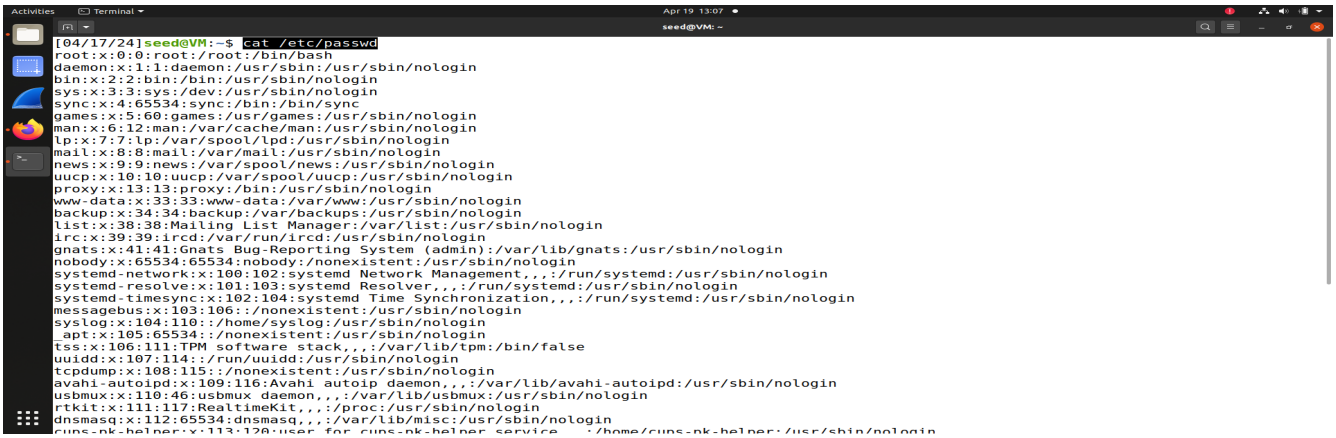
CSC 5290 : Cyber Security Practices

Lab 6:Environment Variables and SetUID

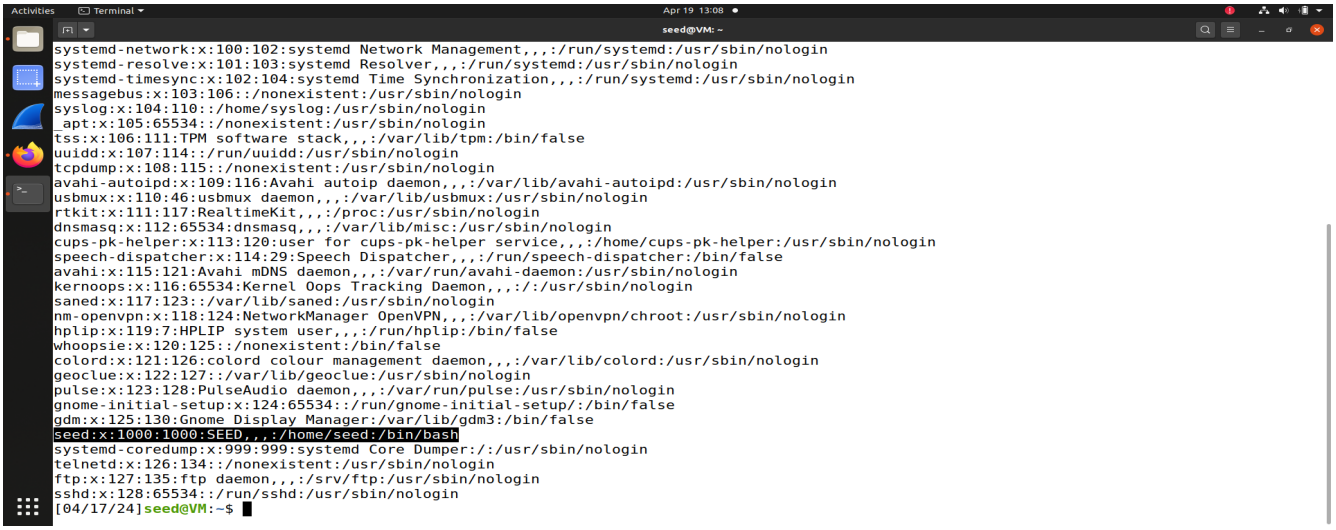
Winter 2024

Student name: Kanchan Chopde

Task 1: Manipulating Environment Variables : We use command cat /etc/passwd to check if we are using Bash in the seed account.

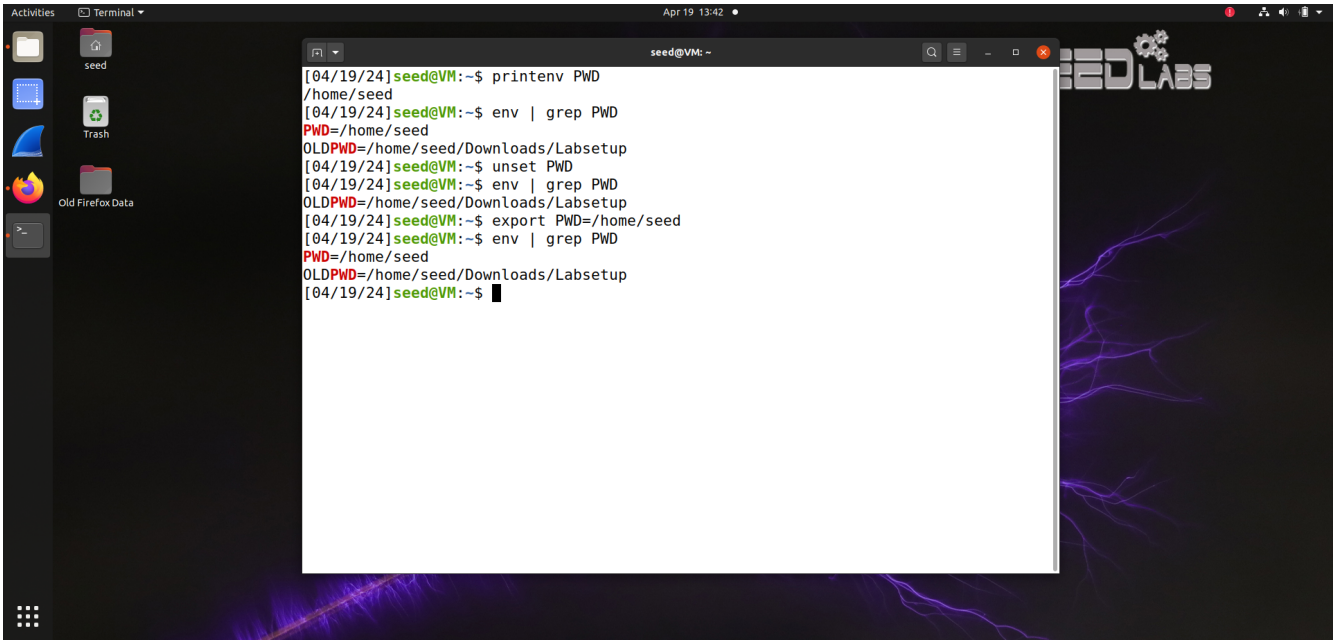


Here we can see seed account:



Now using printenv and env we print out environment varibles.

