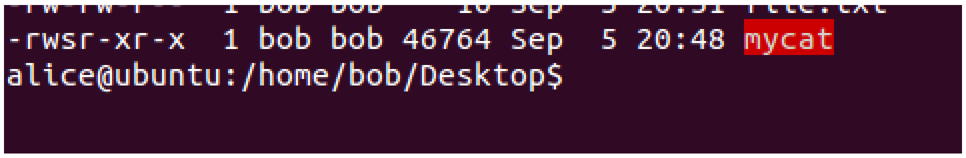
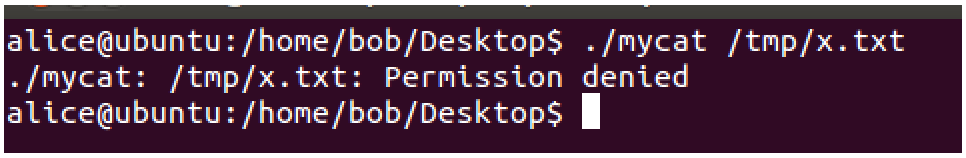
1.1. Alice runs a Set-UID program that is owned by Bob. The program tries to read from /tmp/x, which is readable to Alice, but not to anybody else. Can this program successfully read from the file?



```
-rw-rw-r-- 1 bob bob    10 Sep  5 20:31 file.txt
-rwsr-xr-x  1 bob bob 46764 Sep  5 20:48 mycat
alice@ubuntu:/home/bob/Desktop$
```



```
alice@ubuntu:/home/bob/Desktop$ ./mycat /tmp/x.txt
./mycat: /tmp/x.txt: Permission denied
alice@ubuntu:/home/bob/Desktop$ 
```

The program owned by Bob with Set-UID cannot read /tmp/x as the set-UID runs the program with the privilege of the owner, i.e., Bob. As the file is set to be read-only by the owner, i.e. Alice, it cannot be read by Bob's privilege.

1.2. A process tries to open a file for read. The process's effective user ID is 1000, and the real user ID is 2000. The file is readable to user ID 2000, but not to user ID 1000. Can this process successfully open the file?

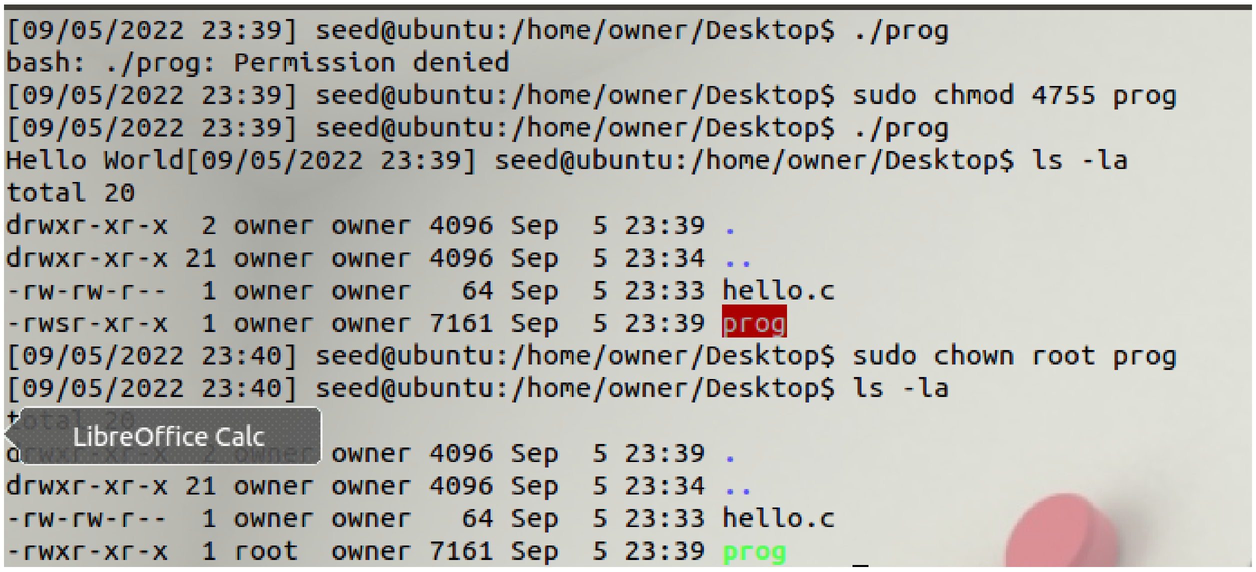
The process's effective user ID is used to access the file. Since , The effective user ID is 1000 which cannot read the file. Hence, The process cannot open this file.

1.3. A root-owned Set-UID program allows a normal user to gain the root privilege while executing the program. What prevents the user from doing bad things using the privilege?

The set-UID programs allows the user to gain root privileges only for that particular program, so the user won't have enough privileges to do bad things.
The program is isolated as mush as possible so that the attacker may not be able to do malicious activities.The code and data are separated as much as possible so that the user may perform such malicious activities.

