Laboratory Report

Environment Variables and Set-UID

The laboratory prompt for this was provided by SEED Security Labs. SEED Security Labs is a project focused on enhancing cybersecurity education through hands-on laboratory exercises. Visit them at https://seedsecuritylabs.org/.

Ramnick Francis P. Ramos
+63 960 277 1720
ramnickfrancisramos@gmail.com

Cybersecurity Portfolio
September 09, 2025

Environment Variables and Set-UID

Ramnick Francis P. Ramos
ramnickfrancisramos@gmail.com

Table of Contents

Introduction

Environment variables are a "set of values with dynamic names that change how running processes in a system behave (Du, 2016). This laboratory report aims to show the understanding of how environment variables work and how they affect specific programs and the behavior of an operating system. It will also examine how Set-UID programs, which are privileged programs, are impacted by environment variables.

As mentioned in the Laboratory Exercise from SEED Labs, this report will cover the following concepts:

- Environment variables;
- Set-UIDprograms;
- Securely invoke external programs;
- Capability leaking; and
- Dynamic loader/linker.

Environment Setup

This lab was tested on the SEED Ubuntu 20.04 VM using Oracle VirtualBox. The prebuilt image for the virtual machine was obtained from CMSC 191: Cybersecurity's Google Classroom, but it can also be downloaded directly from the SEED website. The virtual machine ran locally, and no cloud server was used for this lab exercise.

Laboratory Tasks and Execution

Manipulating Environment Variables

```
[09/08/25]seed@VM:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1962,unix/VM:/tmp/.ICE-unix/1962
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=1925
GTK_MODULES=gail:atk-bridge
PWD=/home/seed

…
[09/08/25] seed@VM:~$ printenv PWD
/home/seed
[09/08/25] seed@VM:~$ env | grep PWD
PWD=/home/seed
OLDPWD=/home/seed/Documents
[09/08/25] seed@VM:~$
```

Figure 1: Output of the printenv command in the SEEDVM terminal.

Figure 1 shows how printenv is used to display the value of the PWD environment variable by appending it with PWD.PWD is the current **w**orking **d**irectory. grep, on the other hand, was used to show only those that start with PWD.

```
[09/08/25]seed@VM:~$ export temp="testing"
[09/08/25]seed@VM:~$ export
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"
declare -x DESKTOP_SESSION="ubuntu"
declare -x DISPLAY=":0"
declare -x GDM_SESSION="ubuntu"
…
declare -x temp="testing"
declare -x pwd
[09/08/25]seed@VM:~$
[09/08/25]seed@VM:~$ unset temp
[09/08/25]seed@VM:~$ export
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"
...
declare -x XDG_SESSION_DESKTOP="ubuntu"
declare -x XDG_SESSION_TYPE="x11"
declare -x XMODIFIERS="@im=ibus"
declare -x pwd
```

Figure 2: Output for using export and set.

Figure 2 shows how an environment variable named “testing” was exported. This variable can then be viewed as part of the environment variables. export was used to print all the environment variables. The unset command removed it.

These tasks show how to add environment variables using the export command and remove them with unset. They illustrate how these variables can be changed directly through shell commands.

Passing Environment Variables from Parent Process to Child Process

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc myprintenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1962,unix/VM:/tmp/.ICE-unix/1962
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
...
```

Figure 3: Examining the contents of Labsetup and myprintenv.c

Figure 3 shows the compilation of the myprintenv.c program and its subsequent execution, which prints the shell's environment variables.

```
[09/08/25] seed@VM:~/.../Labsetup$ gcc myprintenv.c
[09/08/25] seed@VM:~/.../Labsetup$ ./a.out > orig.txt
[09/08/25] seed@VM:~/.../Labsetup$ gcc myprintenv.c
[09/08/25] seed@VM:~/.../Labsetup$ ./a.out > modified.txt
[09/08/25] seed@VM:~/.../Labsetup$ diff orig.txt modified.txt
...
```

Figure 4: Comparing the output of the parents environment variables and child process

Furthermore, Figure 4 shows the code compilation and running where the printenv() command was uncommented, with the output from each run saved to a different file. The diff command is then used to compare the two output files. The absence of output from diff confirms that the files are the same. This implies that the parent environment variables are inherited by the child process.

Environment Variables and execve()

```
[09/08/25] seed@VM:~/.../Labsetup$ gcc myenv.c
[09/08/25] seed@VM:~/.../Labsetup$ a.out
[09/08/25] seed@VM:~/.../Labsetup$
```

Figure 5. Null Parameter for myenv.c

As shown in Figure 5, when NULL is passed as the third argument to execve(), the new process does not inherit or print any environment variables.

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc myenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out
[09/08/25]seed@VM:~/.../Labsetu[09/08/25]seed@VM:~/.[09/08/25]seed@VM:~/.../Labsetup
$ gcc myenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1962,unix/VM:/tmp/.ICE-unix/1962
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
```

Figure 6. Output after changing the parameters of execve() to environ.

