

Computer Security

CS 426

Lecture 10



Discretionary Access Control

Review

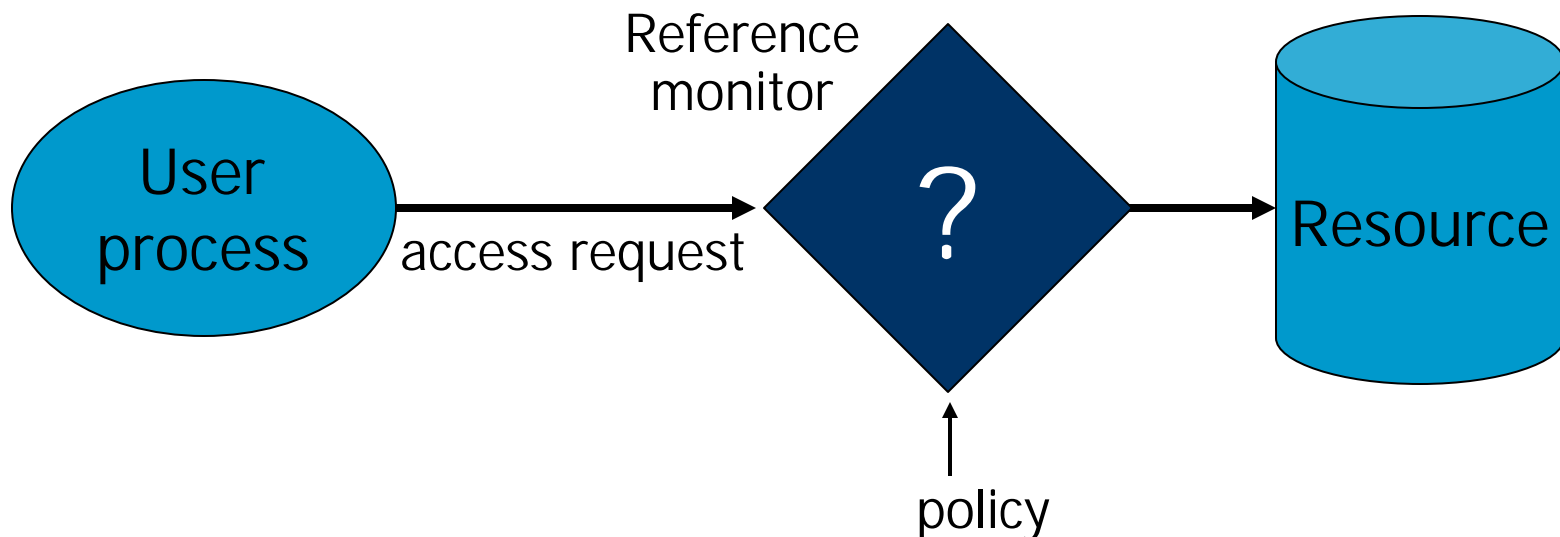
- Approaches to fix/enhance standard OS access control
- Virtualization to achieve confinement
 - operating-system level virtualization, e.g., chroot, jail, etc.
 - virtual machines by hardware emulation, e.g., VMWare
 - paravirtualization, e.g., Xen
- Limiting the power of the all-powerful root
 - Linux/Posix capability model
 - Securelevel

