1. A common technique for inhibiting password guessing is to disable an account after three consecutive failed login attempts.

a. Discuss how this technique might prevent legitimate users from accessing the system. Why is this action a violation of the principle of lease common mechanism?

Answer:

The above technique for inhibiting password guessing might prevent legitimate users from accessing the system in following ways:

- Sometimes, users forget the passwords, and this may also happen three times in consecutive manner. A user usually feels trying different password is easier than to go through "forget password" process.
- Users would be having multiple accounts and it is less likely that they will remember username/password for all accounts. For example, a user has account for amazon, Facebook, google, etc. and each portal has its own requirement of password policies. It is very rare that the user will have single password for all accounts. In that case, user may try all password he remembers for the respective account and it is very likely that he may fail more than three times.
- Sometimes, users are so hurry that they type fast and try to get into the system as quickly as possible. In that case, if a single type error happens, it would go unnoticed as passwords are mostly not visible. This case may happen three times and lead to fail of access.
- There could be situation where one of the keys of keyboard is not working properly or not getting pressed properly while entering password, but the user is giving correct password. In that case, even though the password is correct, but it would land into to disabling account if he tries three times.
- Another example could be if an attacker gets the username and tries three times to get access. The system will disable the account after three unsuccessful fail attempts and when a legitimate user tries to login, he would not be able to login. The tells clearly that even though if there is legitimate user, he will not able to login as the attacker has already disabled the account by invalid login attempts.

The reasons why this action is violation of the principle of least common technique are as follows:

According to definition, the principle of least common mechanism states that mechanisms used to access resources should not be shared. In simple terms, the principle is restrictive because it limits sharing.
The above action allows sharing of a common portal/account/system among different users. The users can be attackers as well who are trying to gain unauthorized access to the system. As the common system/portal is shared, it also leads to sharing of common password policy, encryption algorithm between all the users. These are nothing but, access resources which are been shared across all the users which thereby violates the principle.

b. One can argue that this is an example of fail-safe defaults, because by blocking access to an account under attack, the system is defaulting to a known, safe state. Do you agree or disagree with this argument? Justify your answer.

I do agree with above argument.

According to definition, the principle of fail-safe defaults states that unless a subject is given an explicit access to an object, it should have denied access to the object. i.e. the default access to the object should be NONE.

In above situation, where an account is disabled after three consecutive failed login attempts, there is no other best safest measure according to the principle. The only best safe action is to disable the account in order to restrict the unauthorized access to the system. This simply means, by locking down the account to a default access as NONE/safe state in case of three consecutive failed login attempts, the system ensures that the account will remain uncompromised from further login attempts from attackers.

However, there could be chances where a legitimate user might have forgotten the password and goes into failed state by trying wrong passwords three consecutive times.

2. Consider Multics procedures *p* and *q*. Procedure *p* is running and needs to invoke procedure *q*. Procedure *q*'s access bracket is (5, 8) and its call bracket is (8, 11). Assume that *q*'s access control list gives *p* full (read, write, append, and execute) rights to *q*. In which ring(s) must *p* execute for the following to happen? Justify your answer.
a. *p* can invoke *q*, but a ring-crossing fault occurs.
b. *p* can invoke *q* provided that a valid gate is used as an entry point.
c. *p* cannot invoke *q*.
d. *p* can invoke *q* without any ring-crossing fault occurring, but not necessarily through a valid gate.

Answer:

Multics supports a scheme having a segment descriptor with below terms.

Access bracket: A pair of integers, a1 & a2 such that a1 < = a2

Limit: An integer a3, such that a3 > = a2

Lists of gates: Entry points at which segments.

Example:

Consider a example where a data or procedure wants to execute in ring r with an intention to access certain other procedure segment in the system. Therefore, each segment which needs to be accessed will have access bracket. The segment will also have a call bracket (c1, c2) with c1 <= c2.

The notation for example is (a1,a2,a3), then for call bracket (c1,c2)

c1 = a2

(a1,a2) is access bracket

(a2,a3) is call bracket.

Then, the rules are as follows for accessing a data segment:

• $r < a1$: access is permitted, but a ring-crossing fault occurs
• $a1 <= r <= a2$: all accesses permitted and no fault occurs
• $a2 < r <= a3$: access permitted if made through a valid gate
• $a3 < r$: all accesses denied

Considering above example, we can apply the same for the question and consider below things,

(5,8) is the access bracket

(8,11) is the call bracket

Then, following are the conclusions based on the rules

a. *p* can invoke *q*, but a ring-crossing fault occurs.
Answer:

Consider the below rule,
$r < a1$: access is permitted, but a ring-crossing fault occurs
where a1=5. Therefore, if p executes in any ring below 5, then the ring-crossing fault occurs.

b. *p* can invoke *q* provided that a valid gate is used as an entry point.
Answer:

Consider the below rule,
$a2 < r <= a3$: access permitted if made through a valid gate
Where, a2 =8 and a3 = 11. Therefore, if p executes in any ring between 8 and 11, then p can invoke q as these are valid gates and can be used as an entry point based on the rule.

c. *p* cannot invoke *q*.
Answer:

Consider the below rule,
$a3 < r$: all accesses denied,
where a3 = 11. Therefore, if p executes in any ring number above 11, then all the access will be denied.

d. *p* can invoke *q* without any ring-crossing fault occurring, but not necessarily through a valid gate.