After changing the third argument of execve() to environ, the parent process's environment variables are passed to the new process. The command line's output shows this inheritance. The new program receives its environment variables because the environ variable is declared globally, and this is then inherited by all processes.

Environment Variables and system()

```
[09/08/25]seed@VM:~/.../Labsetup$ touch task4.c
[09/08/25]seed@VM:~/.../Labsetup$ cat task4.c
#include <stdio.h>
#include <stdlib.h>
int main()
{system("/usr/bin/env");return 0 ;}
[09/08/25]seed@VM:~/.../Labsetup$ gcc task4.c
[09/08/25]seed@VM:~/.../Labsetup$ a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1925
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed/Documents
…
```

Figure 7. Output after changing the parameters of execve() to environ.

This block demonstrates how to compile and run task4.c (the contents is as shown in the cat command). This program uses the system() function to execute the /usr/bin/env command. The output shows a list of all current environment variables. This shows that system() has the power to show contents of the environment.

Environment Variable and Set-UID Programs

```
[09/08/25]seed@VM:~/.../Labsetup$ cat foo.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++;}}
[09/08/25]seed@VM:~/.../Labsetup$ gcc foo.c -o foo
```

```
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root foo
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 foo

…
[09/08/25]seed@VM:~/Desktop$ sudo useradd -m -s /bin/bash ramnick_new_user
[09/08/25]seed@VM:~/Desktop$ sudo passwd ramnick_new_user
passwd: password updated successfully
[09/08/25]seed@VM:~/Desktop$ su - ramnick_new_user
ramnick_new_user@VM:~$ whoami
ramnick_new_user
ramnick_new_user@VM:~$ export PATH=PATH:/usr/local/sbin
ramnick_new_user@VM:~$ export LD_LIBRARY_PATH=/home/seed/Desktop
ramnick_new_user@VM:~$ export ZZZ=/home/seed/Desktop
ramnick_new_user@VM:~$…

[09/08/25]seed@VM:~/.../Labsetup$ export
…
declare -x XDG_SESSION_DESKTOP="ubuntu"
declare -x XDG_SESSION_TYPE="x11"
declare -x XMODIFIERS="@im=ibus"
[09/08/25]seed@VM:~/.../Labsetup$

ramnick_new_user@VM:~$ export
declare -x XDG_DATA_DIRS="/usr/local/share:/usr/share:/var/lib/snapd/desktop"
declare -x ZZZ="/home/seed/Desktop"
ramnick_new_user@VM:~$
```

Figure 8. Running foo Set-UID Programs

This task shows how environment variables are separate for different users. The lab creates a new user, ramnick_new_user, and sets a few environment variables. When a user with higher privileges, the seed user, checks their environment, the changes made by the new user do not impact them (See last two blocks in Figure 8 where the ZZZ environment variable cannot be seen when the root user is turned back).

LD_PRELOAD and other changes was then unset after for the continuation of the usage of the operating system.

The PATH Environment Variable and Set-UID Programs

```
[09/08/25]seed@VM:~/.../Labsetup$ cat myprog.c
#include <stdlib.h>
#include <unistd.h>

int main()
{
system("ls");
return 0;
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[09/08/25]seed@VM:~/.../Labsetup$ nano ~/ls
[09/08/25]seed@VM:~/.../Labsetup$ chmod +x ~/ls
[09/08/25]seed@VM:~/.../Labsetup$ export PATH=/home/seed:$PATH
```

```
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
Ramnick was here! Running as: seed
```

Figure 9. Demonstrating malicious scripts

This task shows a weakness where a harmful script can run instead of a trusted program by altering the PATH environment variable. By changing the PATH to give priority to a user-defined directory with a harmful script named ls, the system() function in a Set-UID program will execute the harmful script instead of the real /bin/ls. The harmful code runs with root privileges because it is a Set-UID program owned by root.