Roadmap

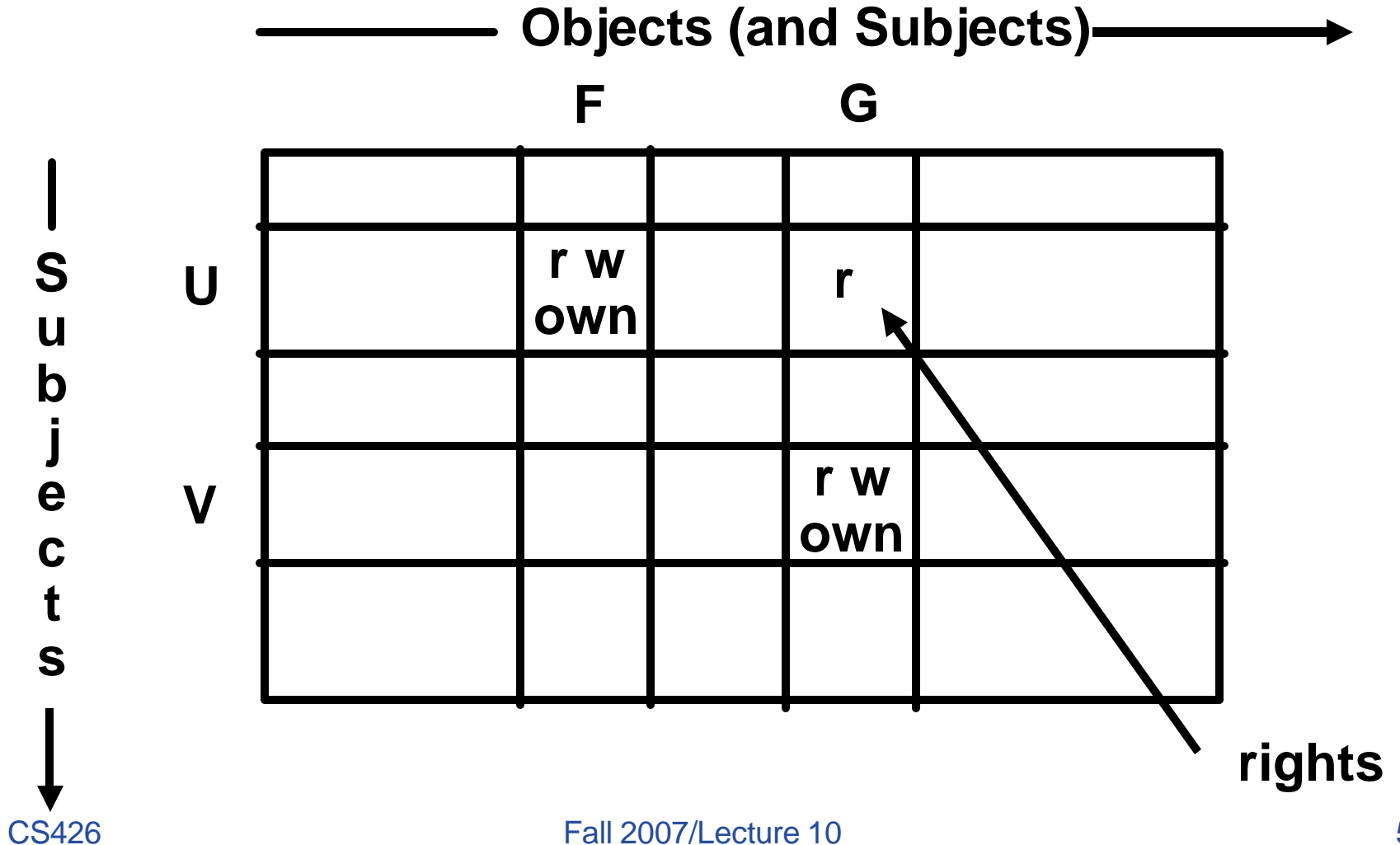
- The Access Matrix Mode
- Access Control List vs. Capabilities
- Discretionary Access Control

Access control

- Reference monitor mediate all access to resources
 - **complete mediation**: control **all** accesses to resources



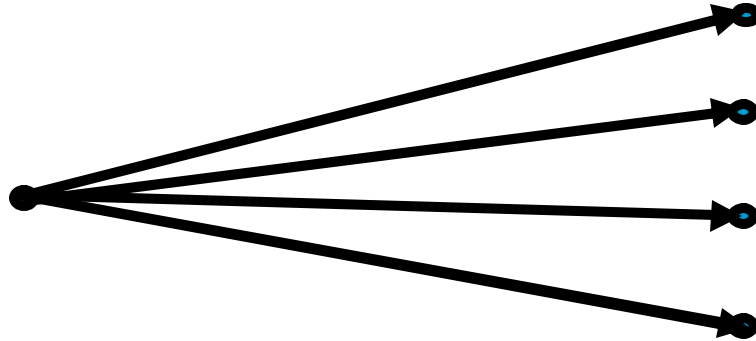
ACCESS MATRIX MODEL



ACCESS MATRIX MODEL

- Basic Abstractions
 - Subjects
 - Objects
 - Rights
- The rights in a cell specify the access of the subject (row) to the object (column)

USERS AND PRINCIPALS



USERS

Real World User

PRINCIPALS

**Unit of Access Control
and Authorization**

the system authenticates the user in context
of a particular principal

USERS AND PRINCIPALS

- There should be a one-to-many mapping from users to principals
 - a user may have many principals, but
 - each principal is associated with an unique user
- This ensures accountability of a user's actions

What are principals in UNIX?