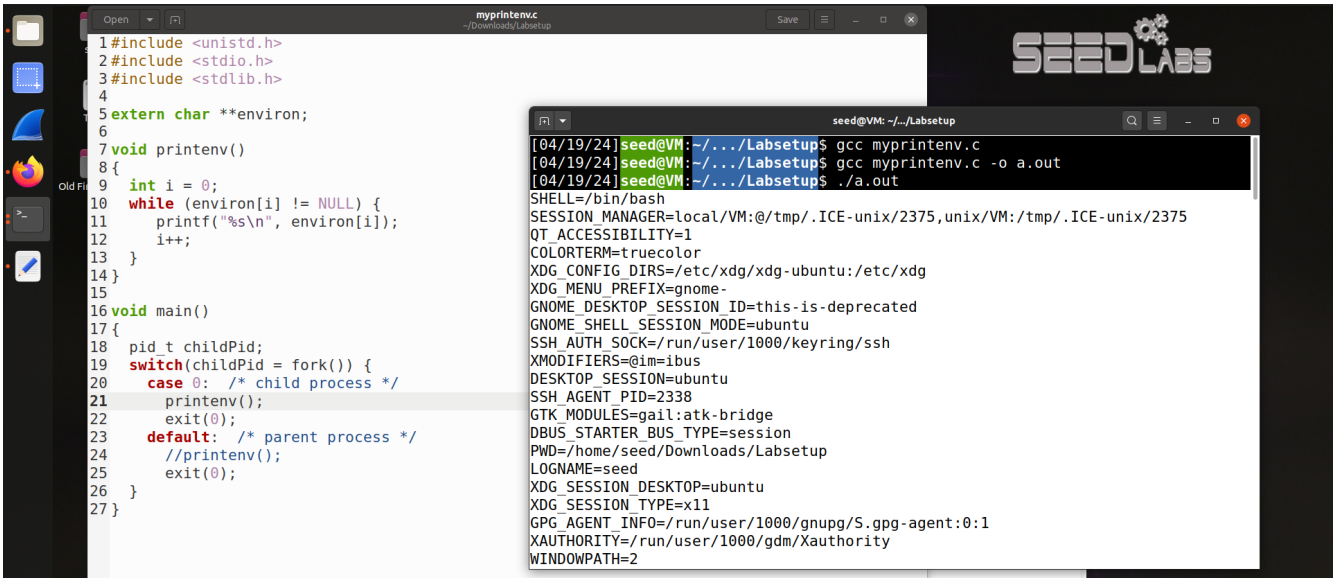
For some particular environment variables, such as PWD, we can use "printenv PWD" or



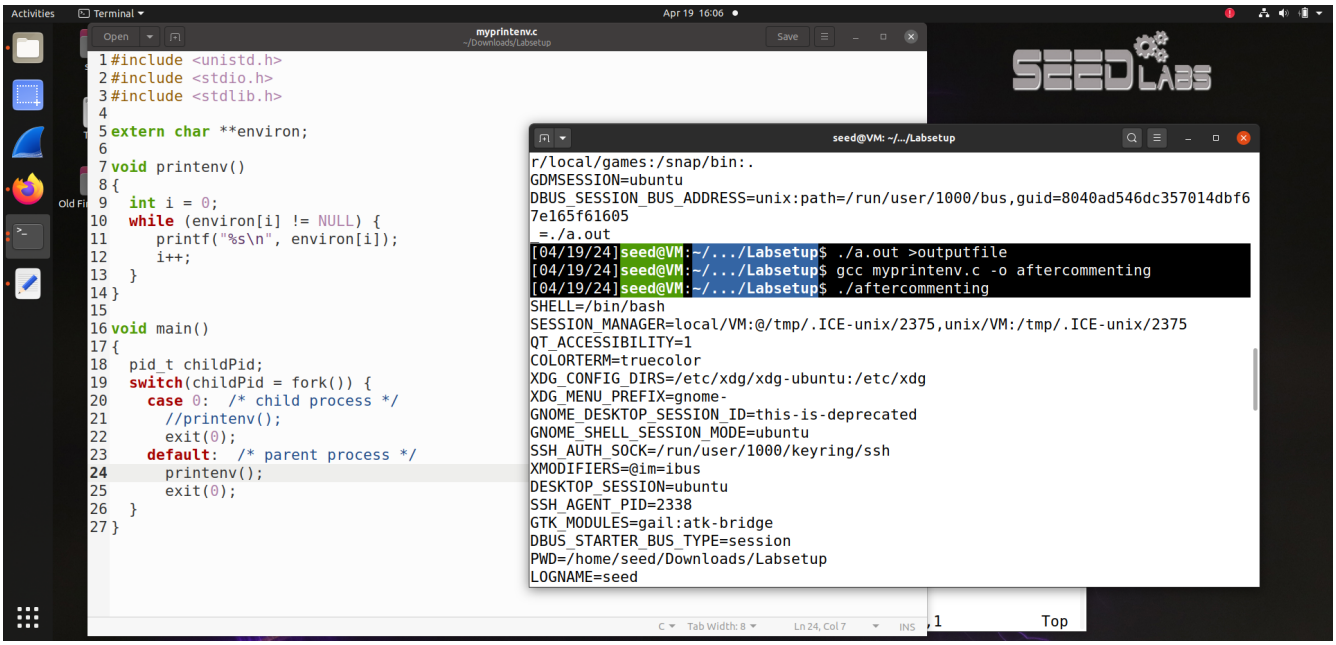
"env | grep PWD".
We can use unset PWD to remove the environment variable as seen in output. Then using export command we can set the variable and value ie. PWD=/home/seed to set the value . It can create and edit a particular variable.

Task 2: Passing Environment Variables from Parent Process to Child Process

The content of the output of myprintenv.c containing child process with printenv is stored in file named a.out file . It displays all the environment variables of the child process.



Output of after commenting child process is store in aftercommenting file.

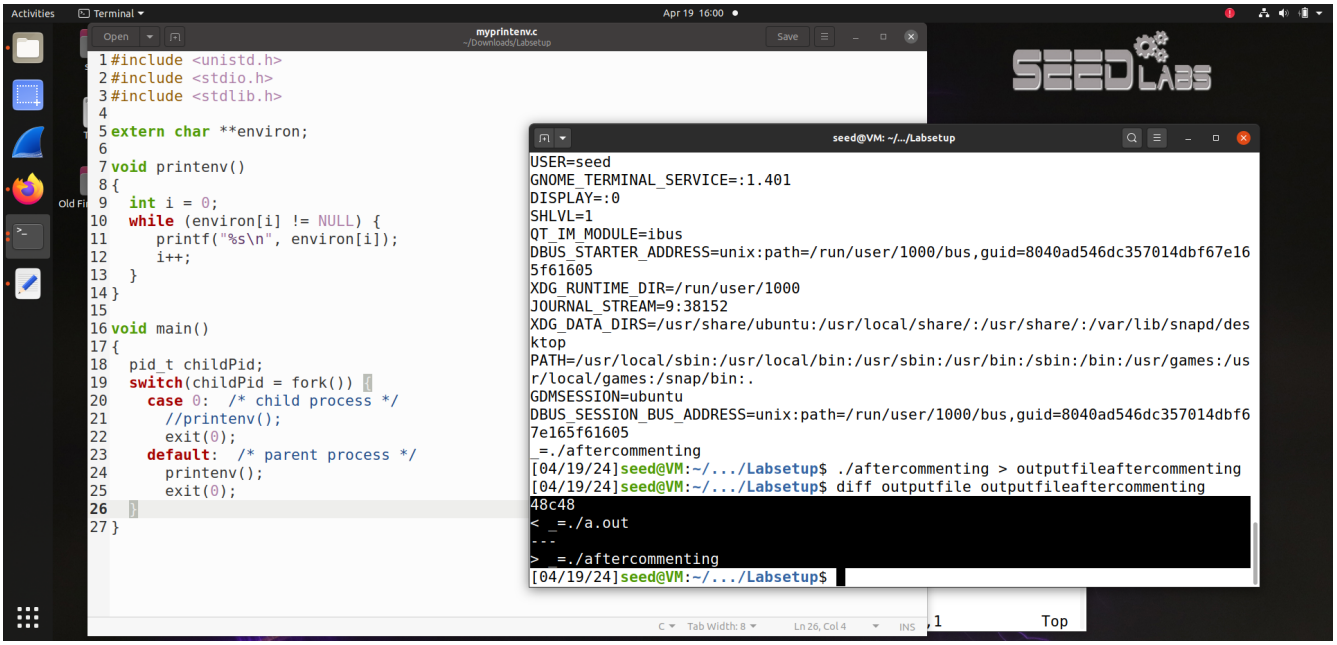


Now we use diff command to see the difference between a.out and aftercommenting file.

