CS 458 Information Security

Homework 6: Access Control

1. Concepts

(a) The three most important components in access control, all starting with the letter 'A', are what?

Authentication (verifying the user is who he says he is),

Authorization (verifying the user is allowed to execute the command),

Audit (a log of all events for future reference).

(b) Three models of access control we'll talk about in class are DAC, MAC and RBAC. What do they stand for?

Discretionary **A**ccess **C**ontrol (owner of object decides who has access at her *discretion*),
Mandatory **A**ccess **C**ontrol (user *must* have certain clearance, even if the owner allows access),
Role-**B**ased **A**ccess **C**ontrol (users are assigned *roles*, and permissions are given to roles).

(c) In access control, what does an "open policy" mean? What does a "closed policy" mean?
An open policy allows all access except where noted (uses a blacklist).

A closed policy forbids all access except where noted (uses a whitelist).

(d) What did MAC stand for in cryptography? **Bonus:** what does MAC usually stand for in networking (as is "MAC address")?

Message Authentication Code.
Media Access Control. This is not quite the same 'access control' as what we're talking about right now.

2. Access Control Matrix

(a) Three important elements in access control are subjects, objects and access rights. In an access control matrix, how are the three elements presented?
Each row is a subject. Each column is an object. Each intersection denotes what access rights that subject has on that object.

(b) [False] "Objects" are another name for the files on a system.
Objects often refer to files, but can be any resource on a computer (such as a network port or a mutex).

(c) An access control matrix is often sparse, and in implementation people generally use some other data structures. What are some common ways to implement an access control matrix?
Access Control Lists (slice by column, used by most modern OS),
Capability Lists (slice by row, sometimes known as *capability tickets* because they can be passed around to grant other users access),
Authorization Table (list of all triples (subject, object, access right) in the matrix).

3. Principles

(a) What does *least privilege* mean? What problems could *least privilege* prevent?
A user executing a command should not have more privilege than needed for that command.
It prevents user error, as well as malicious activities carried out in the user's name.

This is why sometimes the OS warns you when you try to `sudo` (run as admin).

(b) What does *separation of duty* mean? What problems could *separation of duty* prevent?
A single user should not have enough power to corrupt the entire system (or any major part of it).
This prevents conflicts of interest, as well as fraud (both by humans and by malicious programs).

Separation of duty is a more than a computer security principle. The major idea is that the more people needed to do a certain task, the more difficult it is for them to collude and do harm.
In security we *always* assume the worst of people.

4. RBAC

(a) What advantages does role-based access control have?
RBAC is:

- Flexible (can implement a wide range of access control policies)
- Inexpensive (costs less to administrate)
- Easy to use (less likely for administrators to make mistakes, especially for a system with 500+ users).

Answers may vary. It's hard to define what RBAC exactly is because there are many different interpretations and implementations, but they all share certain properties like the ones listed above.

(b) Core RBAC specifies users, roles, permissions and sessions. What does $RBAC_1$ add on top of core RBAC? What does $RBAC_2$ add on top of core RBAC?
$RBAC_1$ supports role hierarchy.
$RBAC_2$ supports role constraints, such as *exclusive roles* or *prerequisite roles*.

$RBAC_3$ is a consolidated model that supports everything in $RBAC_0$, $_1$ and $_2$.

(c) [True] RBAC supports the principles of *least privilege* and *separation of duty*.