What does the above imply in UNIX?

sudo: A Unix command enabling accounting for root actions

- Sudo (superuser do) intends to replace su; it allows certain users (or groups of users) to run some (or all) commands as root while logging all commands and arguments.

Sample /etc/sudoers file

User privilege specification

```
root    ALL=(ALL) ALL
```

Uncomment to allow people in wheel to run all commands

```
# %wheel    ALL=(ALL)  ALL
```

Same thing without a password

```
# %wheel    ALL=(ALL)  NOPASSWD: ALL
```

Samples

```
# %users    ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
```

```
# %users    localhost=/sbin/shutdown -h now
```

PRINCIPALS AND SUBJECTS

- A subject is a program (application) executing on behalf of a principal
- A principal may at any time be idle, or have one or more subjects executing on its behalf

What are subjects in UNIX?

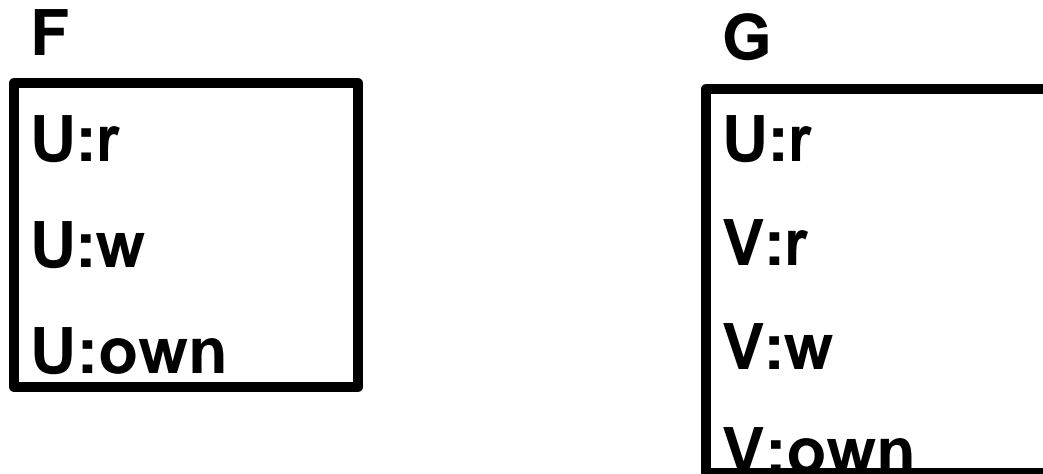
OBJECTS

- An object is anything on which a subject can perform operations (mediated by rights)
- Usually objects are passive, for example:
 - File
 - Directory (or Folder)
 - Memory segment
- But, subjects can also be objects, with operations
 - kill
 - suspend
 - resume

IMPLEMENTATION OF AN ACCESS MATRIX

- Access Control Lists
- Capabilities
- Relations

ACCESS CONTROL LISTS (ACLs)



each column of the access matrix is stored with the object corresponding to that column

Are UNIX permission bits ACL?

CAPABILITY LISTS

U **F/r, F/w, F/own, G/r**

V **G/r, G/w, G/own**

each row of the access matrix is stored with the subject corresponding to that row

ACCESS CONTROL TRIPLES

Subject	Access	Object
U	r	F
U	w	F
U	own	F
U	r	G
V	r	G
V	w	G
V	own	G

commonly used in relational DBMS

Roadmap

- The Access Matrix Mode
- Access Control List vs. Capabilities
- Discretionary Access Control

ACL vs. Capabilities

The Confused Deputy Problem

- The compiler program is SYSX/FORT.
- Other files under SYSX include STAT and BILL.
- The compiler program needs to write to files in SYSX directory, so it is given authority to write to files in SYSX.
- A user who runs SYSX/FORT can provide a file name to receive output info.
- A malicious user may use SYSX/BILL as the output name, resulting in billing info being erased.

Analysis of The Confused Deputy Problem

- The compiler runs with authority from two sources
 - the invoker
 - the system admin (who installed the compiler and controls billing and other info)
- It is the deputy of two masters
- There is no way to tell which master the deputy is serving when accessing a piece of resource

How the Capability Approach Solves the Confused Deputy Problem

- The compiler program is given capabilities to access SYSX/STAT and SYSX/BILL, which are stored in capability slots 1 & 2
- When the invoker runs the compiler program, it gives a capability to write to the output file, which is stored in capability slot 3. The invoker cannot give a capability for SYSX/BILL if it doesn't have the capability.
- When writing billing info, the program uses capability in slot 2. When writing the output, it uses capability in slot 3.