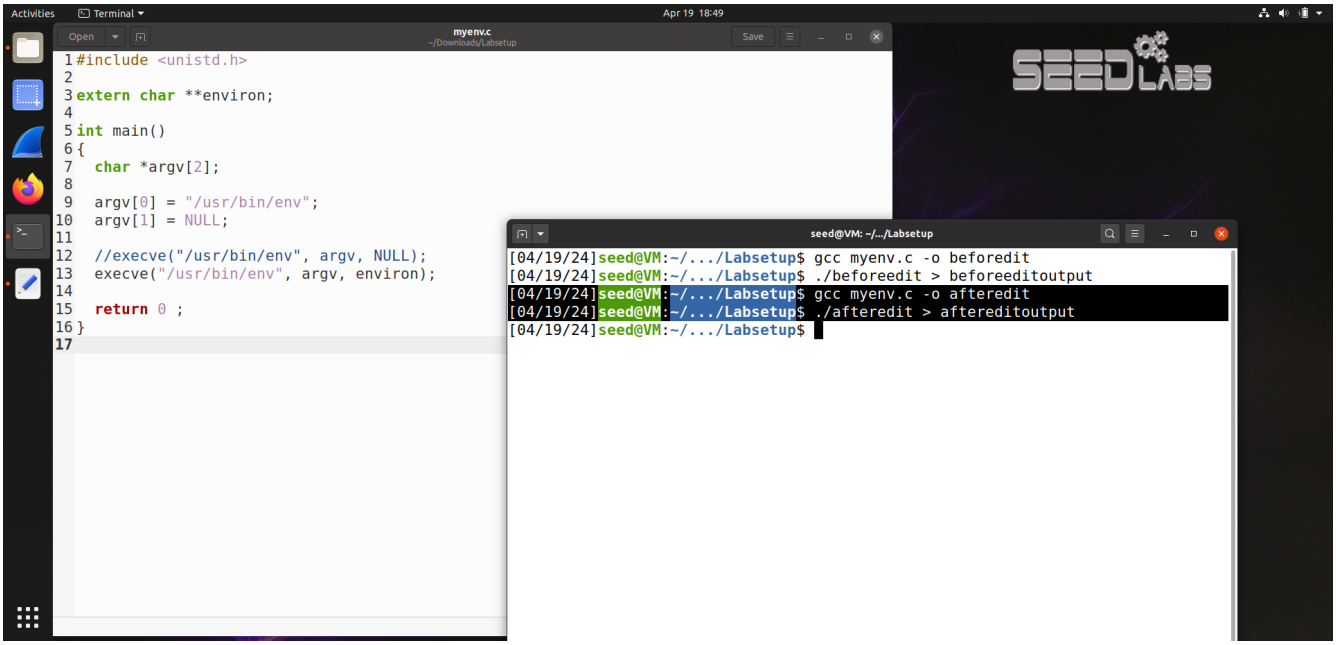
The output is interpreted as follows: 48c48 means that in the 48th line (left) in left file is changed to the 48th line (right) in the right file, where c stands for changing and the left and right numbers indicate the line number.

The < denotes lines in the left file and > indicates in the right file showing the changed content. This shows that the _ environment variable takes on the value of the last command executed, here the command of program execution. It is considered a special shell variable and contains different values

depending on the scenario. This shows that the _ environment variable changed depending on the compiled program being run but other than that there is no change in the environment variables. If both the programs were compiled into a file with the same name, there would not be any difference between the output of the parent and child process



Task 3: Environment Variables and execve()



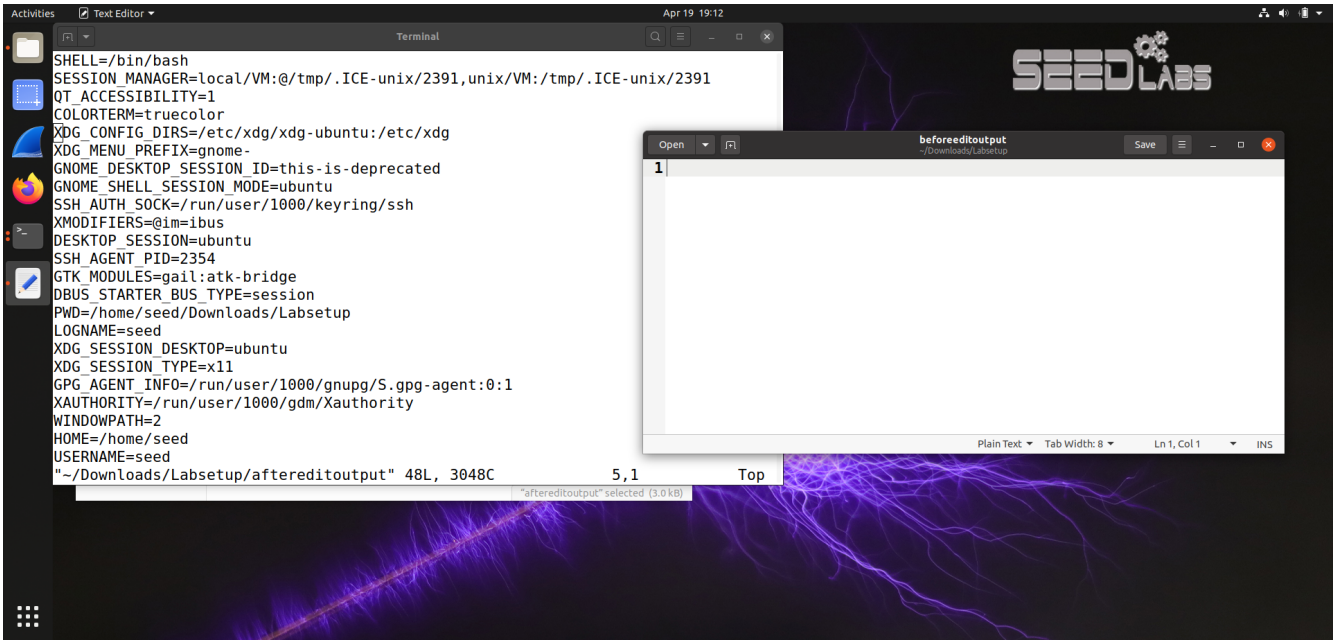
Here, as seen, the Task 3 program is compiled and executed into respective output files and the output is stored in beforeeditoutput (with NULL as the argument) and aftereditoutput (with environ as the argument).

Output: even though the global environ variable was specified in the program, the beforeedit program contained NULL as the third argument of the execve and the afteredit program contained environ variable as the third argument of the execve.

This change affected the output of the program because the third argument to execve() function specifies the environment variable of the current process. Since the environ variable was not passed in the initial program and hence no environment variables were associated with this new process, the output was null.

