Introduction to Information Security
14-741/18-631 Fall 2021
Unit 3: Lecture 1:
Access Control in Operating Systems

Hanan Hibshi hhibshi@andrew

This Lecture's Agenda

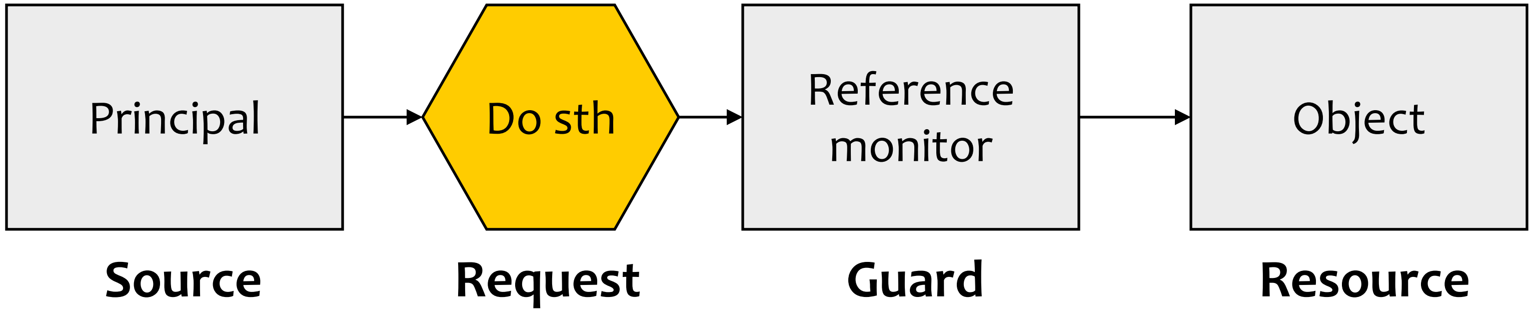
- **Outline**
  - Basic concepts in access control
    - Policy vs mechanism
    - Trusted Computing Base (TCB)
  - Access control principles
  - Access control in computer systems
    - Policies
    - Mechanisms
      - Access control lists
      - Capabilities
      - Case studies: Windows NT4, UNIX
- **Objective**
  - Introduce you to one of the most important security topics in computer security

The Problem

- **Resources need to be protected and shared**
- **Resources:**
  - Valuable objects: bike, phone, cash, …
  - Computation resources
    - Communication channels
    - CPU
    - Memory
    - Files …
  - Information
    - Medical records
- **How do we regulate access?**
  - Saltzer-Schroeder: Access only for computer systems
  - Access control not limited to computer systems (next lecture)

Access Control

- Principal makes a request for an object
- Reference monitor grants or denies the request

