7.27. A process having euid as its effective user id and egid as its effective group id is authorized to perform an operation requiring a privilege p provided (7.9) holds. What are the consequences of replacing (7.9) with the following.

(p ∈ Privs F .owner ∧ euid = owner F)
∨ (p ∈ Privs F .group ∧ egid = group F)
∨ (p ∈ Privs F .other)

Let us assume a scenario where a process has an effective egid and euid such that euid = owner F and egid = group F. The process wants to perform a read operation which is defined under privileges of both PrivsF.owner and PrivsF.group. As per 7.9 only one set of permissions is ever used at a time because the dis-junction (1) would evaluate to T where as the dis-junction (2) will evaluate to F. But for the above given equation both the dis-junctions (1) and (2) would evaluate to T which violates the Principle of Least Previleges.

As per the 7.9 if you are the owner, only the owner permissions are looked at the time and the group permissions are ignored. If you are not the owner but are in the group, then only the group permissions are used. If you are not in the group and not a owner then the other permissions are used. If you are *both* the owner and also in the group, then the group permissions does not matter.

7.33. NDP (rts) below is an alternative definition for the set of privileges associated with the protection domain for a run-time stack rts.

$$NDP(rts): \begin{cases} FPrivs(\mathbf{top}(rts)) & \text{if } \mathbf{empty}(\mathbf{pop}(rts)) \\ NDP(\mathbf{pop}(rts)) \cap FPrivs(\mathbf{top}(rts)) & \text{if } \neg\mathbf{empty}(\mathbf{pop}(rts)) \\ & \quad \wedge \neg IsPriv(\mathbf{top}(rts)) \\ NDP(\mathbf{pop}(rts)) \cup FPrivs(\mathbf{top}(rts)) & \text{if } \neg\mathbf{empty}(\mathbf{pop}(rts)) \\ & \quad \wedge IsPriv(\mathbf{top}(rts)) \end{cases}$$

NDP (·) and DomPrivs(·) differ in how stack frames marked privileged are handled.

Let us first understand the definition stated in the question.

1.
a. If empty(pop(rts)) then it means that the stack becomes empty after popping a file. If there is only one frame in the stack then it takes the privileges of that file.
b. Fprivs(top(rts)) means it takes privilege of file F1.

2.
a. If ~empty(pop(rts)) ^ ~IsPriv(top(rts)) it means that the top of the stack should not become empty and the top frame of the stack should not be a privileged one.
b. NDP(pop(rts)) ^ Fprivs(top(rts)) means pop one frame from stack and is intersection of the top frame. Therefore it takes the privileges of file F1 which is the top frame of the stack.

3.
a. If ~empty(pop(rts)) ^ IsPriv(top(rts)) it means the stack should not be empty and the top frame of the stack should be privileged one.
b. NDP(pop(rts)) v Fprivs(top(rts)) means there is more than one frame in the stack and the top frame is privileged one. It has more privileges than compared to the first two conditions.

We also know that DomPrivs(rts) is the intersection of all frames in stack and the top frame is privileged one.

a. Can NDP(rts) = DomPrivs(rts) hold some snapshot of some runtime stack rts? If so, give example code units, associated declarations of privileges, and indicate the call sequence for the snapshot. If not, explain why.

NDR(rts) intersects with all the frames of the stack and DomPrivs(rts) also intersects with all the frames of the stack up to the top if there is no privileged frame. Therefore NDP(rts) = DomPrivs(rts) will hold true if there is no privileged frame in the stack.

b. Can NDP(rts) < DomPrivs(rts) hold some snapshot of some runtime stack rts? If so, give example code units, associated declarations of privileges, and indicate the call sequence for the snapshot. If not, explain why.

As per the definition NDP(rts) unions in case of marked frames present in the stack whereas DomPrivs(rts) resets the privilege. Therefore NDP(rts) can never be a subset of DomPrivs(s).

c. Can NDP(rts) > DomPrivs(rts) hold some snapshot of some runtime stack rts in a system of code units? If so, sketch the code units and associated declarations.

As per the definition DomPrivs(rts) can become a subset of NDP(rts) when we have a privileged frame present in the top of the stack.

d. Discuss why DomPrivs(rts) is a more sensible definition for protection domains than NDP(rts) is.