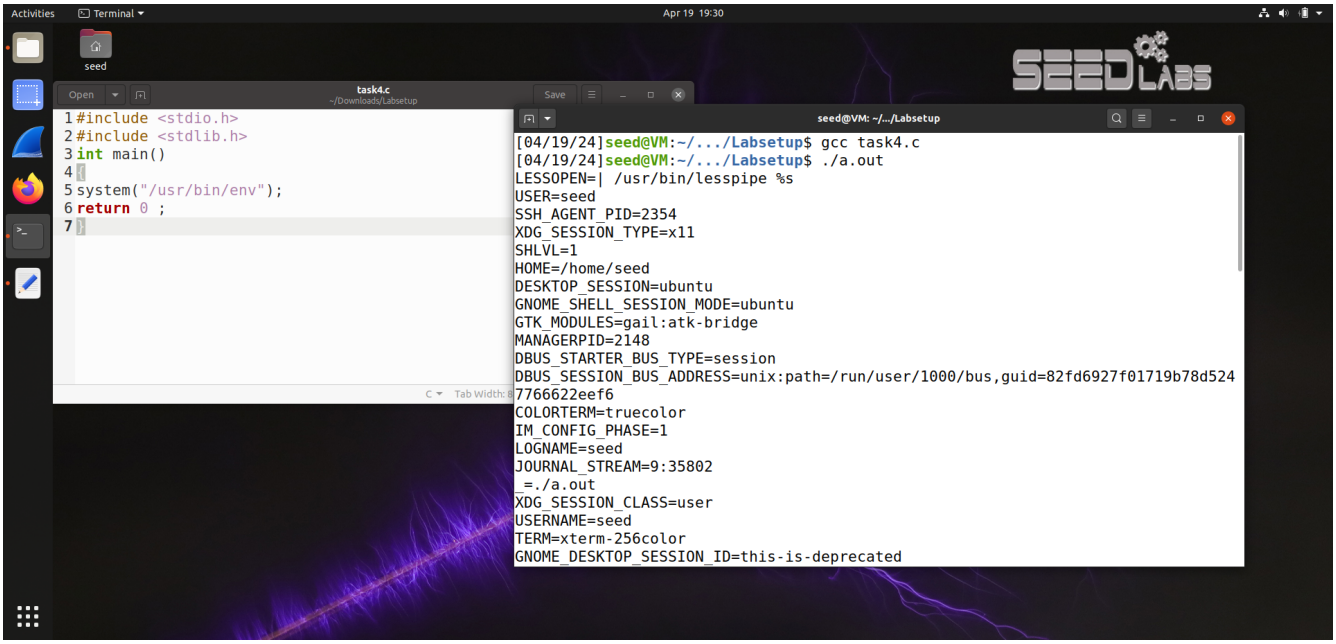
But after editing the program, we passed the environ variable as the third argument to execve, which contained all the environment variables of the current process, the output of the program had all the environment variables, as expected.

In conclusion, the third argument of the execve() command gets the program its environment variables.

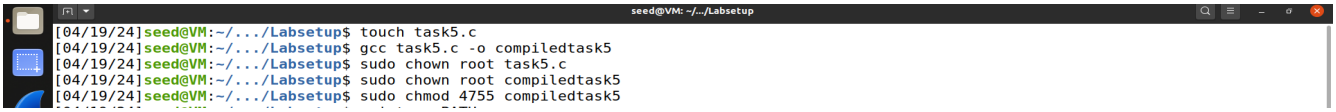

Activities
Text Editor
Apr 19 19:12
Terminal
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2391,unix/VM:/tmp/.ICE-unix/2391
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2354
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
"~/Downloads/Labsetup/aftereditoutput" 48L, 3048C 5,1 Top
"aftereditoutput" selected (3.0 kB)
Open
beforeeditoutput
~/Downloads/Labsetup
Save
1
Plain Text
Tab Width: 8
Ln 1, Col 1
INS
SEED LABS

Task 4: Environment Variables and system()

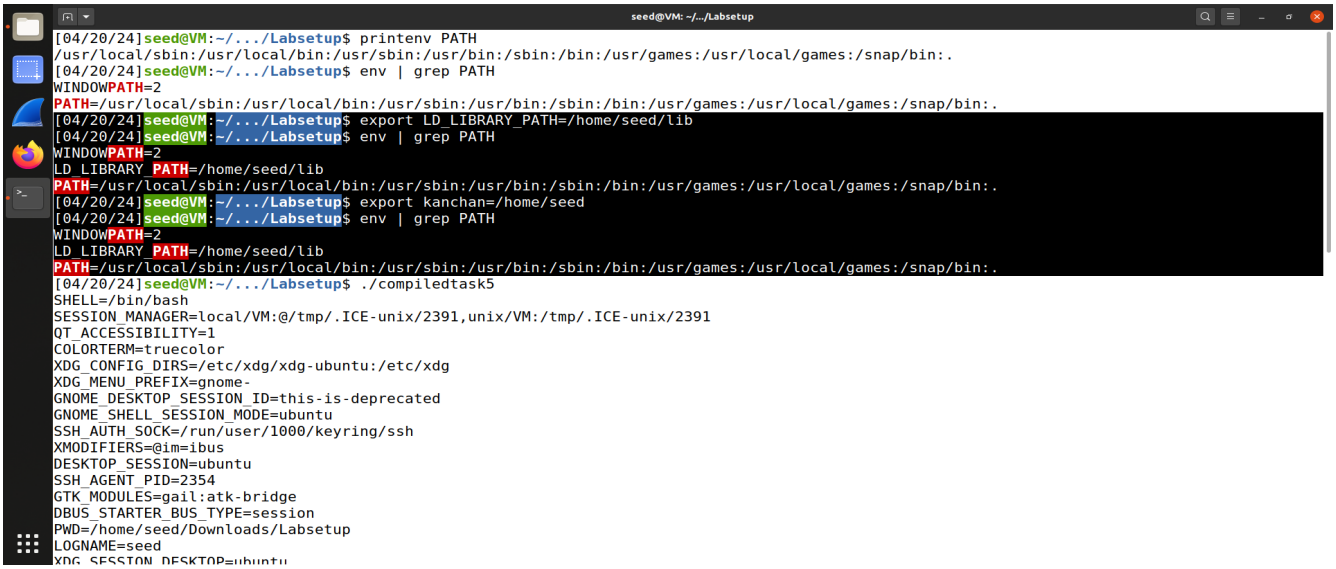
The program is compiled and executed , even though we don’t explicitly send any environment variables in the program, the output shows the environment variable of the current process. This happens because the system function in code implicitly passes the environment variables to the called function /bin/sh.



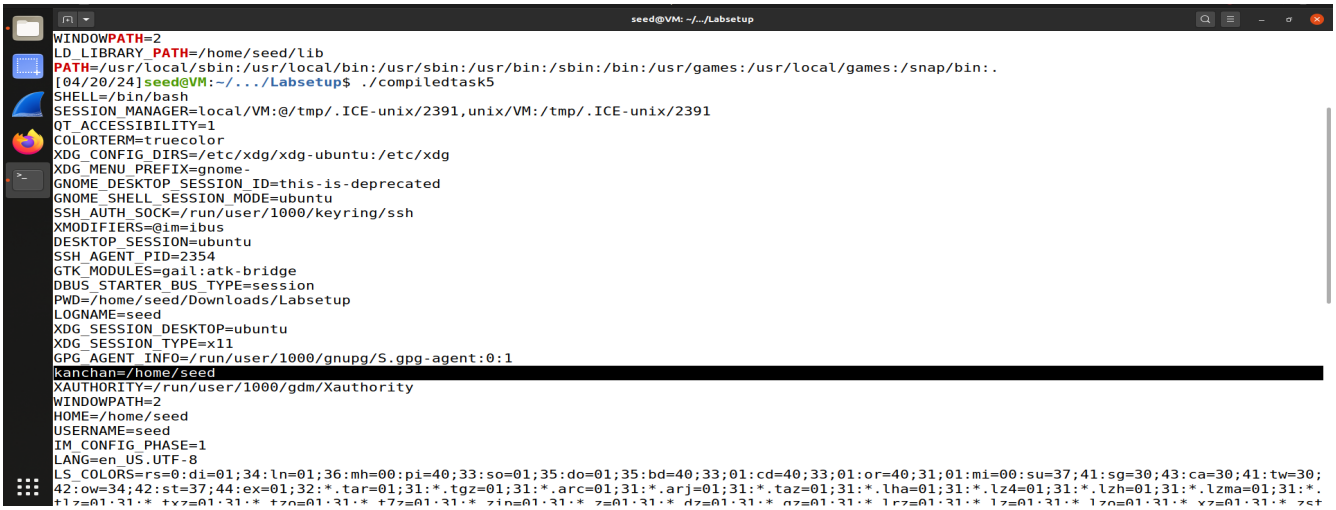
Task 5: Environment Variable and Set-UID Programs After compiling the given program, we change the ownership and permission of the file using the following commands: sudo chown root filename (making the root as the owner of filename) sudo chmod 4755 filename (making the program a SET-UID program by setting set-uid bit) This makes the program a SET-UID root program.



