Protection and Security

Chapters 17 and 16

Protection

- Protection is all about *mechanisms*
  - We need some way to record:
    - What different programs or users can do with different resources
    - For example, I can access the grades.txt file, but others can’t
    - Or, maybe you can access the DVD burner, but I can’t
  - We need the system to enforce these rules automatically
  - The file system is a typical application area, but it’s not the only one
    - *e.g.,* shared memory segments or named semaphores

Mechanism vs Policy

- With the right mechanisms, we can implement an appropriate *policy* for our needs
- We could do something like this:
  - We'll have three types of users, faculty, grad students and undergrads
  - Printers
    - Maybe the printer in the workroom can only be used by faculty
    - Maybe the printer in the lounge can be used by faculty or grad students
    - Maybe undergrads can't use any printers ☹
  - CPU time
    - Maybe faculty get unlimited CPU time for their jobs
    - Maybe grad students get 10 hours of CPU time per job
    - Maybe undergrads only get an hour ☹
- A good mechanism could let us implement policies like these, but it wouldn't force us to use a particular policy

Protection in the Abstract

- *Object* : a thing you may want to use
  - e.g., a file, a shared memory segment, a portion of the screen
- *Operation* : something you can do with an object
  - e.g., read a file, write to a section of shared memory, draw on the screen
- *Access Right* : a rule that says it's OK to do a particular operation on a particular object
  - e.g., it's OK to write to a file, it's OK to check the toner level in the printer

Protection in the Abstract

- *Protection Domain* : our name for the thing that has access rights
  - An abstraction for a collection of available access rights (essentially, a statement of all the stuff it's OK to do)
  - So, we can talk about a set of rights independently from the idea of a user or a process
  - Maybe each protection domain is associated with a particular user, that's fine
  - Or, maybe a protection domain is associated with a particular process, that's fine also
  - Or, maybe the protection domain could be something more fine-grained (e.g., a particular procedure or a library)

The Access Matrix

- A big, pretend, 2D table that contains all the access rights

	O_1	O_2	O_3	O_4
D_1	read			print
D_2	read	write		check-toner
D_3		read, write	read, execute	print
D_4			execute	print

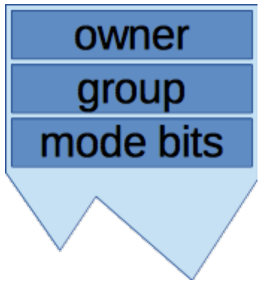
- Each row is a domain
- Each column is an object
- Internal cells say what each domain can do

What’s in the Access Matrix?

- All the standard rights you would expect for files
  - read, write, execute
  - Append (as opposed to overwrite)
  - Create (a new object/file)
  - Delete (an object/file)
  - List (related objects/directory contents)
- Specific rights for certain kinds of objects
  - Print a file on the printer (or, maybe that’s just writing)
  - Check the toner level on the printer (or, maybe that’s just reading)

Standard Unix Permissions

- Permissions associated with each object (file, shared memory segment, etc).
- Every user has a *user id*
  - A number that uniquely identifies them.
- A user can be a member of multiple groups
  - Each with its own numeric *group id*
- Objects have two notions of ownership
  - What user owns the object
  - What group is the object associated with
- Objects store access rights in a fixed-sized field of *mode bits*.



Standard Unix Permissions

- The mode bits are in three groups, indicating
  - What the owner can do
  - What a member of the objects group can do
  - What other users can do (world)
- We have three bits for each, normally called
  - The read bit
  - The write bit
  - The execute bit
- They're organized like this:

For directories, can you get a listing?

Can you add/delete files?

Can you see file attributes?

Owner			Group			Other		
R	W	X	R	W	X	R	W	X

Working with Unix Permissions

- You're probably familiar with the shell commands:
  - chmod 750 file.txt
  - chmod o+r file.txt
  - chown bill file.txt
  - ls -l file.txt

Owner can read, write and execute.
Group can read and execute.
Other can't do anything.

Yay, a reason to know octal.

… or not.

Print files with attributes, including permissions.

```
-rwxr-x---  1 sjiao2  NCSU  39 Feb 11 09:12 file.txt
```

Working with Unix Permissions

- Underneath, there must be a system call interface:
  - fchmod(int fd, mode_t mode);
  - fchown(int fd, uid_t owner, gid_t group);
  - fstat(int fd, struct stat *buf);

Security

- Protection is focused on internal concerns
  - What mechanisms do we need …
  - … and what actions do we want to permit on a system.
- Security is focused on external threats
  - For example, can I get around your protection mechanisms by just walking away with your hard drive?
  - Security is a (never ending) effort to make sure these kinds of attacks won't be successful.

Security Fundamentals

- An *attack* an attempt to gain unauthorized access to a system.
- There are different types of attacks:
  - Breach of confidentiality
  - Breach of integrity
  - Breach of availability
  - Theft of service
  - Denial of service

Types of Attacks

- Bad programs can represent security threats:
  - Back-door
  - Trojan horse
  - Logic bomb
  - Timing attack
  - Stack/Buffer overflow
  - Virus
    - A lot can be said about viruses
    - Where do they store their code?
    - How do they avoid detection?

Security and Networks

- The network interface to your system can offer a very large *attack surface*.
  - Every misconfigured service or software bug could be a vector for attack.
- Port scanning
- Worms
- Zombie systems
- Distributed denial of service