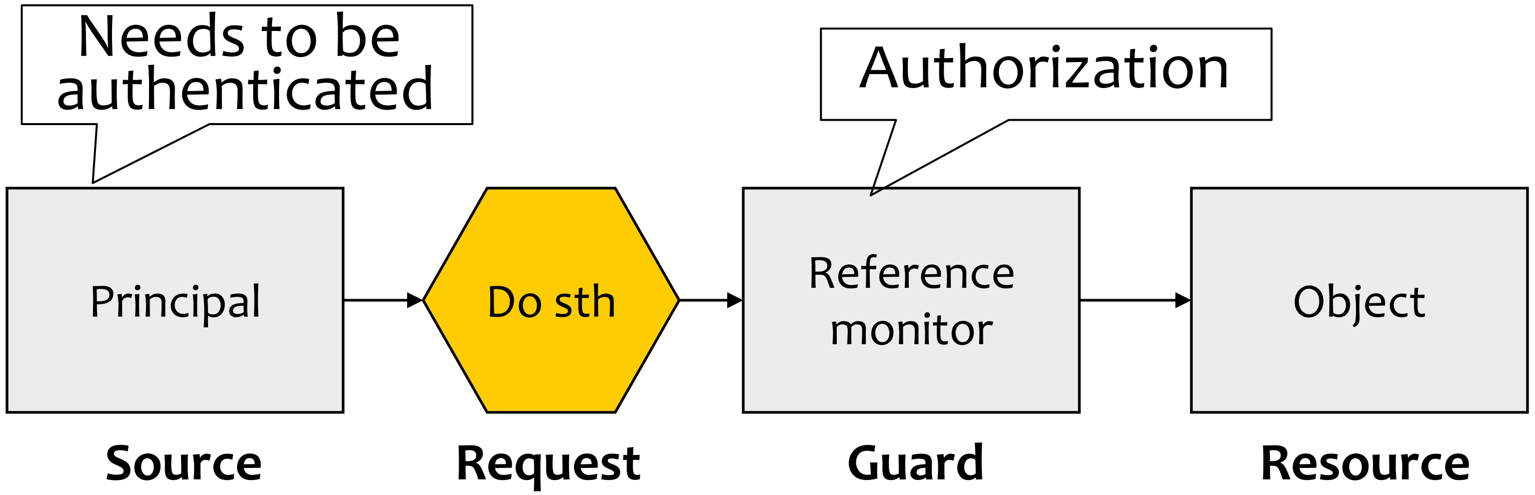


(from Lampson et al., 1992)

Reminder: sth here is short for “something”

Access Control

- **Principal makes a request for an object**
- **Reference monitor grants or denies the request**



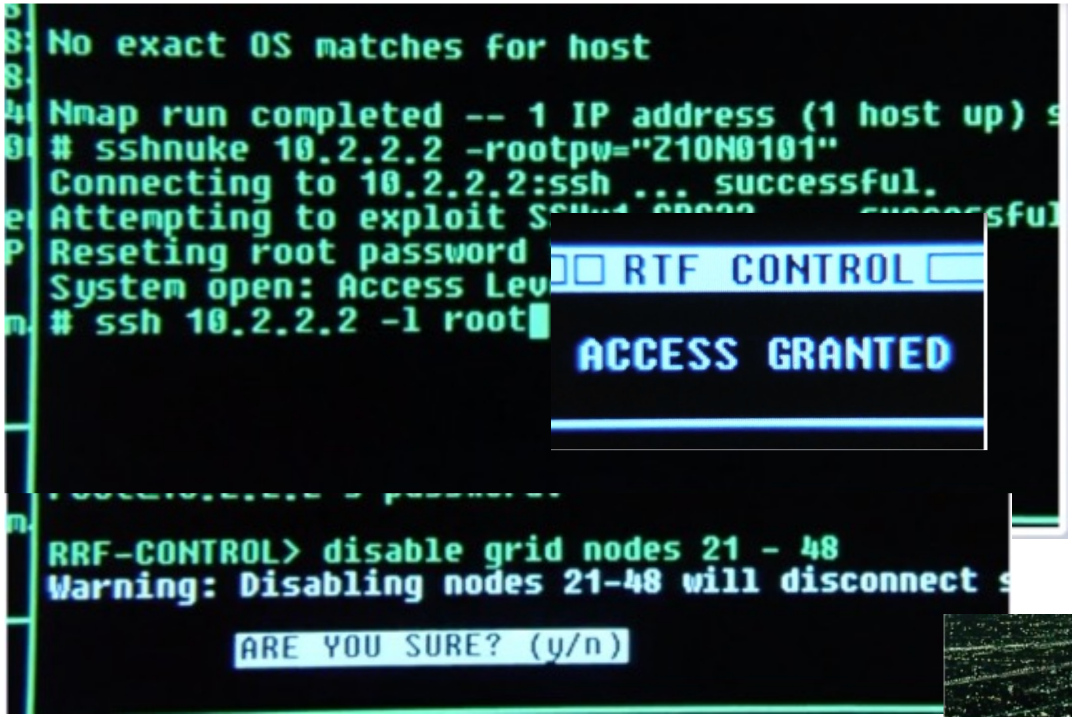
(from Lampson et al., 1992)

- **Authentication: Determining who made request**
- **Authorization: Determining who is trusted to access an object**
  - The "decision" the reference monitor must make

Access Control Aims to Prevent…

- Unauthorized information release (including traffic analysis)
- Unauthorized information modification
- Unauthorized denial of use

Access Control Popular in Movies!



```
No exact OS matches for host

Nmap run completed -- 1 IP address (1 host up) s
# sshnuke 10.2.2.2 -rootpw="Z10N0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit S
Reseting root password
System open: Access Lev
# ssh 10.2.2.2 -l root

RRF-CONTROL> disable grid nodes 21 - 48
Warning: Disabling nodes 21-48 will disconnect s

         ARE YOU SURE? (y/n)
```