Then on looking for the environment variables, since PATH is already present, I initialized the new variable with name LD_LIBRARY_PATH with value /*home*/seed/lib and kanchan and value /home/seed using export command and allow the other environment values to be the same.



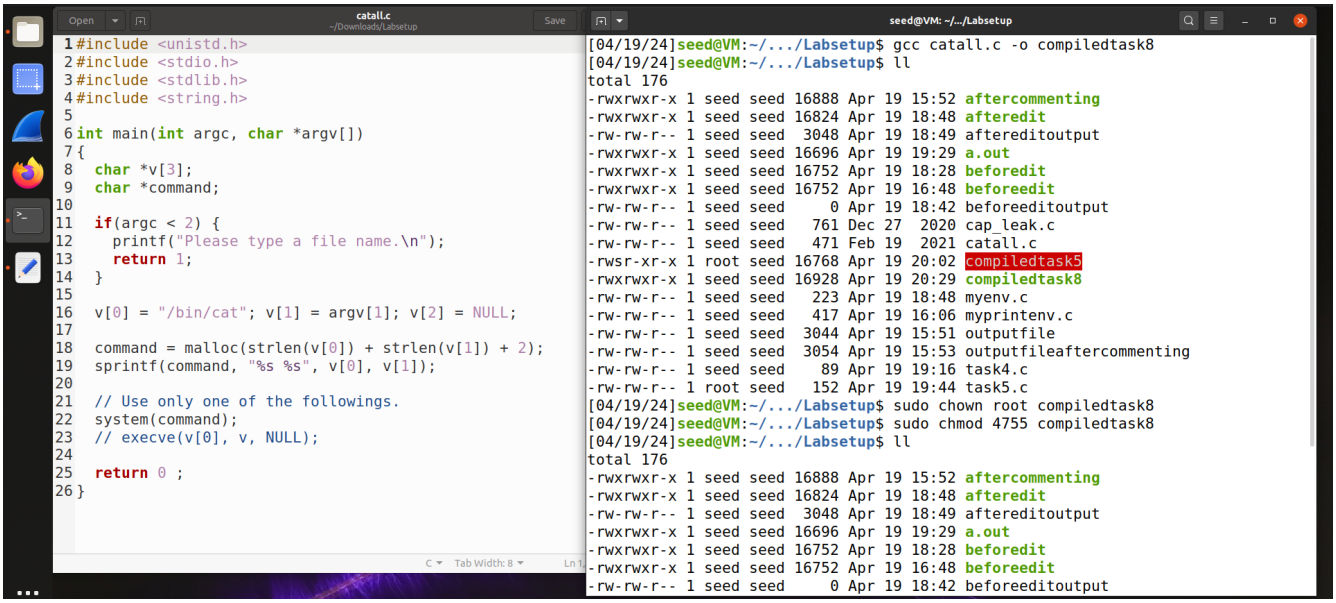
On compiling and running the program ,it's seen that the child process inherits the PATH and kanchan environment variable but there is no LD environment variable, as can be seen in the screenshots.



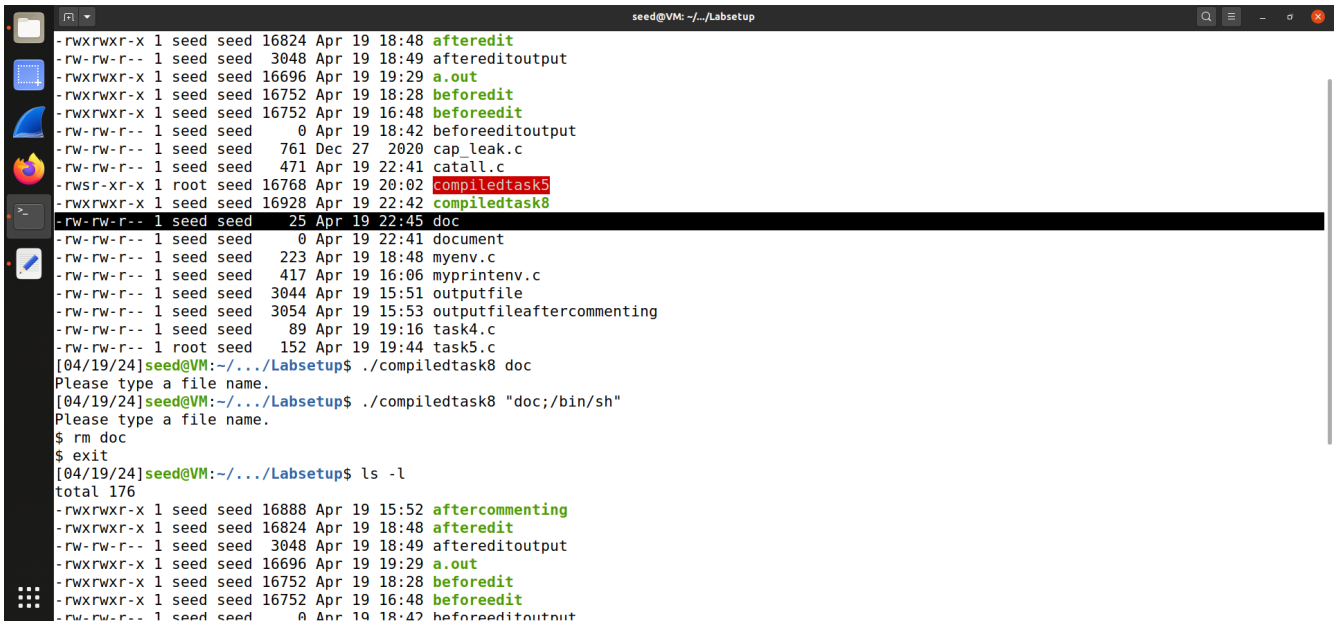
This shows that the SET-UID program's child process may not inherit all the environment variables of the parent process, LD_LIBRARY_PATH being one of them over here. This is a security mechanism implemented by the dynamic linker. The LD_LIBRARY_PATH is ignored here because the real user id and effective user id is different. That is why only the other two environment variables are seen in the output.

