Security – Access Control

Note Title 1/26/2011

Announcements

- Essay should be done
- Next week, you'll get lab 1

Access Control

The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.

$\boxed{\text{Probably the central element of computer security.}}$

Incorporates (Chapter 4)

1) Authentication (last lecture) (Ch10)

2) Authorization

3) Audit

Access Control Policies
(governs authorization)

1) Discretionary Access Control (DAC)
2) Mandatory Access Control (MAC)
3) Role-Based Access Control (RBAC)

(These aren't necessarily mutually exclusive.)

Terminology

- subject : a process (or user)
  - ↳ 3 classes
    - owner
    - group
    - world
- object : a resource

Dfn: <u>Access rights</u> describe ways which subjects may interact with objects.

Ex:
- read
- write
- execute
- delete
- create
- search

Discretionary Access Control (DAC)

- Most common in modern OS
- Based on subject's identity combined with access rules stating what each subject is allowed to do.

<u>Note</u>: An entity may be given access rights which allow it to give <u>another</u> subject access rights.

Visualization: Access Control Matrix

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(based on Lampson in '71, image taken from course text)

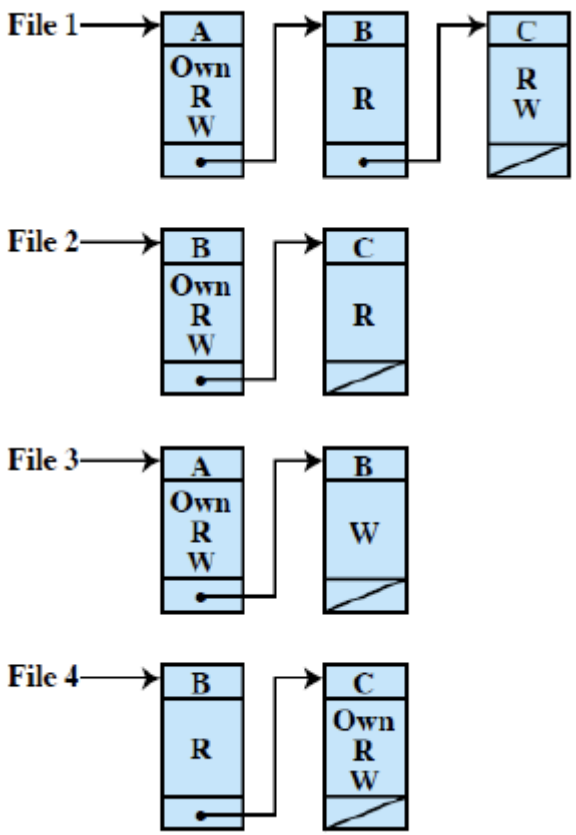
How to implement this matrix?

In practice, this matrix is <u>very</u> sparse.

(Think of the number of users and files on our Linux systems, much less on larger lab's.)

How to solve this?

Windows OS: Access Control Lists



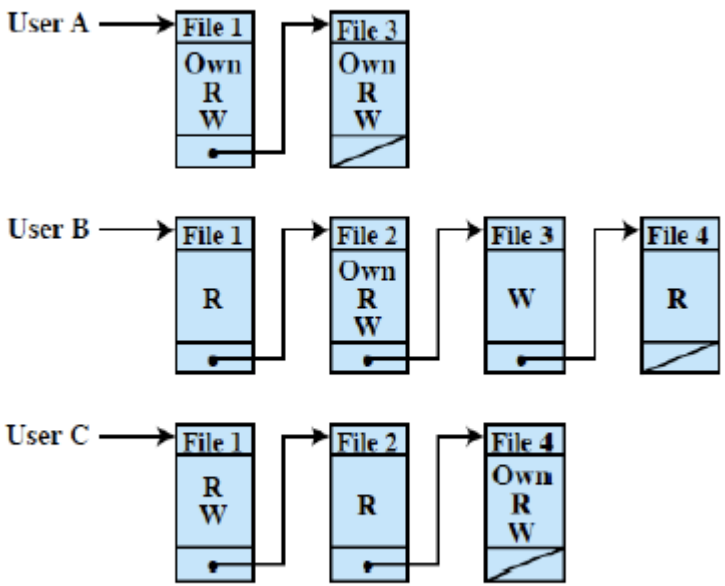
The good:

- space efficient
- fast if accessing a particular file

The bad:

- a particular list can get very long
- listing files A has access to is slow

Flip this: Capability Lists



The good:

The bad:

Case Study: UNIX

- Files & directories in a tree structure
- Each user has a user id (uid) and at least one group id (gid)
- At creation, a file is set to its creator's uid and either its owners gid
  <u>or</u> its parent directory's gid
  (if directory has setgid permissions)
- Each file gets 12 protection bits

Protection Bits (in UNIX)

- 9 bits specify read, write and execute privledges for:
  owner, group, world

eg:

111	101	000
rwx	r--	---
7	5	0

chmod 777 myfile

- remaining bits specify additional behaviors

$\underbrace{\text{setuid, setgid}}_{\text{govern executables}}$, "sticky"

when someone else executes file, it runs with their user id

in a directory, means only owner of <u>any</u> file can rename, move, or delete that file

Security in UNIX

- the super user account (root) can do anything

How is this good and bad?

Good: some one has to fix it all

Bad: big, fat target

UNIX (cont)

Most UNIX filesystems (including ours) also use ACLs.

- Admin can assign any # of uids & gids to a file using setfacl
- Any file may <u>not</u> have additional entries. If it does have ACL entries, an extra bit is set

(use getfacl to check)

<u>demo</u> - chmod

getfacl

Mandatory Access Control (MAC)

Based on comparing security labels with security clearances.

(Evolved for military + government settings)

TS/SCI → top secret, secret, unclassied

Mandatory — a subject with access to some resource may not share that access with another subject

Top Secret!

In the real world,
can't just wave
your gun & "hack"
to get the information!

Image made available by Wikimedia Commons

Example: Bell-Lapadula Model

- Every subject gets a security clearance
- Every object gets a security classification

2 principles

1. No read up (simple security property)

   don't want individuals to access info w/ a higher classification

2. No write down (*-property)

   Should not be able to "unclassify" anything

Many others exist (Ch 10 in the text)

- Biba (military)
- Clark-Wilson (commercial)
- Chinese wall
  (for conflict of interests in commercial applications)

Role-Based Access Control

Access rights are based on what roles the user assumes in the system rather than the user's identity

Example: Doctor's office

Medical records

role division

Visualization:

users \ roles	R_1	R_2	R_3	R_4 ...
U_1	X		X	
U_2		X		
U_3	X			
U_4				
⋮				

	roles			files		directories	
	R_1	R_2	...	F_1	F_2 ...	D_1	D_2 ..
R_1							
R_2							
R_3							
R_4							
R_5							
⋮							

- roles may own or control other roles, as well as files + directories

Next week:

- First lab
- Cryptography