The LD PRELOAD Environment Variable and Set-UID Programs

```
[09/08/25]seed@VM:~/.../Labsetup$ cat mylib.c
#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");
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc -fPIC -g -c mylib.c
[09/08/25]seed@VM:~/.../Labsetup$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/08/25]seed@VM:~/.../Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/08/25]seed@VM:~/.../Labsetup$ cat myprog.c
/*myprog.c*/
#include <unistd.h>
int main()
{
sleep(1);
return 0;
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc -o myprog myprog.c

[SCENARIO A]
[09/08/25]seed@VM:~/.../Labsetup$ unset LD_PRELOAD
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
[09/08/25]seed@VM:~/.../Labsetup$


[SCENARIO B]
[09/08/25]seed@VM:~/.../Labsetup$ export LD_PRELOAD=$PWD/libmylib.so.1.0.1
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
[09/08/25]seed@VM:~/.../Labsetup$

[09/08/25]seed@VM:~/.../Labsetup$ export LD_PRELOAD=$PWD/libmylib.so.1.0.1
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
[09/08/25]seed@VM:~/.../Labsetup$

[SCENARIO C]
root@VM:/home/seed/Documents/Labsetup# export LD_PRELOAD=$PWD/libmylib.so.1.0.1
root@VM:/home/seed/Documents/Labsetup# ./myprog
```

```
I am not sleeping!
root@VM:/home/seed/Documents/Labsetup#
```

Figure 10. LD_PRELOAD Environment Variable and Corresponding scenarios

This task explores how LD_PRELOAD can be used to replace a function (sleep()) with a custom function (mylib.c). For regular programs, the LD_PRELOAD variable works as expected where the sleep(1) halts the running for a second. However, for Set-UID root programs including ones that are ran as a root user (Scenario B and C), the dynamic linker ignores the LD_PRELOAD environment variable for security reasons. This stops a malicious user from replacing system functions in privileged programs.

Invoking External Programs Using system() versus execve()

TABLE 1. Comparing system() and execve()

Function	Behavior	Security Implications
system()	Runs through invoking a shell command	Since this utilizes shell commands, the PATH variable becomes vulnerable
execve()	This uses replacement of the current process with a new one	Since no shell is invoked, this function is deemed more secure

Table 1 compares the system() and execve(). On the question Can Bob compromise the integrity of the system? For instance, can someone remove a file that is not writable for them? The answer is yes. Since the script calls system(), we can include a harmful script by changing the PATH environmental variable, just as was done in Task 6. Therefore, even though files are not writable for Bob, he can bypass the system by manipulating its prioritization.

The following coding scenarios were executed to exhibit:

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc -o catall catall.c
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root catall
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[09/08/25]seed@VM:~/.../Labsetup$ ./catall /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
….
```

Figure 11. Using chmod with catcall c using system()

As seen on Figure 11, this block of code utilizes the given catall.c file that utilizes the system() command. This provides the content of the /etc/pwd.

For the next part, we attempt to attack this vulnerability using malicious programs. Below is how the cat was hijacked using a malicious script that overwrote the cat.

```
[09/08/25]seed@VM:~/Desktop$ mkdir liar
[09/08/25]seed@VM:~/Desktop$ echo -e "echo Ramnicked" > liar/cat
[09/08/25]seed@VM:~/Desktop$ chmod +x liar/cat
[09/08/25]seed@VM:~/Desktop$ export PATH=$PWD/liar:$PATH
[09/08/25]seed@VM:~/Desktop$ ./catall /etc/passwd
Ramnicked
```

Figure 12.2. Using chmod with catcall c using system().

In the case of execve(), the program is changed to comment out system() command and uncomment the execve() call. This adjustment makes it more secure by executing the program directly without a shell.