Capability vs. ACL: Naming

- ACL systems need a namespace for objects
- In capability systems, a capability can serve both to designate a resource and to provide authority.
- ACLs also need a namespace for subjects
 - as they need to refer to subjects
- Implications
 - the set of subjects cannot be too many or too dynamic
 - most ACL systems treat users as subjects, and do not support fine-grained subjects

Capability vs. ACL: Authority Management

- Subject-Aggregated Authority Management
- In (almost) all ACL systems, the power to edit authorities is aggregated by resource
 - naturally compatible with Discretionary Access Control, where there is often the notion of an owner
- In capabilities systems, the power to edit authorities is aggregated by subject

Capabilities vs. ACL: Ambient Authority

- Ambient authority means that a user's authority is automatically exercised, without the need of being selected.
 - causes the confused deputy problem
- No Ambient Authority in capability systems

ACL'S VS CAPABILITIES

- ACL's require authentication of subjects
- Capabilities do not require authentication of subjects, but do require unforgeability and control of propagation of capabilities

ACL'S VS CAPABILITIES

ACCESS REVIEW

- ACL's provide for superior access review on a per-object basis
- Capabilities provide for superior access review on a per-subject basis

REVOCACTION

- ACL's provide for superior revocation facilities on a per-object basis
- Capabilities provide for superior revocation facilities on a per-subject basis

ACL'S VS CAPABILITIES

LEAST PRIVILEGE

- Capabilities provide for finer grained least privilege control with respect to subjects, especially dynamic short-lived subjects created for specific tasks

Conjectures on Why Most Real-world OS Use ACL, rather than Capabilities

- Capability is more suitable for process level sharing, but not user-level sharing
 - user-level sharing is what is really needed
- Processes are more tightly coupled in capability-based systems because the need to pass capabilities around
 - programming may be more difficult

Roadmap

- The Access Matrix Mode
- Access Control List vs. Capabilities
- Discretionary Access Control

Discretionary Access Control

- No precise definition. Basically, DAC allows access rights to be propagated at subject's discretion
 - often has the notion of owner of an object
 - used in UNIX, Windows, etc.
- *"A means of restricting access to objects based on the identity and need-to-know of users and/or groups to which the object belongs. Controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (directly or indirectly) to any other subject."*

Mandatory Access Control

- Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
 - denying users full control over the access to resources that they create. The system security policy (as set by the administrator) entirely determines the access rights granted

Multi-level Security (MLS)

- The capability of a computer system to carry information with different sensitivities (i.e. classified information at different security levels), permit simultaneous access by users with different security clearances and needs-to-know, and prevent users from obtaining access to information for which they lack authorization.
- Typically use MAC
- Primary Security Goal: Confidentiality

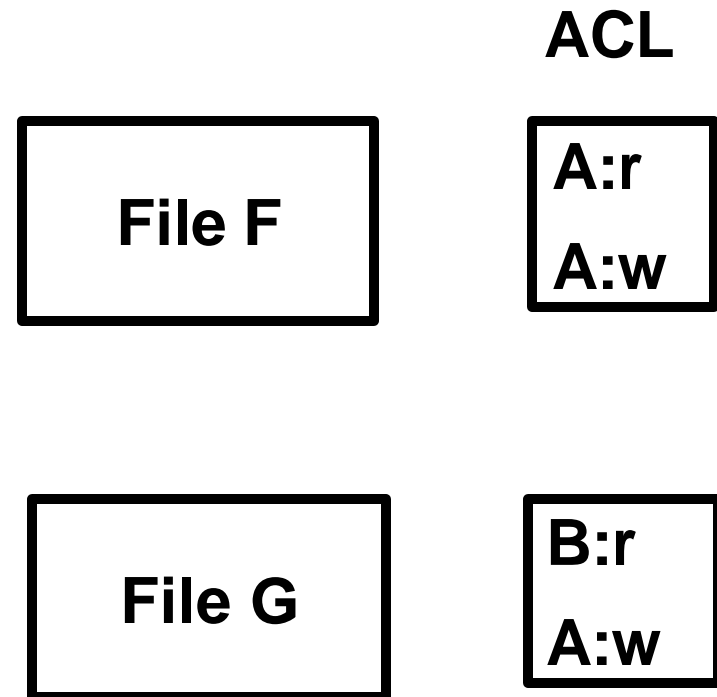
INHERENT WEAKNESS OF DAC

- Unrestricted DAC allows information from an object which can be read to any other object which can be written by a subject
 - do not provide multi-level security
- Suppose our users are trusted not to do this deliberately. It is still possible for Trojan Horses to copy information from one object to another.