Task 8: Invoking External Programs Using system() versus execve()

Compile the program provided into a file named compiledtask8. Next, this file is converted into a root-owned SET-UID program with executable permission to other users:



On running this program, the normal functionality will output the contents of the file specified doc:

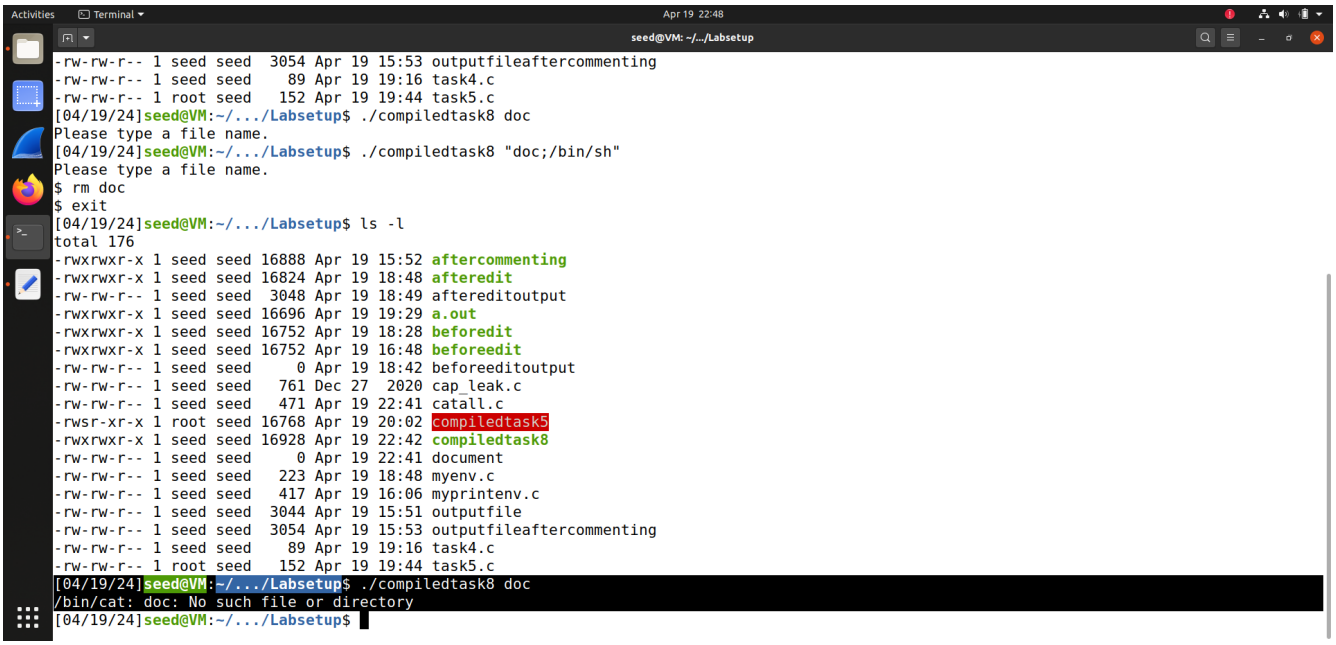


Next, consider that Bob is using the seed user account (Treating Bob as others (normal user)). Here, as we can see the program runs normally when we just provide the file to be read.

But, if we provide a malicious input such as “doc;/bin/sh”, here the program will first read the contents of the document and then run /bin/sh as a command (according to the program.) The /bin/sh allows Bob to run the shell program which has root privileges and bob then runs the “rm doc” command to remove a file on which it did not have the write permission. The root terminal is indicated by the $.

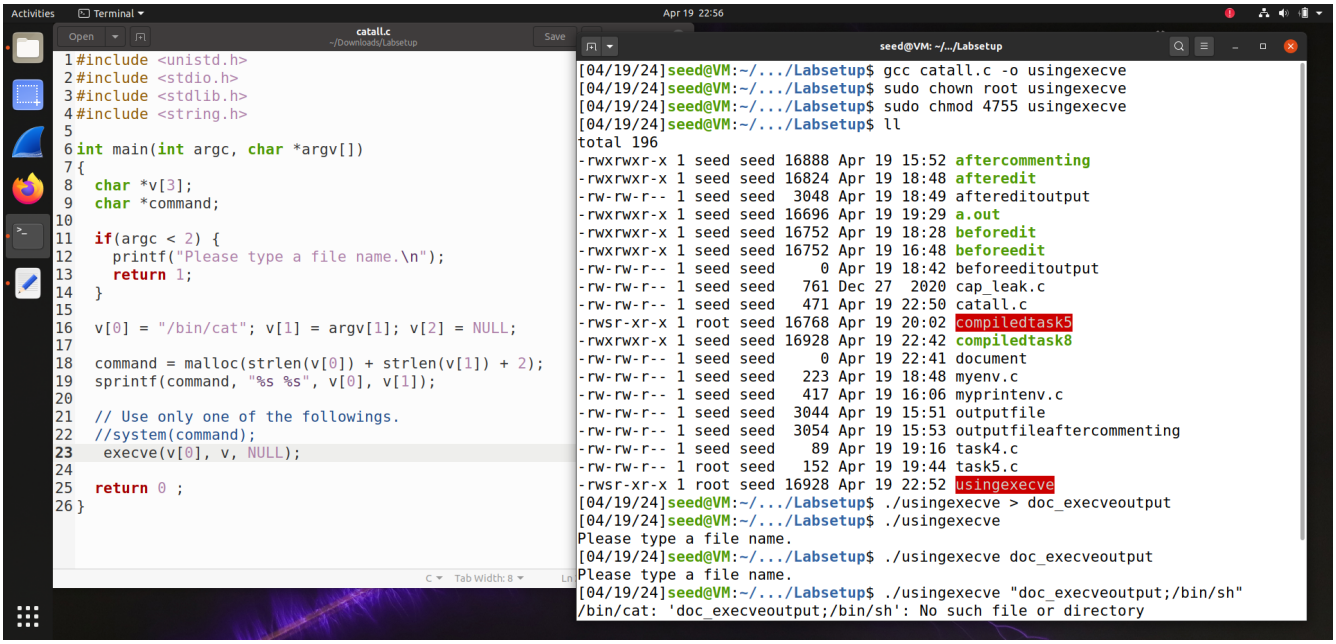
This shows that even though Bob did not have any permission to write, it could remove a file easily by assuming the privileges of the root user.

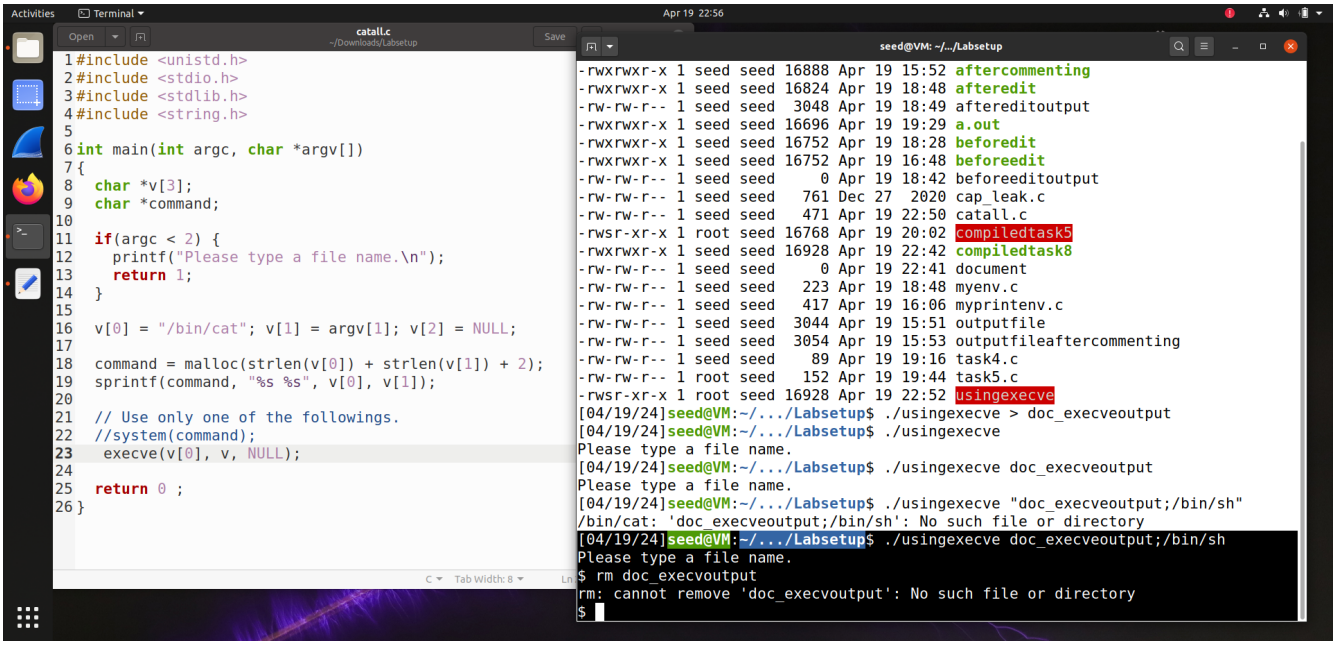
The problem here is the system call inside the program which does not separate the command and user input. The user input is eventually treated as a command instead of document name.