RTF CONTROL

ACCESS GRANTED

Access Control Policy vs Mechanism

- ***Policy* is a specification of *who* can access *what*, *when* (*under what condition*)**
  - E.g., cat may enter basement, dog may not
  - Different approaches to organizing policy are *models*
    - Discretionary = users can give access to each other
    - Mandatory = administrator sets policy
    - Role-based access control = all policy specified via roles
- ***Mechanisms* make it possible to implement policy**
  - E.g., cat door (too small for dog)
  - Access control bits for UNIX files & access control lists
  - Firewalls & firewall rules, physical locks & keys
- **Policy and mechanism are related**

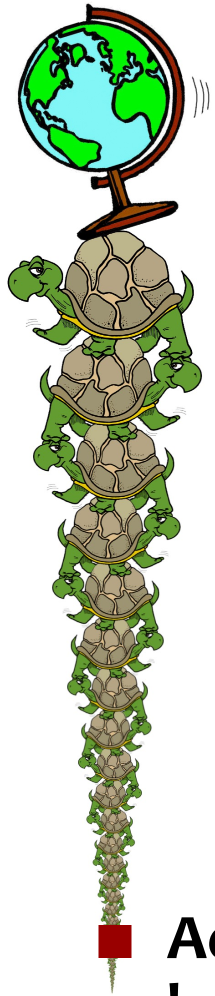
Ack: Lujo Bauer and Michael Reiter

Trusted Computing Base (TCB)

- **The set of components that must function properly for the system to be secure**
- **Must be as small as possible!**
- **Failure of TCB may result in additional access being granted**

- **Example: house guest, cat, food in the pantry**
  - Policy: cat cannot access food
  - System 1: lock pantry door
  - System 2: additionally give house guest key to the pantry and ask the house guest to lock the pantry door

Why is Access Control Complicated?



Can Alice access file X?

Access control policy

Can Alice change who is authorized to have access to file X?

Access control policy of access control policy

Can Alice change the list of people that can change the access policy on file X?

…

- **Add time-dependency into the mix, and you have a very hard problem**

Principles for Access Control (1/3)

(Saltzer and Schroeder, 1975)

- **Economy of mechanism**
  - Keep It Simple, Stupid!
  - Any design or implementation error might break the entire system
  - 3 lines of code are (relatively) easy to secure, 3 million lines are essentially bound to have bugs
  - 3 lines of code can (in general) be formally verified
- **Fail safe defaults**
  - When in doubt, deny access
  - Anybody know of a very popular system where this principle has not been applied?

Principles for Access Control (2/3)

- **Complete mediation**
  - Every access to every object must be checked for authority
- **Open design**
  - See Kerckhoff's principle
- **Privilege separation**
  - Where feasible, ask **two** principals (or more) to unlock the mechanism
  - Avoids single points of failure

Principles for Access Control (3/3)

- **Least privilege**
  - Every program and every user of the system should operate using the least set of privileges necessary to complete the job

```
root@ece001:/ # rm -rf *    (Oops... )
ljia@ece001:/ $ rm -rf *    (Permission denied)
```

- **Least common mechanism**
  - Minimize the amount of shared information
  - Every shared mechanism (e.g., shared variables) represents a potential (untrusted) information path
- **Psychological acceptability**
  - Make it easier for users to use system properly, otherwise they are likely to incorrectly use the protection mechanisms

Access Control in Computer Systems

- **Dates back to first time-sharing systems**
  - CTSS (1961)
- **Multics (late 1960's - running in 1969)**
  - 56,000 lines of PL/1 code
    - (as opposed to 55 millions of C/C++/Asm in Win2K)
  - Vision: one computer for all of Boston (!)
  - Basis for UNIX
    - UNIX is a "Multics light"
- **Difficulty in computer systems:**
  - Dynamic use

Simple Access Control Policies

- **Unprotected**
  - E.g., DOS
  - May contain mistake prevention features, but no security
- **Complete isolation**
  - Each user is in a “sandbox”, has no access at all to other users
  - Maybe with the exception of a few libraries
  - E.g., virtualization (jails in BSD, vservers in Linux…)
- **Controlled sharing**
  - Control explicitly who can access what and under which conditions

Protected Subsystems

- **Collection of programs and data s.t. only the programs of the subsystem have direct access to the data**
- **Access to those programs require calling specific entry points**
- **Indirection**
  - Control access to the entry points implies controlling access to the whole subsystem
  - Facilitates user programmed sharing controls
  - e.g., UNIX subdirectories