Answer:

Consider the below rule,

$a1 <= r <= a2$: all accesses permitted, and no fault occurs,

where a1 = 5 and a2 = 8. Therefore, if p executes in any ring number between and equal to 5 to 8, then all access will be permitted, and no fault will occur.

3. A computer system provides protection using the Bell-LaPadula policy. How would a virus spread if

Answer:

Definition of virus:

A virus is a piece of program that intends to replicate itself by manipulating other files/programs with a focus to insert a code which will further do the same task [replicate itself]. There needs some human intervention in order for the virus to propagate itself.

Based on Bell-LaPadula policy , The policy is "No read up, No write down" which says that a subject(user) with clearance level lower than the classification level of the object cannot read that object and further subject(user) with clearance level higher than the classification level of the object cannot write into that object.The policy also majorly states that a subject can read an object if the subject dominates object and a subject can write into an object if object dominates subject

Specifically coming to the question,

a. the virus were place on the system at system low (the compartment dominated by all other compartments)? Justify your answer.

b. the virus were place on the system at system high (the compartment that dominates all other compartments)? Justify your answer.

Lets say if the above policy is applied in the given computer system, then

a. the virus were place on the system at system low (the compartment dominated by all other compartments.
From the Bell-LaPadula policy a subject can write into an object if the object dominates the subject. A subject with the lower most clearance level can write into all the objects in the system, from the property of the virus we have that it requires some type of human intervention and in the given case we have virus is placed in the lowest part on the system. Thus, the virus spreads in the system when the subject present in the lowest clearance level writes into any object in the system.

b. the virus were place on the system at system high (the compartment that dominates all other compartments)
Considering the Bell-LaPadula policy, a subject can read an object if the subject dominates object. A subject with the Highest most clearance level can read all the objects in the computer system, from the property of the virus we have that it requires some type of human intervention and in the given scenario we have virus is placed in the highest part on the system. Thus, the virus spreads in the system when the subject present in the highest clearance level reads any object in the system.

4. Classify the following vulnerabilities using the RISOS model. Assume that the classification is for the implementation level. Justify your answer.

a. The presence of the “wiz” command in the *sendmail* program (see section 20.2.8).

Type of vulnerability: Inadequate identification/authorization/authentication.
Justification:
The above category of flaws occur when a system allows a user to be erroneously identified, when one user can assume another’s privilege, or **when a user can trick the system into executing a program without authorization.** Here, wiz command is helping the attacker to gain access into the remote system with the help of SMTP agent. The access is provided without any authorization/authentication

b. The failure to handle the **IFS** shell variable by *loadmodule* (see section 20.2.8).

Type of vulnerability: Incomplete Parameter Validation

Justification:

Reading the section 20.2.8 for this case study, It says that loadmodule is a program that loads modules dynamically into a running kernel. Loadmodule uses system to execute ld.so to perform actual load. The critical information of the system is provided by the program arch/built in programs . An un authenticated user can manipulate above information by using built in library functions. Now, here it is the job of this library function to reset the environment variable. But at the same time, by leveraging the features of IFS – an attacker can manipulate above environment information by using this built in functions through system calls. So, we can say that IFS (the IFS is not completely validated to check if it is safer state or not before it calls the system) is giving a way to attacker here to exploit the system. This should be fixed by properly sanitizing environment in which that program executes before invoking any sub programs that are to be trusted.

c. The failure to select an Administrator password that was difficult to guess (see section 20.2.9).

Type of vulnerability: Exploitable logic error
Justification:

When a certain problems does not fall into any of the required category, it falls under such type of category. In the above case, the policy is designed in such a way that it fails to state that passwords if not strong can be easily guessed by attackers which may result in attacks.
As the Administrator is not enforced to set hard to guess password, there are chances that users fail to set hard to guess password this helps the attacker to easily guess the password of the Administrator and login and access various sensitive information.

Which implies that the policy on a whole fails to establish necessary rules for setting hard to guess passwords this failure is exploited by the attackers hence the above scenario seems to be having the flaw which falls into above category.