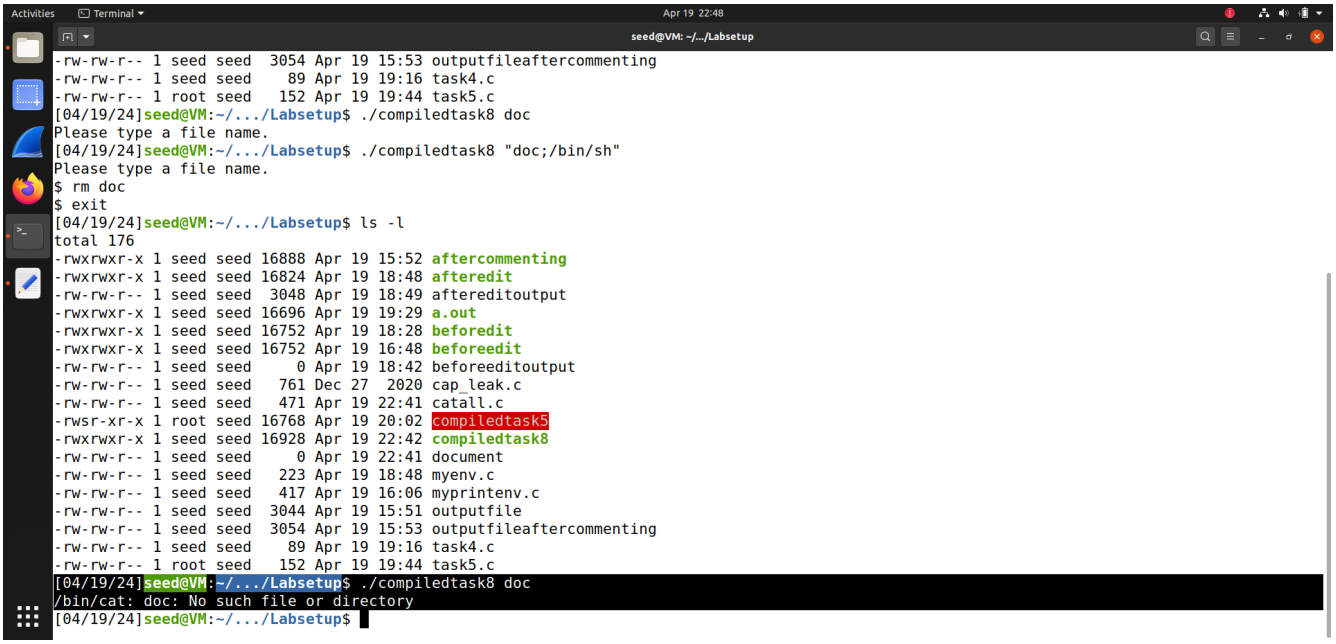


Now Commenting out the system(command) statement, and uncommenting the execve() statement; the program will use execve() to invoke the command.

We again try to perform the same attack and see that it fails because the entire user inputted string is considered as a file name rather than separating the string on ‘;’ as document name and command as before. Also, if a user forgets the quotes and just types in the string, the terminal of the same user is opened and not of the root user, hence Bob will not have the permission to write.


Activities
Terminal
Apr 19 22:56
Open
catall.c
~/Downloads/Labsetup
Save
1#include <unistd.h>
2#include <stdio.h>
3#include <stdlib.h>
4#include <string.h>
5
6int main(int argc, char *argv[])
7{
8 char *v[3];
9 char *command;
10
11 if(argc < 2) {
12 printf("Please type a file name.\n");
13 return 1;
14 }
15
16 v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
17
18 command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
19 sprintf(command, "%s %s", v[0], v[1]);
20
21 // Use only one of the followings.
22 //system(command);
23 execve(v[0], v, NULL);
24
25 return 0 ;
26}
C
Tab Width: 8
seed@VM: ~/.../Labsetup
[04/19/24]seed@VM:~/.../Labsetup$ gcc catall.c -o usingexecve
[04/19/24]seed@VM:~/.../Labsetup$ sudo chown root usingexecve
[04/19/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 usingexecve
[04/19/24]seed@VM:~/.../Labsetup$ ll
total 196
-rwxrwxr-x 1 seed seed 16888 Apr 19 15:52 aftercommenting
-rwxrwxr-x 1 seed seed 16824 Apr 19 18:48 afteredit
-rw-rw-r-- 1 seed seed 3048 Apr 19 18:49 aftereditoutput
-rwxrwxr-x 1 seed seed 16696 Apr 19 19:29 a.out
-rwxrwxr-x 1 seed seed 16752 Apr 19 18:28 beforedit
-rwxrwxr-x 1 seed seed 16752 Apr 19 16:48 beforeedit
-rw-rw-r-- 1 seed seed 0 Apr 19 18:42 beforeeditoutput
-rw-rw-r-- 1 seed seed 761 Dec 27 2020 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Apr 19 22:50 catall.c
-rwsr-xr-x 1 root seed 16768 Apr 19 20:02 compiledtask5
-rwxrwxr-x 1 seed seed 16928 Apr 19 22:42 compiledtask8
-rw-rw-r-- 1 seed seed 0 Apr 19 22:41 document
-rw-rw-r-- 1 seed seed 223 Apr 19 18:48 myenv.c
-rw-rw-r-- 1 seed seed 417 Apr 19 16:06 myprintenv.c
-rw-rw-r-- 1 seed seed 3044 Apr 19 15:51 outputfile
-rw-rw-r-- 1 seed seed 3054 Apr 19 15:53 outputfileaftercommenting
-rw-rw-r-- 1 seed seed 89 Apr 19 19:16 task4.c
-rw-rw-r-- 1 root seed 152 Apr 19 19:44 task5.c
-rwsr-xr-x 1 root seed 16928 Apr 19 22:52 usingexecve
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve > doc_execveoutput
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve
Please type a file name.
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve doc_execveoutput
Please type a file name.
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve "doc_execveoutput;/bin/sh"
/bin/cat: 'doc_execveoutput;/bin/sh': No such file or directory