1.4. We are trying to turn a program prog owned by the seed user into a Set-UID program that is owned by root. Can running the following commands achieve the goal?
$ sudo chmod 4755 prog
$ sudo chown root prog

Runnig the above changes the ownership of the file from seed user to that of root, But the Set-UID bit that was set is removed and the default file permissions are added.



```
[09/05/2022 23:39] seed@ubuntu:/home/owner/Desktop$ ./prog
bash: ./prog: Permission denied
[09/05/2022 23:39] seed@ubuntu:/home/owner/Desktop$ sudo chmod 4755 prog
[09/05/2022 23:39] seed@ubuntu:/home/owner/Desktop$ ./prog
Hello World[09/05/2022 23:39] seed@ubuntu:/home/owner/Desktop$ ls -la
total 20
drwxr-xr-x  2 owner owner 4096 Sep  5 23:39 .
drwxr-xr-x 21 owner owner 4096 Sep  5 23:34 ..
-rw-rw-r--  1 owner owner   64 Sep  5 23:33 hello.c
-rwsr-xr-x  1 owner owner 7161 Sep  5 23:39 prog
[09/05/2022 23:40] seed@ubuntu:/home/owner/Desktop$ sudo chown root prog
[09/05/2022 23:40] seed@ubuntu:/home/owner/Desktop$ ls -la
total 20
drwxr-xr-x  2 owner owner 4096 Sep  5 23:39 .
drwxr-xr-x 21 owner owner 4096 Sep  5 23:34 ..
-rw-rw-r--  1 owner owner   64 Sep  5 23:33 hello.c
-rwxr-xr-x  1 root  owner 7161 Sep  5 23:39 prog
```

1.5. The chown command automatically disables the Set-UID bit, when it changes the owner of a Set-UID program. Please explain why it does that.

The chown command automatically disable Set-UID bit when it changes ths owner of a Set-UID program because the Set-UID bit allows a program to run with the privileges of the owner as it was decided by the previous owner. Here, Since the owner is changed , leaving the Set-UID bit set allows anyone to access the file as new owner which may have unintended consequences for new owner.

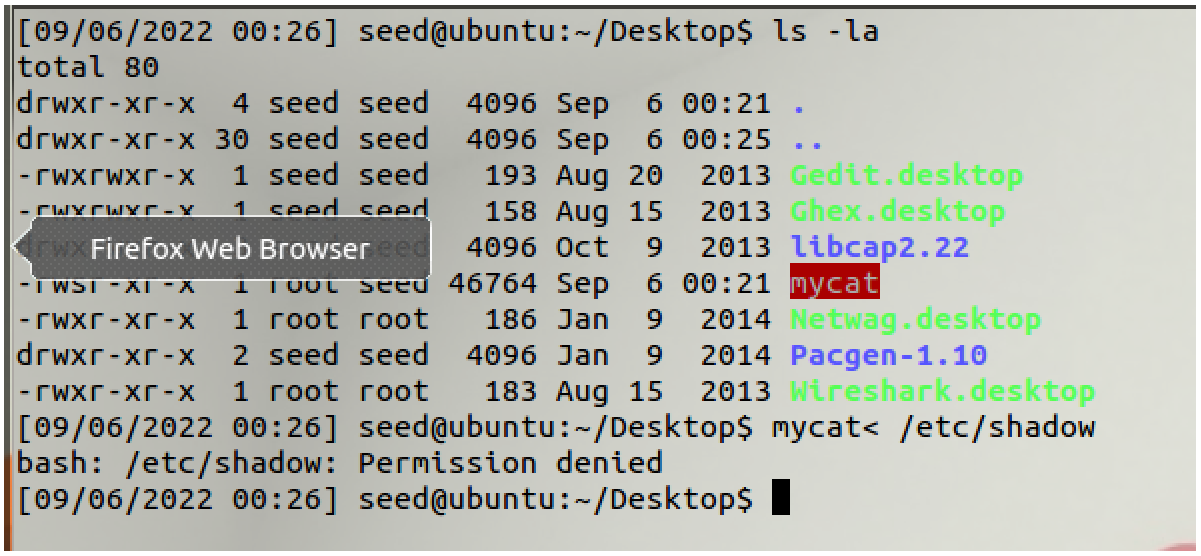
1.6. When we debug a program, we can change the program's internal variables during the execution. This can change a program's behavior. Can we use this technique to debug a Set-UID program and change its behavior? For example, if the program is supposed to open the /tmp/xyz file, we can modify the filename string, so the Set-UID program ends up opening /etc/passwd.
The debugger program is a program that runs with roots privileges. So , A user cannot change the change the behaviour of a Set-UID program using a debugger as user does not have enough privileges to modify filename string.

1.7. Both system() and execve() can be used to execute external programs. Why is system() unsafe while execve() is safe?
System () calls a fork() command to run child process that runs the /bin/sh internally.while giving the input to such program the attacker can attached more commands with intended input like “a;rm -rf *;” . since it is executed in new child shell, the “rm -rf *” command will be executed too deleting the files.

execve() replace the current process image with a new process image.Code and data are clearly separated here so the attacker can use the above attack here.

