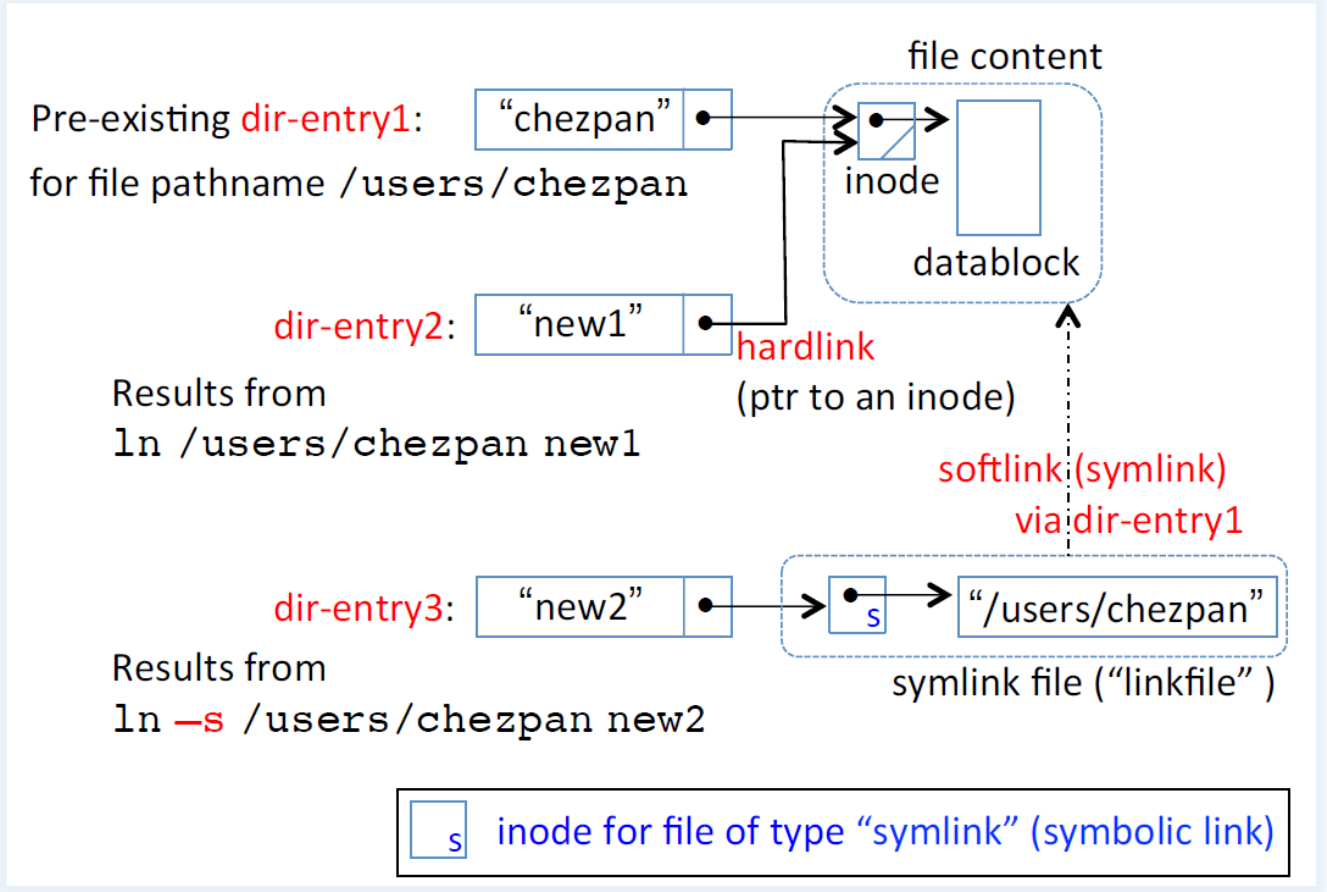
- Setuid
  - » *can be set for any binary executable by the owner*
  - » *effect: any invoking process runs on behalf of the owner*
    - potentially granting additional access, not normally available
- OS tracks
  - » *rUID → process owner*
  - » *eUID → effective UID*
  - » *sUID → saved UID*
  - » *to facilitate switching privilege levels*
- rGID, eGID, sGID → for groups
- Inherited userid → fork()

Directory permissions

- Tree structure, "/" is the root
- Permission
  - *R → allows listing of content (filenames & attributes)*
  - *W → allows file creation, renaming/deleting files (X req)*
  - *X → allows traversal; absence denies file content access*
  - *setuid → no meaning*
  - *setgid → files created inherit GID of directory creator (not invoking process)*
  - *t bit → text, or sticky bit → prevents modification of files created by other users*
    - e.g., /tmp

LINKS (TURN FS TREE INTO A DAG)

- Symbolic link
  - » *a text file containing the name of the linked-to file*
  - » *deletion does not affect the original file*
- Hard link
  - » *creates a file entry, which points to the same metadata structure*
  - » *deletion (unlink) reduces reference count*
    - if count==0 then file content is deleted



[CREDIT: Oorschot]

MANDATORY AC & RBAC

- Mandatory vs. discretionary
  - *discretionary (D-AC) → resource owner decides permissions*
    - e.g., Unix
  - *mandatory (M-AC) → access defined by policy*
    - e.g., MLS model (DoD)
- Role based AC (RBAC)
  - *users are assigned roles on a per session basis*
  - *each role has policy determined permissions*
  - *it usually maps well to organizational structure*