```
[09/08/25]seed@VM:~/Desktop$ mkdir liar
[09/08/25]seed@VM:~/Desktop$ echo -e "echo Ramnicked" > liar/cat
[09/08/25]seed@VM:~/Desktop$ chmod +x liar/cat
[09/08/25]seed@VM:~/Desktop$ export PATH=$PWD/liar:$PATH
[09/08/25]seed@VM:~/Desktop$ ./catall /etc/passwd
```

Figure 12.2. Using chmod with catcall c using execve().

After that, the terminal successfully prints the contents of /etc/passwd because the malicious script was not executed.

Capability Leaking

```
[09/08/25]seed@VM:~/Desktop$ gcc cap_leak.c -o cap_leak
[09/08/25]seed@VM:~/Desktop$ sudo touch /etc/zzz
[09/08/25]seed@VM:~/Desktop$ sudo chmod 644 /etc/zzz
[09/08/25]seed@VM:~/Desktop$ sudo chown root cap_leak
[09/08/25]seed@VM:~/Desktop$ sudo chmod 4755 cap_leak
[09/08/25]seed@VM:~/Desktop$ sudo -u user1 ./cap_leak
fd is 3
$ echo "Ramnick was here" >&3
$ cat /etc/zzz
Ramnick was here
```

Figure 13. Capability Leaking

This task shows how the privileged file descriptors can leak this privilege from a set-UID program to a regularly run user's shell. The cap_leak file opens a file (initially with root privileges) and, through the running of this in a regular shell, the privilege is leaked through the exposition of its file descriptor (3). Then, I was able to run the "Ramnick was here" using the cat command.

Challenges and Troubleshooting

The main challenge in this lab exercise was the limited skill with Linux shell commands. To tackle this, I often referred to an online cheat sheet, which turned out to be a helpful resource. Additionally, as this laboratory exercise incorporates changing the environment variables of the system, the challenge of how the altered behavior of the OS impedes the proper functioning of shell commands was faced. To troubleshoot this, the unset keyword was used after all changes are done and finished (if they are not to be used anymore.

Discussion

This laboratory exercise showed how environmental variables can be changed and how this change gives flexibility for attackers to exploit the vulnerabilities of the system. Attack surfaces on environment variables can be leveraged to promote privileges and possibly alter the system without proper authentication – this poses a risk to security. As a solution, capitalizing on using commands that do not require invoking any shell commands is imperative.

References

Du, W. (2016). SEED Labs: Environment Variable and Set-UID Program Lab. SEED Project.

Appendix: Screenshots

```
[09/08/25]seed@VM:~/.../Labsetup$ [09/08/25]seed@VM:~$ printenv
bash: [09/08/25]seed@VM:~$: No such file or directory
[09/08/25]seed@VM:~/.../Labsetup$ SHELL=/bin/bash
[09/08/25]seed@VM:~/.../Labsetup$ SESSION_MANAGER=local/VM:@/tm
p/.ICE-unix/1962,unix/VM:/tmp/.ICE-unix/1962
[09/08/25]seed@VM:~/.../Labsetup$ QT_ACCESSIBILITY=1
[09/08/25]seed@VM:~/.../Labsetup$ COLORTERM=truecolor
[09/08/25]seed@VM:~/.../Labsetup$ XDG_CONFIG_DIRS=/etc/xdg/xdg-
ubuntu:/etc/xdg
[09/08/25]seed@VM:~/.../Labsetup$ XDG_MENU_PREFIX=gnome-
```

```
[09/08/25]seed@VM:~/.../Labsetup$ PWD=/home/seed
[09/08/25]seed@VM:~$ printenv PWD
/home/seed
[09/08/25]seed@VM:~$ env | grep PWD PWD=/home/seed
grep: PWD=/home/seed: No such file or directory
[09/08/25]seed@VM:~$ env | grep PWD
PWD=/home/seed
OLDPWD=/home/seed/Documents
[09/08/25]seed@VM:~$
```

```
[09/08/25]seed@VM:~/.../Labsetup$ export temp=
"testing"
[09/08/25]seed@VM:~/.../Labsetup$ export
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path
=/run/user/1000/bus"
declare -x DESKTOP_SESSION="ubuntu"
```

```
[09/08/25]seed@VM:~/.../Labsetup$ export temp=
"testing"
[09/08/25]seed@VM:~/.../Labsetup$ export
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path
=/run/user/1000/bus"
declare -x DESKTOP_SESSION="ubuntu"
```

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc myprintenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out >orig.txt
[09/08/25]seed@VM:~/.../Labsetup$ gcc myprintenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out > modified
.txt
[09/08/25]seed@VM:~/.../Labsetup$ diff orig.txt  mod
ified.txt
[09/08/25]seed@VM:~/.../Labsetup$
```