1.8. When a program takes an input from users, we can redirect the input device, so the program can take the input from a file. For example, we can use prog < myfile to provide the data from myfile as input to the prog program. Now, if prog is a root-owned Set-UID program, can we use the following method to to get this privileged program to read from the /etc/shadow file? $ prog < /etc/shadow

Here the /etc/shadow file is a privileged program. So even if the file prog is a Set-UID prograit can’t take input from the /etc/shadow file



```
[09/06/2022 00:26] seed@ubuntu:~/Desktop$ ls -la
total 80
drwxr-xr-x  4 seed seed  4096 Sep  6 00:21 .
drwxr-xr-x 30 seed seed  4096 Sep  6 00:25 ..
-rwxrwxr-x  1 seed seed   193 Aug 20  2013 Gedit.desktop
-rwxrwxr-x  1 seed seed   158 Aug 15  2013 Ghex.desktop
drwx[illegible]eed  4096 Oct  9  2013 libcap2.22
-rwsr-xr-x  1 root seed 46764 Sep  6 00:21 mycat
-rwxr-xr-x  1 root root   186 Jan  9  2014 Netwag.desktop
drwxr-xr-x  2 seed seed  4096 Jan  9  2014 Pacgen-1.10
-rwxr-xr-x  1 root root   183 Aug 15  2013 Wireshark.desktop
[09/06/2022 00:26] seed@ubuntu:~/Desktop$ mycat< /etc/shadow
bash: /etc/shadow: Permission denied
[09/06/2022 00:26] seed@ubuntu:~/Desktop$
```
.

1.9. When a parent Set-UID process (effective user ID is root, and the real user ID is bob) creates a child process using fork(), the standard input, output, and error devices of the parent will be inherited by the child. If the child process drops its root privilege, it still retains the access right to these devices. This seems to be a capability leaking, similar to what we covered in Chapter 1.4.4. Can this pose any danger?
The parent process of the child process was executed by the user i.e. bob. While the child process is create using fork(), The original parent process privileges i.e. the user’s privilege will be inherited by the child. Hence the child process won’t pose any danger.

1.10. ★
The superuser wants to give Alice a permission to view all the files in the system using the more command. He does the following:

```
$ cp /bin/more /tmp/mymore
$ sudo chown root /tmp/mymore
$ sudo chmod 4700 /tmp/mymore
```

Basically, the above commands turns /tmp/mymore into a Set-UID program. Right now, because the permission is set to 4700, other users cannot execute the program. The superuser uses another command (now shown) to grant the execution permission only to Alice. We are not assuming that Alice is completely trusted. It is OK if Alice can only read other people's files, but it is not OK if Alice can gain any privilege beyond that, such as writing to other people's files. Please read the manual of the more program and find out what Alice can do to gain more privilege.

More program has a parameter as:

- v

Drop into the vi editor at the current line of the current file.
This allows the user to open the vim editor. Since the it is Set-UID program Alice is given the root privilege while using vim editor and can gain more privilege than intended.

1.11 Assume that you have a file that you would allow other users to read, only if a user's ID is smaller than 1000. Please describe how you can actually achieve this.

The user ID of all the user's is stored in /etc/passwd file. A group can be created with the help of /etc/passwd in which all users with user ID smaller than 1000 are added. Then the read permission may ge given to the group.

1.12. Sam found a very useful web page, which contains links to many interesting papers. He wants to download those papers. Instead of clicking on each of the links, he wrote a program that parses a HTML web page, get the papers URLs from the web page, and then use a program called wget to fetch each identified URL. The following is the code Snippet:

```
char command[100];
char* line, url;
line = getNextLine(file);// Read in one line from the HTML file.
while (line != NULL) {
// Parse the line to get a URL string.
url = parseURL (line);
if (url != NULL){
// construct a command, and execute it
sprintf(command, "%s %s", "wget", url);
system(command);
}
line = GetNextLine(file);
}
```

The function sprintf() is quite similar to printf(), except that sprintf() puts the output in a buffer pointed by the first argument, while printf() sends the output to the display. Please be noted that the functions getNextLine() and parseURL() are also implemented by Sam (their code is not displayed here). The program wget is a command-line program in Unix that can be used to download web files from a given URL.
The owner of the web page knows what Sam is doing with his page; he wants to attack Sams program. He knows the code above, but he does not know how Sam implements GetNextLine() or ParseURL(), but he suspects that Sam may make some mistakes there. (1) If you are the attacker, please describe how you plan to attack. (2) How do you fix the problem?

Here Sam is making use of system(command). We may use command injection to make use of it's vulnerability.

If we change the command that we want to run on sam's machine and make it look like an url, the system command may run the command, and the attack may be sucessful.

For eg. we can modify "www.google.com/hompage" to look like "www.google.com/homepage;rm -rf/" here the parseURL may treat the following text as url and pass it to system(). system() treats the text after ";" as a separate command and runs it in child process.So it will executes rm -rf/ deleting the files.

To prevent the attack using the program I would implement the following steps.

- Replacing system() with execve() since it replace the current process image with a new process image.So, this attack won't work.
- Not running the program in privilege mode unnecessarily.
- Not downloading data from potentially malicious website.
- Make edit in ParseURL() so that it can differentiate between proper url and modified ones.