DomPrivs(rts) has the more sensible definition for the protection domains because it avoids the risk of confused deputy attacks. In case of NDP(rts), whenever we get a marked frame, a privilege is not only amplified but is also added to the privilege of the below frame which was calculated so far thereby increasing the risk of accessibility of unauthorized objects leading to confused deputy attacks.

7.29. The text (page 136) draws a parallel between Unix file descriptors and capabilities. What are the similarities and differences?

A file descriptor defines a table which consists of the list of files and the permissions such as read/write/execute on those files whereas a capability extends the UNIX file descriptors encapsulating the rights on specific objects. Both the file descriptors and capabilities serve the purpose of identifying a memory segment and specifying the access rights to that segment.

The primary difference between the descriptor and capabilities is on the unbound latency on revocation and manipulation of the access rights which depends on the underlying system hardware and software.

7.35. Consider a social network, comprising individuals linked to each other according to a relation BFF. In particular, access to postings is based on BFF , as follows: If <I, I'>∈ BFF holds then individual I is authorized to read postings by individual I'. Goals that might constrain BFF as it applies to a given
individual I could include
(i) restrict which individuals can read postings by I, and
(ii) restrict which individuals' postings I can read.

Let us consider a social networking platform like Facebook for this question. Let us assume that there are two individuals I and I' such that <I,I'> belongs to relation BFF. I can read the postings of I' which can happen only in two cases of relation.

1. Either I' is a friend of I and I' privacy is limited to friends (or)
2. I is not a friend of I' but I' privacy is limited to public.

Let us consider that I is related to a set of I' such that {I'1,I'2,I'3,….}

(a) Give a plausible situation where goal (i) is useful.

Incase if I is related to a set of I' through relation (1) then I can use this goal to hide his/her postings from his friends in the set I'.

(b) Give a plausible situation where goal (ii) is useful.

Incase if I is related to a set of I' through relation (2) then I can use this goal to prevent viewing of the postings by some individuals from the set I' on whom he is not interested on.

(c) To what extent do (i) and/or (ii) become more difficult to preserve if the
BFF relation is symmetric and/or it is transitive.

Let us consider the BFF relation to be symmetric where if I can view posting by I' then vice versa is also possible. Incase of relation (1) then restrictions I and ii can be maintained but for relation (2) it is difficult to maintain both the goals. If the BFF relation is transitive then both the goals i and ii cannot be maintained for both the relations (1) and (2).

7.26 Construct an authorization relation Auth that models the semantics of access control lists in Unix. Explain how your construction supports the protection domain changes possible in Unix. For simplicity, assume that each user is a member of exactly one group (as was the case for the very first version of Unix).

UNIX divides the Authorization into two levels. One is the ownership. UNIX has three owner ships namely user, group and other. The second is the permissions such as read, write and execute defined on a file or directory.

An Auth is typically defined using following semantics

Auth = <P, Ob, Op> where P is the Principal(user,group,..), Ob is the Object(directory,file,...) and Op is the operation(read,write,..)

For example consider the below screenshot,

```
mithilaesh@mithilaesh-HP-Notebook:~/Documents/Job$ ls -l
total 1832
-rw-r--r-- 1 mithilaesh mithilaesh  16896 Feb  4 13:48 'cover letter.doc'
-rw-r--r-- 1 mithilaesh mithilaesh 706330 Jan 23 21:19  Resume.pdf
-rw-r--r-- 1 mithilaesh mithilaesh 699768 Feb  6 21:32  SodexoRetirement.pdf
-rw-r--r-- 1 mithilaesh mithilaesh 446227 Feb  7 01:08  TaxReturn2019.pdf
```

Here the first name ‘mithilaesh’ defines user who can read/write ‘Resume.pdf’. The second name ‘mithilaesh’ defines group where members can only read ‘Resume.pdf’. The last one specifies read permission of ‘Resume’ for others.

The same in terms of semantics can be written as

Auth = {<mithilaesh, Resume.pdf, read>, <mithilaesh, Resume.pdf, write>,….}

In order to change the protection domain during run time we need the suid and sgid of the objects. In this case we have only the uid and gid therefore it does not support runtime protection domain changes.

```
mithilaesh@mithilaesh-HP-Notebook:~$ id mithilaesh
uid=1000(mithilaesh) gid=1000(mithilaesh) groups=1000(mithilaesh),4(adm),24(cdro
m),27(sudo),30(dip),46(plugdev),119(lpadmin),130(lxd),131(sambashare)
```