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc myenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out
[09/08/25]seed@VM:~/.../Labsetu[09/08/25]seed@VM:~/.
[09/08/25]seed@VM:~/.../Labsetup$ gcc myenv.c
[09/08/25]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1962,unix/V
M:/tmp/.ICE-unix/1962
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
[09/08/25]seed@VM:~/.../Labsetup$ gcc task4.c
[09/08/25]seed@VM:~/.../Labsetup$ a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1925
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed/Documents
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1722
```

```
[09/08/25]seed@VM:~/.../Labsetup$ cat foo.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++;}}
[09/08/25]seed@VM:~/.../Labsetup$ gcc foo.c -o foo
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root foo
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 foo
[09/08/25]seed@VM:~/.../Labsetup$ sudo useradd -m -s /bin/bash ramnick_new
_user
[09/08/25]seed@VM:~/.../Labsetup$ sudo passwd ramnick_new_user
New password:
Retype new password:
passwd: password updated successfully
[09/08/25]seed@VM:~/.../Labsetup$ su - ramnick_new_user
Password:
ramnick_new_user@VM:~$ whoami
ramnick_new_user
ramnick_new_user@VM:~$ export PATH=PATH:/usr/local/sbin
ramnick_new_user@VM:~$ export LD_LIBRARY_PATH=/home/seed/Desktop
ramnick_new_user@VM:~$ export ZZZ=/home/seed/Desktop
declare -x XDG_DATA_DIRS="/usr/local/share:/usr/share:/va
r/lib/snapd/desktop"
declare -x ZZZ="/home/seed/Desktop"
ramnick_new_user@VM:~$
declare -x XDG_SESSION_TYPE="x11"
declare -x XMODIFIERS="@im=ibus"
[09/08/25]seed@VM:~/.../Labsetup$
```

```
[09/08/25]seed@VM:~/.../Labsetup$ cat myprog.c
#include <stdlib.h>
#include <unistd.h>

int main()
{
system("ls");
return 0;
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[09/08/25]seed@VM:~/.../Labsetup$ nano ~/ls
[09/08/25]seed@VM:~/.../Labsetup$ chmod +x ~/ls
[09/08/25]seed@VM:~/.../Labsetup$ export PATH=/home/seed:$P
ATH
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
Ramnick was here! Running as: seed
[09/08/25]seed@VM:~/.../Labsetup$
```

```
[09/08/25]seed@VM:~/.../Labsetup$ cat mylib.c
#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");
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc -fPIC -g -c mylib.c
[09/08/25]seed@VM:~/.../Labsetup$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/08/25]seed@VM:~/.../Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/08/25]seed@VM:~/.../Labsetup$ cat myprog.c
/*myprog.c*/
#include <unistd.h>
int main()
{
sleep(1);
return 0;
}
[09/08/25]seed@VM:~/.../Labsetup$ gcc -o myprog myprog.c
[09/08/25]seed@VM:~/.../Labsetup$
[09/08/25]seed@VM:~/.../Labsetup$ gcc -o myprog myprog.c
[09/08/25]seed@VM:~/.../Labsetup$
[09/08/25]seed@VM:~/.../Labsetup$ unset LD_PRELOAD
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
```

```
[09/08/25]seed@VM:~/.../Labsetup$ export LD_PRELOAD=$PWD/libmylib.so.1.0.1
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[09/08/25]seed@VM:~/.../Labsetup$ ./myprog
[09/08/25]seed@VM:~/.../Labsetup$ sudo -i
root@VM:/home/seed/Documents/Labsetup# export LD_PRELOAD=$PWD/libmylib.so.1.0.1
root@VM:/home/seed/Documents/Labsetup# ./myprog
I am not sleeping!
root@VM:/home/seed/Documents/Labsetup#
```

```
[09/08/25]seed@VM:~/.../Labsetup$ gcc cap_leak.c -o cap_leak
[09/08/25]seed@VM:~/.../Labsetup$ sudo touch /etc/zzz
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 644 /etc/zzz
[09/08/25]seed@VM:~/.../Labsetup$ sudo chown root cap_leak
[09/08/25]seed@VM:~/.../Labsetup$ sudo chmod 4755 cap_leak
[09/08/25]seed@VM:~/.../Labsetup$ ./cap_leak
fd is 3
$ echo "Ramnick was here" >&3
$ cat /etc/zzz
Ramnick was here
$
```