Computer Security

03r. Assignment 2 review

Paul Krzyzanowski
Rutgers University
Spring 2017

Question 1

What is an access control list (ACL)?

- A list of user access permissions associated with an object
- An access control list stores the access control matrix one column at a time, each

Question 1 Discussion

An access control matrix is a general way of representing access control rights

- Each row represents a *domain (subject)* = usually a user or a group of users
- Each column represents an *object* = usually a file or a device

Objects (usually files or devices)

domains of protection (users or groups)

	F₀	F₁	Printer
D_0	read	read-write	print
D_1	read-write-execute	read	
D_2	read-execute		
D_3		read	print
D_4			print

Question 1 Discussion

It is not practical to manage an access control matrix in an operating system

- Often 100,000+ objects (& shared systems may have 1,000s of users)
- Many files get created and deleted throughout the day
- You'd need to run a database to manage the matrix
- OS needs something efficient:
  read as few blocks as possible from the file system
- ACL:
  - Stored with a file: part of metadata that contains information about the file
  - Contains a set of Access Control Entries (ACEs)
  - Each ACE contains
    (1) user or group
    (2) access rights

Question 1 Discussion

It is not practical to manage an access control matrix in an operating system

- Often 100,000+ objects (& shared systems may have 1,000s of users)
- Many files get created and deleted throughout the day
- You'd need to run a database to manage the matrix
- OS needs something efficient:
  read as few blocks as possible from the file system
- ACL:
  - Stored with a file
  - Contains a set of Access Control Entries (ACEs)
  - Each ACE contains
    (1) user or group
    (2) access rights

ACL for F_1

Objects (usually files or devices)

domains of protection (users or groups)

	F₀	F₁	Printer
D_0	read	read-write	print
D_1	read-write-execute	read	
D_2	read-execute		
D_3		read	print
D_4			print

Question 1 Discussion

- Unix used a simplified version of an ACL
  - Three sets of access rights
    - Owner of the file
    - Group associated with the file
    - Everyone else
  - Each set includes *read*, *write*, and *execute* permissions
    - Owner: rwx, Group: rwx, Other: rwx ⇒ rwxrwxrwx = 9 bits of data!
- The simplified access rights use a fixed amount of data
- Fits into an inode
  - Fixed-length data structure that stores file metadata (size of file, creation time, last modification time, last access time, owner ID, group ID)
- Full ACLs are supported in Linux
  - But accessing them requires the kernel to read extra blocks from the file system (extended attributes)

Question 2

What is the purpose of the set user id (setuid) file attribute in Unix systems?

The set user ID (called setuid) attribute enables a program to run with the privilege of the owner of the file rather than the privilege of the user who ran the program

Question 2 Discussion

- Normally, when you run a program, it runs under your user ID
  - If the program needs to open a file, it checks access rights based on your user ID and your group ID
- If the setuid bit is set in the file permission bits of the program file
  - The program runs under the ID of the owner of that executable file
  - If it needs to open files, it checks access rights based on the owner of the executable file
- For example, on Ubuntu Linux:

```
-rwsr-xr-x 1 root root   40152 Dec 16 10:40 mount
-rwsr-xr-x 1 root root   44168 May  7  2014 ping
-rwsr-xr-x 1 root root   54256 Mar 29  2016 passwd
```

Question 2 Discussion

Setuid allows normal users to run programs that require elevated privileges

For example:

- Programs that need to access restricted files
  - *passwd* (the program that lets you change your password) needs to be able to open and write to the password file
- Programs that need special network access
  - *ping* (the program that lets you test network links) needs to create & receive IP ICMP packets. This requires access to raw sockets
  - Note that some systems created a special ICMP socket type so users can send and receive ping messages without elevated privileges
- Programs that need access to restricted devices
  - The *mount* command mounts file systems to the system namespace.
  - File systems are block devices that do not have read-write access for everyone
  - An administrator can configure some file systems to be user mountable (e.g., USB-connected disks) but this means that the user needs to have access to *mount* & *unmount*

Question 2 Discussion

- There's also a "set group ID" bit (setgid) that works the same way
  - Programs run under the user's ID
  - BUT with the group ID of the program's group

Question 3

What is the purpose of the wheel group on BSD and macOS systems?