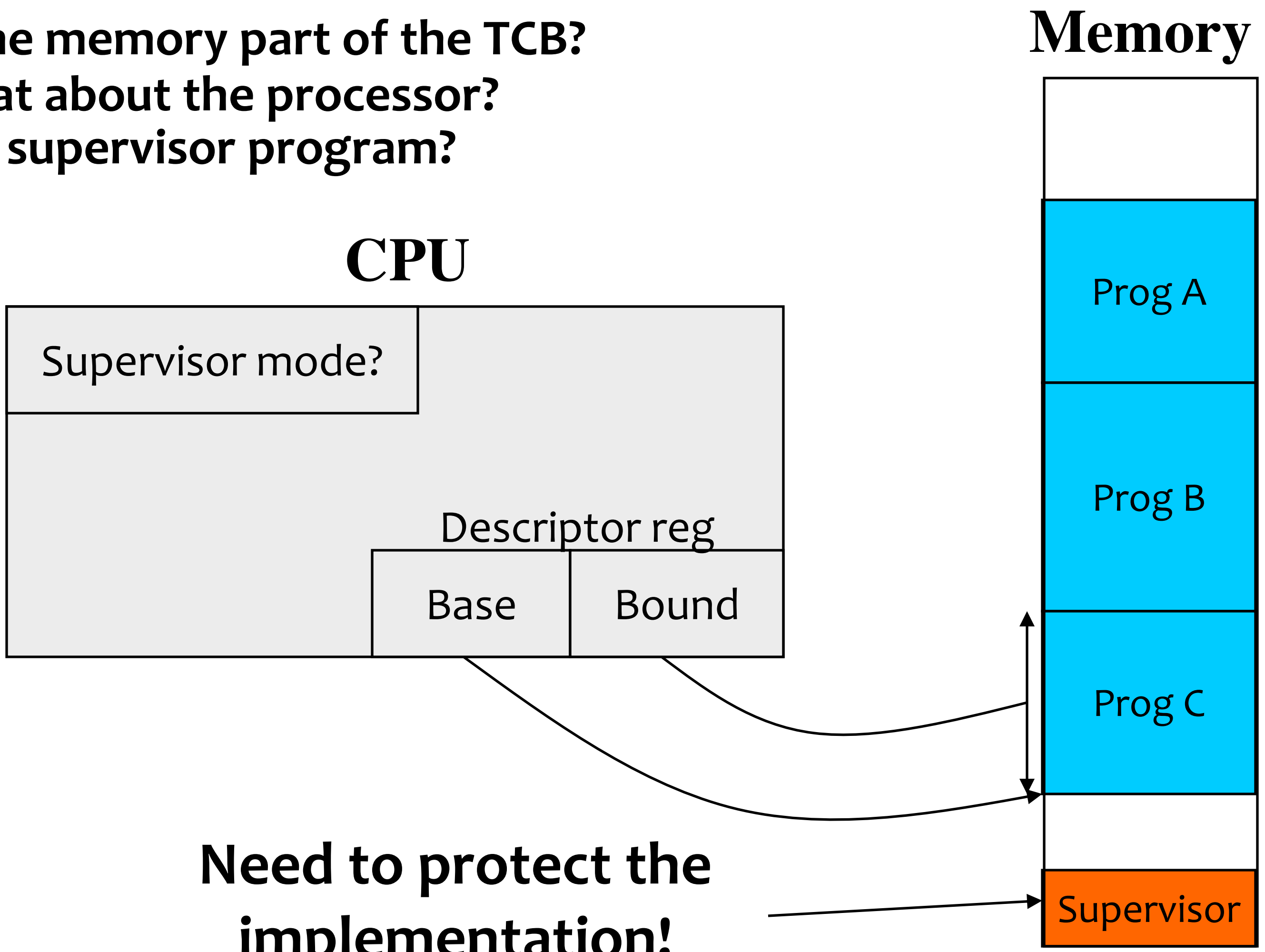
Other Variations

- **Access only to statistical data**
  - E.g., what I got out of the pre-assessment tests for this course
- **Access based on clearance levels**
  - Static labeling of the resource
    - “Top secret” documents

A Simple Isolation Mechanism

TCB?

- **Is the memory part of the TCB?**
- **What about the processor?**
- **The supervisor program?**



Need to protect the implementation!

Shared Information

- **Perfect isolation not necessarily practical**
  - People have to access shared data
    - E.g., databases
  - Programs may want to use code libraries
    - Saves space, software maintenance
  - Hardware resources are shared
    - What happens if one process can read everything being typed on the keyboard while another process is running?

