COM-301 Computer Security
Exercise sheet: Access Control

1. Which of these principles are related to access control and how?

   (a) Least common mechanism
   (b) Least privilege
   (c) Open design
   (d) Complete mediation

   Justify your answer, both when positive and when negative

   Solution:

   (a) Access control does not support least common mechanism. In fact, it is generally the opposite. All users in a system tend to share the access control system.
   (b) Access control supports least privilege. Access control should be built to give permissions to subjects on a need to know basis.
   (c) Access control does not support open design. This is an orthogonal principle implementation/design oriented. Of course, one should follow this principle when designing an access control mechanism.
   (d) It does not support complete mediation. In fact, we do have access control in all the operating systems we work with, yet we do not have complete mediation (i.e., the access control does not happen before every operation). This is because, as said in class, having complete mediation has a huge overhead and would make the system unusable. Yet, the definitions are similar and related. In fact, (the quest for) Complete Mediation is the principle that guides Access Control designs. Ideally, one would have a reference monitor that controls any access (see lecture slides: Implementing Access Control). In this sense, Complete mediation is the perfect access control (i.e., complete mediation supports access control); but the opposite is not necessarily true. This is why the answer to the question is no.

2. When you decide to assign permissions to users according to a job function, what is this called?

Solution:
We call this type of access control "Role-based access control". In this type of access control, having a particular job is associated to a set of permissions. When a new subject is added to the system, the subject is associated to a job and inherits the permissions available to that job.

3. ACME Corporation is planning to implement an access control mechanism in which employees control who has access to their own information assets. What is this type of access control?

   Solution:
   This type of access control is Discretionary Access Control. The reason is that in the described system, the owner of an object is the one establishing the policy to access that object. This is in contrast with Mandatory Access Control, in which the policy is established by the system itself and object owners cannot take access control decisions.

4. What is a negative permission? How are negative permissions different from blacklisting permissions as access control? Provide an example from Linux access control.

   Solution:
   A negative permission is an attribute that removes a permission from a user/group.

   A negative permission is different from blacklisting in that a negative permission is a modification of a whitelist: We first define all the "good permissions" and then we remove the ones that are not needed.

   A blacklisting access control system would contain **all** the non-authorized permissions. Such a mechanism is prone to errors (how do you know you have all the negative permissions marked?). Constructing access control in this way would be against the fail-safe principle.

   Two mechanisms in UNIX implement negative permissions:

   The sticky bit implements negative permissions. It prevents others from deleting a file, even if they have been given other permissions on the directory.

   The permission system enables the owner to remove permissions from themselves while giving it to the group. Since user permission supersedes group, this effectively creates a negative permission for the user on the file.

5. Check the following table and list all files which Alice can write into.

   - Alice is a member of groups *alice* and *pcrack*
   - Dave is a member of groups *dave* and *gdev*

Permissions	Owner	Group	Size	Last update	File name
- rws - - - - -x	dave	gdev	134516	Sep 08 21h10	hello
drwxrwxrwt	dave	gdev	14586	Aug 01 14h30	program
- rwx - - x - - x	alice	alice	214768	Sep 10 07h35	hosts
- rw - r - - - - -	alice	pcrack	12486	Sep 10 08h09	config
- rw - r - - r - -	dave	pcrack	98774	Aug 28 15h10	data
- rw - - wxr - -	root	pcrack	12257	Sep 15 10h46	setup

Solution:
Alice can write into *hosts* (direct permission as owner), *config* (direct permission as owner), and *setup* (permission as pcrack group member)

program is a directory. Alice can write to it (i.e., add a file). However, due to the sticky bit, she cannot delete or rename the directory.

6. Imagine a grocery store in which the only way of leaving with goods is having a cashier scan the barcode of each bought item to determine the total cost. A user willing to cheat the system could replace some of the barcodes of his expensive items with a barcode of cheaper items such that, when the cashier scans the items, the bill is lower than the actual price. What is the role the cashier plays in this situation?

   Solution:
   This is a case of the confused deputy problem.

   - The thief, as a client, cannot directly change the prices to be written in his bill.
   - The cashier, as an employee of the grocery store, has the right to add items to the bill.

   When being tricked into scanning the wrong prices for the items that the thief is buying, the cashier becomes a confused deputy that enables an unprivileged client to modify the cost of the groceries.

7. What principle is supported by user Nobody in UNIX? Justify your answer.

   Solution:
   The Nobody user supports the least privilege principle. Such an account ensures that programs run with the minimum set of permissions possible so that if compromised (or buggy) they cannot do any damage to the system.

8. Consider two mechanisms to access a bank account. In Mechanism A, each owner is authorized in multiple accounts. By showing your userID, you are provided with a token to access to all accounts you are authorized. In Mechanism B, each account has associated a list of userIDs (accountID,userIDs). If your userID is in the account's list, you can access the

account. Which one implements ACL and which one Capabilities. In this scenario how are the problems of these approaches materialized?

Solution:
Mechanism A implements Capabilities. For a given user, there is a list of all the accounts she has access to.

- Revoking access to an account is difficult. One needs to go through all the users to and find who has access to it.
- Once you have the token to access the accounts, there is no way of avoiding transfers.

Mechanism B implements ACL. For a given account, being or not on the list establishes access rights.

- There is no record of what accounts can a user access. It is difficult to get an overview of the users' permissions in the system.
- Deleting a user from the system is hard. How to ensure that she is in no account's list?
- Delegation is difficult, permission is account-based.