Activities
Terminal
Apr 19 22:56
Open
catall.c
~/Downloads/Labsetup
Save
1#include <unistd.h>
2#include <stdio.h>
3#include <stdlib.h>
4#include <string.h>
5
6int main(int argc, char *argv[])
7{
8 char *v[3];
9 char *command;
10
11 if(argc < 2) {
12 printf("Please type a file name.\n");
13 return 1;
14 }
15
16 v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
17
18 command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
19 sprintf(command, "%s %s", v[0], v[1]);
20
21 // Use only one of the followings.
22 //system(command);
23 execve(v[0], v, NULL);
24
25 return 0 ;
26}
C
Tab Width: 8
seed@VM: ~/.../Labsetup
-rwxrwxr-x 1 seed seed 16888 Apr 19 15:52 aftercommenting
-rwxrwxr-x 1 seed seed 16824 Apr 19 18:48 afteredit
-rw-rw-r-- 1 seed seed 3048 Apr 19 18:49 aftereditoutput
-rwxrwxr-x 1 seed seed 16696 Apr 19 19:29 a.out
-rwxrwxr-x 1 seed seed 16752 Apr 19 18:28 beforedit
-rwxrwxr-x 1 seed seed 16752 Apr 19 16:48 beforeedit
-rw-rw-r-- 1 seed seed 0 Apr 19 18:42 beforeeditoutput
-rw-rw-r-- 1 seed seed 761 Dec 27 2020 cap_leak.c
-rw-rw-r-- 1 seed seed 471 Apr 19 22:50 catall.c
-rwsr-xr-x 1 root seed 16768 Apr 19 20:02 compiledtask5
-rwxrwxr-x 1 seed seed 16928 Apr 19 22:42 compiledtask8
-rw-rw-r-- 1 seed seed 0 Apr 19 22:41 document
-rw-rw-r-- 1 seed seed 223 Apr 19 18:48 myenv.c
-rw-rw-r-- 1 seed seed 417 Apr 19 16:06 myprintenv.c
-rw-rw-r-- 1 seed seed 3044 Apr 19 15:51 outputfile
-rw-rw-r-- 1 seed seed 3054 Apr 19 15:53 outputfileaftercommenting
-rw-rw-r-- 1 seed seed 89 Apr 19 19:16 task4.c
-rw-rw-r-- 1 root seed 152 Apr 19 19:44 task5.c
-rwsr-xr-x 1 root seed 16928 Apr 19 22:52 usingexecve
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve > doc_execveoutput
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve
Please type a file name.
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve doc_execveoutput
Please type a file name.
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve "doc_execveoutput;/bin/sh"
/bin/cat: 'doc_execveoutput;/bin/sh': No such file or directory
[04/19/24]seed@VM:~/.../Labsetup$ ./usingexecve doc_execveoutput;/bin/sh
Please type a file name.
$ rm doc_execvoutput
rm: cannot remove 'doc_execvoutput': No such file or directory
$



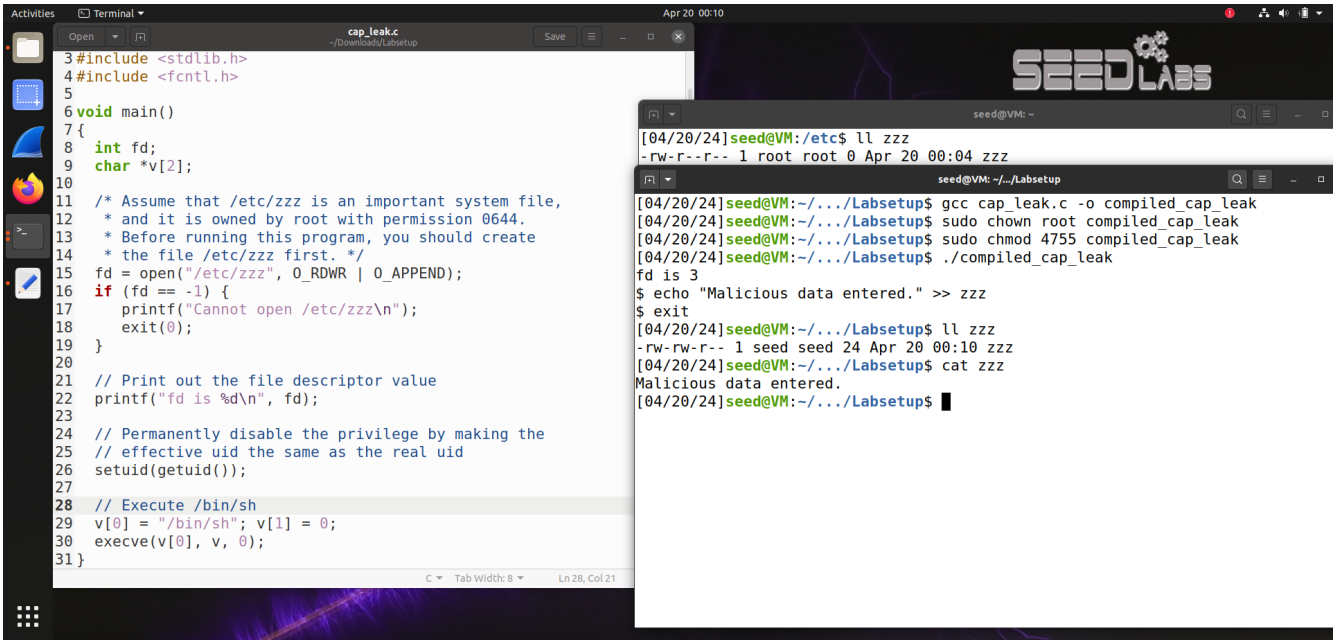
Task 9: Capability Leaking

First we create a file zzz in /etc . Then we compile and change permission to root for cap_leak.c

Now we run the program and see file descriptor value as 3. Also we are now able to exploit capability leak vulnerability as normal user seed because I can write any command here. As we can see I used echo “Malicious data entered.” >> zzz to write some malicious code to file zzz.

Now we check using cat command and data is successfully written.This happens because even though in the program, we dropped the privileges, we did not close the file at the right time and hence the file was still running with privileged permissions that allowed the data in the file to be modified, even without the right permissions.

Here, after calling fork, the control is passed to the child process and hence the malicious user is successful in modifying the content of a privileged file. Hence, this tells us that it’s important to close the file descriptor after dropping privileges, in order for it to have the appropriate permissions.


Activities
Terminal
Apr 20 00:10
Open
cap_leak.c
~/Downloads/Labsetup
Save
#include <stdlib.h>
#include <fcntl.h>
void main()
{
int fd;
char *v[2];
/* Assume that /etc/zzz is an important system file,
* and it is owned by root with permission 0644.
* Before running this program, you should create
* the file /etc/zzz first. */
fd = open("/etc/zzz", O_RDWR | O_APPEND);
if (fd == -1) {
printf("Cannot open /etc/zzz\n");
exit(0);
}
// Print out the file descriptor value
printf("fd is %d\n", fd);
// Permanently disable the privilege by making the
// effective uid the same as the real uid
setuid(getuid());
// Execute /bin/sh
v[0] = "/bin/sh"; v[1] = 0;
execve(v[0], v, 0);
}
C
Tab Width: 8
Ln 28, Col 21
SEED LABS
seed@VM: ~
[04/20/24]seed@VM:/etc$ ll zzz
-rw-r--r-- 1 root root 0 Apr 20 00:04 zzz
seed@VM: ~/.../Labsetup
[04/20/24]seed@VM:~/.../Labsetup$ gcc cap_leak.c -o compiled_cap_leak
[04/20/24]seed@VM:~/.../Labsetup$ sudo chown root compiled_cap_leak
[04/20/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 compiled_cap_leak
[04/20/24]seed@VM:~/.../Labsetup$ ./compiled_cap_leak
fd is 3
$ echo "Malicious data entered." >> zzz
$ exit
[04/20/24]seed@VM:~/.../Labsetup$ ll zzz
-rw-rw-r-- 1 seed seed 24 Apr 20 00:10 zzz
[04/20/24]seed@VM:~/.../Labsetup$ cat zzz
Malicious data entered.
[04/20/24]seed@VM:~/.../Labsetup$