Test - 1

Tanusi Mittal
1901CS65

1. (D) None of These

⇒ 664 = rw- rw- r- - -

But sudo chmod u+s abc.txt

= rwS rw- r- - -

Now 'S' is no sort of a permission even though it is set-uid.

So, the answer is None of These.

2. (B) Only I is safe as Set-UID program

⇒ vi is not safe because of the usage of it. The vi command opens the vim editor in our terminal and hence is used by everyone including the unprivileged users/groups. which ~~too~~ can lead to security issues.

3. (A) Yes

⇒ If he converts it to set-UID, it will give him the permissions due to the attacks. Hence he will be able to read the file.

(4) (D) NULL

⇒ The execve() system call overwrites the current process's memory with the data provided by the new program, therefore all the environment variables stored inside the process are lost. If we want to pass its environment variables to new process we pass it through the execve in its third argument

So if we donot want to pass any we write NULL, execve(filename, arg, envp)

5) (C) Environ will always point to the updated set of environment variables

⇒ whenever some new variables are added, only the global variable environ changes but the envp always point to the original copy of the environment variables

6. (c) Carol Alice

→ We know, that when we change a shell variable our environment variable still remains the same
first command is used for environment variable so it will print Carol
And second is we printing the shell variable which is now updated. so it will print Alice.

7. (A) Static Linking
With dynamic linking a part of the code is undecided during compile time, the missing code is now decided during run time when user who might be untrusted are in control.
Therefore static is better.

8. Yes, I agree with Dev
We can use a guard in between buffer and frame pointer.
Stack based buffer overflow attacks need to modify the return address. If we ~~use~~ place a guard, then we can use this guard to detect whether the return address

is modified or not. This is also known as stack Guard Approach.

Before returning the return address from the function, we check whether value is modified or not. So now we just have to detect that if our guard is overwritten or not and can simply detect the buffer overflow.