- It identifies the set of users that are permitted to use the *su* command to change to root

Question 3 Discussion

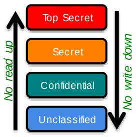
- The *su* command is a setuid command owned by root
  - When run, the user is prompted for the root password
  - After authentication, the user has a root shell (runs with root privileges)
  - The user can then use the su command to change to any other user's privileges without being prompted for a password

    su bob # become bob
- This program has a lot of power (you become the adminstrator)
  - To limit possible abuse (e.g., trying to guess the root password), only users in the wheel group will be permitted to become root
- Note: The "wheel" concept isn't implemented on every Unix variant:
  - It came from BSD and is present on BSD systems, macOS, Red Hat Enterprise Linux, CentOS

Question 4a

What is the **simple security property** of the Bell-LaPadula model?

- No process may read data from a higher level of classification: *No read up*.
- If your classification level is Secret, you can only read Secret, Confidential, or Unclassified files – but not Top Secret

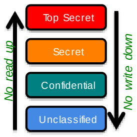


Confidential cannot read Secret
Confidential cannot write Unclassified

Question 4b

What is the ***-property** (star property) of the Bell-LaPadula model?

- No process may write data to a lower level of classification: *No write down*
- If your classification level is Secret, you can only write Secret & Top Secret files



Confidential cannot read Secret
Confidential cannot write Unclassified

Question 4 Discussion

- The Bell-LaPadula model is all about confidentiality
  - You cannot read data from higher clearance levels than you are
  - You cannot create data that is a lower clearance level than you are
- It's difficult for only the operating system to enforce this
  For example:
  - A mail application should have defined policies on whether you are allowed to mail a file … or even send a message (a person at a *top secret* level should not be able to send a message to someone with *secret* clearance)
  - Databases can be challenging if they hold a mix of data levels

Question 4 Discussion

- The Bell-LaPadula model is all about confidentiality
  - Simple **security** property:
    - You cannot read data from higher clearance levels than you are
  - Star *-property:
    - You cannot create data that is a lower clearance level than you are
- The Biba model is similar but is all about integrity
  - Simple **integrity** property:
    - You cannot read an object from a lower integrity level than you are
    - Example: A process will not read a system configuration file created by a lower-integrity-level process
  - Star *-property:
    - You cannot write to an object of a higher integrity level than you are
    - Example: A web browser may not write a system configuration file

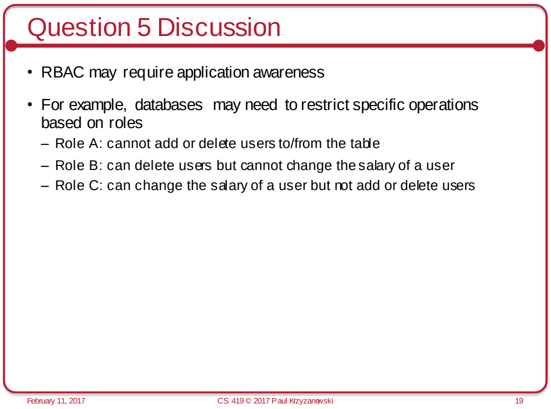
Question 5

What is meant by a role in a role-based access control (RBAC) system?

A *role* can be thought of as a set of transactions or operations that a user or set of users can perform within the context of an organization

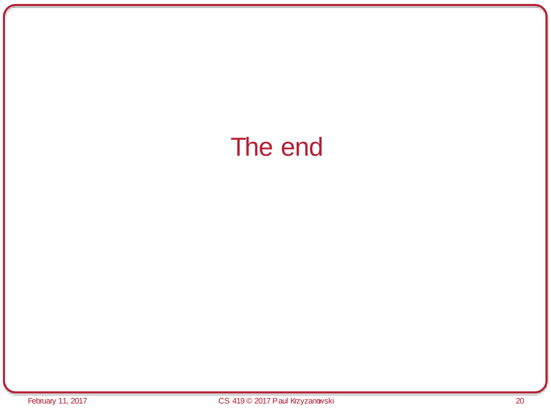
Question 5 Discussion

- Role-based access control (**RBAC**) is built around identifying specific sets of tasks (roles) users need to do
- Steps
  1. An administrator assigns a user one or more roles
  2. The user logs in or may need to specifically authenticate for one of their roles
  3. The system validates operations on the object based on the user's role
- Unlike access control lists,
  - RBAC assigns permissions to roles rather than users
  - It's one extra level of indirection user → role → object
- Mapping between users and roles can change dynamically
  - For example, get a substitute worker or a new employee in a group
- Mapping between roles and objects can change dynamically
  - For example, if developers in a certain project no longer need access to a specific source repository



Question 5 Discussion

- RBAC may require application awareness
- For example, databases may need to restrict specific operations based on roles
  - Role A: cannot add or delete users to/from the table
  - Role B: can delete users but cannot change the salary of a user
  - Role C: can change the salary of a user but not add or delete users



The end