Access Matrices

- **Access control is defined by a triplet (User, Resource, Access)**
- **User**
  - Can correspond to “real” users (alice) or to administrative users/programs (mail, bin, …)
- **Resource**
  - E.g., file, device
- **Access**
  - Read, write, execute, …

Access Matrix Example

	OS	Accounts program	Accounting data	Audit trail
Alice (manager)	`rx`	`x`	`--`	`--`
Bob (auditor)	`rx`	`r`	`r`	`r`
Sam (sysadmin)	`rwx`	`rwx`	`r`	`r`
Accounts program	`rw`	`r`	`rw`	`w`

Access Control List (ACL)

	OS	Accounts program	Accounting data	Audit trail
Alice (manager)	`rx`	`x`	`--`	`--`
Bob (auditor)	`rx`	`r`	`r`	`r`
Sam (sysadmin)	`rwx`	`rwx`	`r`	`r`
Accounts program	`rw`	`r`	`rw`	`w`

Access Control in UNIX

- **Optimization: group by accessor types**

Accessor	**Access to File object X**
owner	read, write, execute
group	read, execute
others	read

- **ACL**
  - store rights with the target object to be accessed
  - ACL representation is compressed to 9 bits (r,w,x * 3 accessor types)
  - Primitive version; has been extended…

suid, sgid

- What you would want for more flexibility is a triple (user, program, file)
- UNIX solves this by allowing programs to essentially pick which user (or group) they are going to run under
- Is this a good idea?

Tale of an `suid` Disaster

- A program needs low level access (e.g., to a CD-ROM drive)
- The programmer makes it suid root rather than going through the trouble of establishing proper communication channels (or forcing authentication on the user's side)
- The program crashes and leaves a "core dump" (information about memory/stack at the time the program crashed)
- The core file is owned by root
- The core file is readable by "world" (rw-rw-rw-)…

Access Control in Windows

- **NT4**
  - Similar to UNIX, but with added flags (take ownership, delete, change permissions)
- **Win2K**
  - Adds capabilities
  - Security policy decided on groups
    - Active directories
- **Vista, Win7/8**
  - User Account Control (eqv. sudo)
- **Windows 10**
  - Going mobile (device management)

Capabilities

	OS	Accounts program	Accounting data	Audit trail
Alice (manager)	`rx`	`x`	`--`	`--`
Bob (auditor)	`rx`	`r`	`r`	`r`
Sam (sysadmin)	`rwx`	`rwx`	`r`	`r`
Accounts program	`rw`	`r`	`rw`	`w`

Capabilities

- The user has a list of his/her access rights
  - Analogy: Your keychain
- But list must be unforgeable!
- Trusted party signs a *ticket* giving you rights to certain resources
- User presents ticket to reference monitor when attempting an access
- Example: Kerberos
- Resource must still keep a list of what *capabilities* can access the resource

ACL vs. Capabilities

(slide adapted from Adrian Perrig)

	ACL	Capabilities
Instant revocation of one object	Easy (one list)	Hard (track down all caps)
Revocation of one subject	Not easy (all lists)	Not easy (all caps for that subject) Eventual revocation is easy (create no more caps)
Giving access to new subjects	Update relevant objects	Issue new caps (easy)
Delegate	Difficult	Easy
Enumerating legitimate users	Easy	Hard
Enumerating accessible objects	Hard	Easy

Different Types of Access Control

- **Discretionary Access Control**
  - The user owning an object decides how other users can access the object
    - Example: UNIX Access Control
- **Mandatory Access Control**
  - Each object has a sensitivity level associated with it, and users have clearances for different sensitivity levels
    - Example: Military security clearances
    - More on this in a next lecture

Role-Based Access Control

- **Each user acts on objects acting in a given role (e.g., manager, programmer) etc**
- **Permissions are assigned to roles**
- **Adding one level of indirection to the access control problem**
- **Role: set of transactions that a user can perform**
    - Allowed transactions for each role determined by sysadmin
    - Each user can act in one of several roles
        - Possible roles determined by sysadmin
        - Active role for a given transaction chosen by user

Take Away Slide

- **Access control in operating systems**
- **Principles for access control**
  - Economy of mechanism, Fail-safe defaults, complete mediation, open design, separation of privileges, least privilege, least common mechanism, psychological acceptability
- **Implementations in modern OS's**
  - Access control lists, Capabilities
  - Potential problems resulting from convenience (the suid bit)
- **Different types of access control possible**
  - By user, by object, by role