TROJAN HORSES

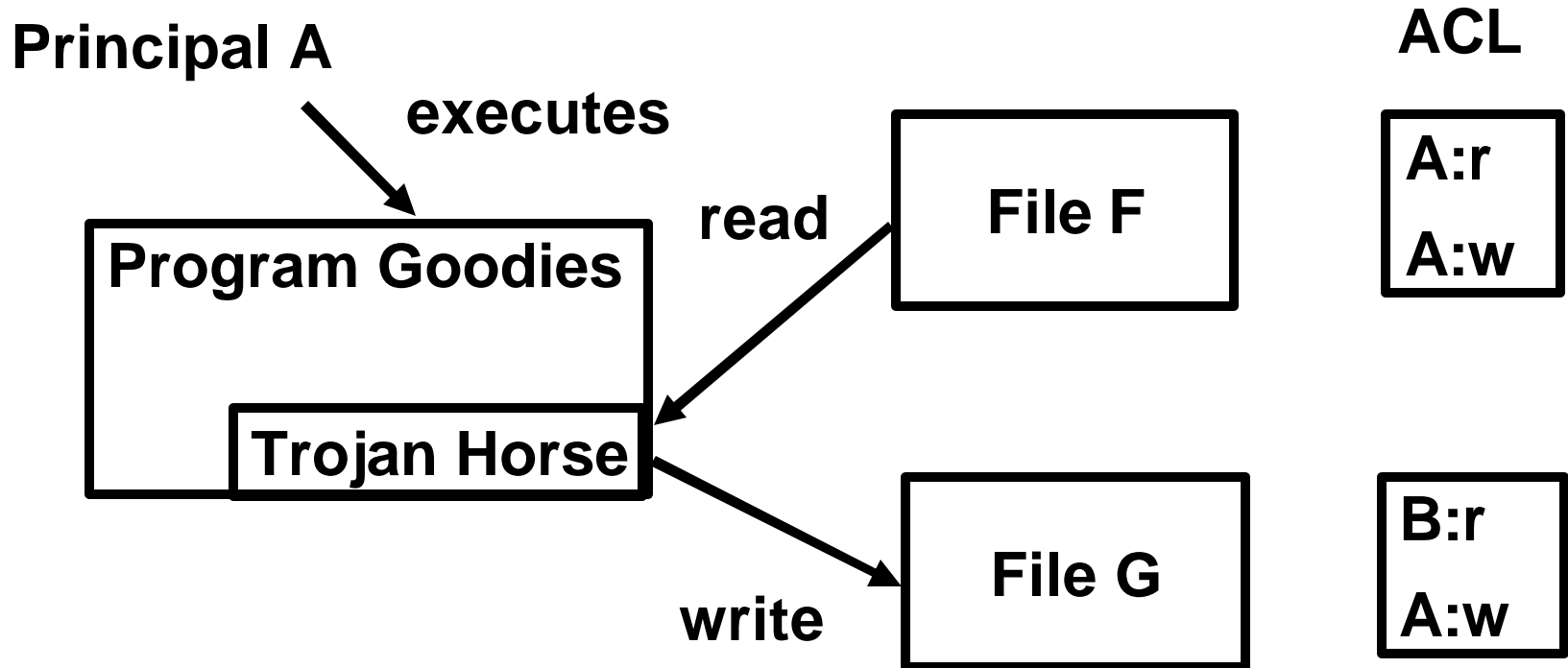
- A Trojan Horse is rogue software installed, perhaps unwittingly, by duly authorized users
- A Trojan Horse does what a user expects it to do, but in addition exploits the user's legitimate privileges to cause a security breach

TROJAN HORSE EXAMPLE



Principal B cannot read file F

TROJAN HORSE EXAMPLE



Principal B can read contents of file F copied to file G

Buggy Software Can Become Trojan Horse

- When a buggy software is exploited, it execute the code/intention of the attacker, while using the privileges of the user who started it.

Coming Attractions ...

- September 25:
 - Lecture by Prof. Cristina Nita-Rotaru:
Introduction to Cryptography

