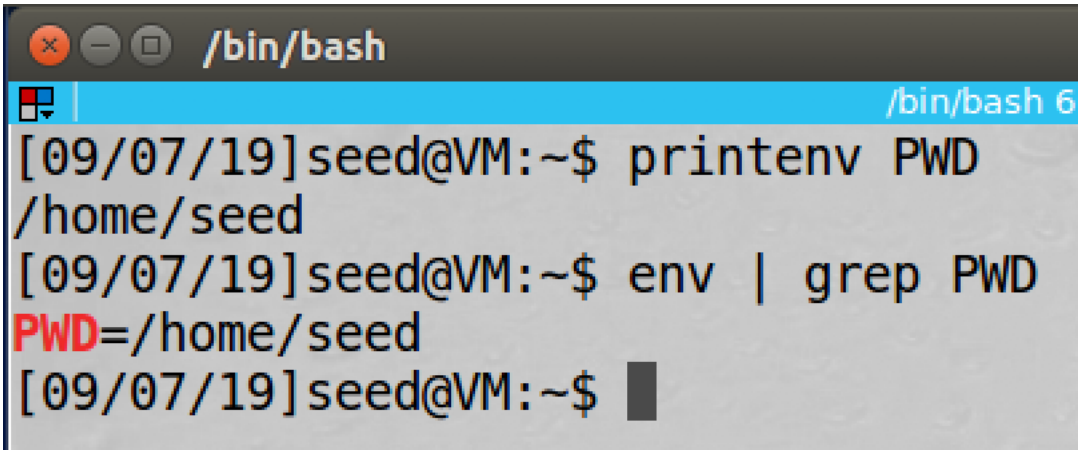
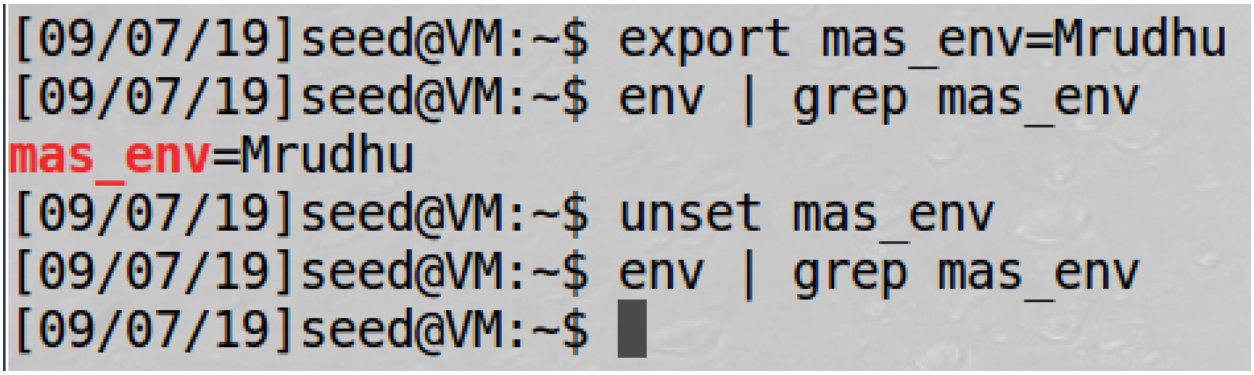
LAB 1: Environment Variable and Set-UID Program Lab

Task 1: Manipulating environment variables

Observation:

When we give 'env' command, the environment variables are printed





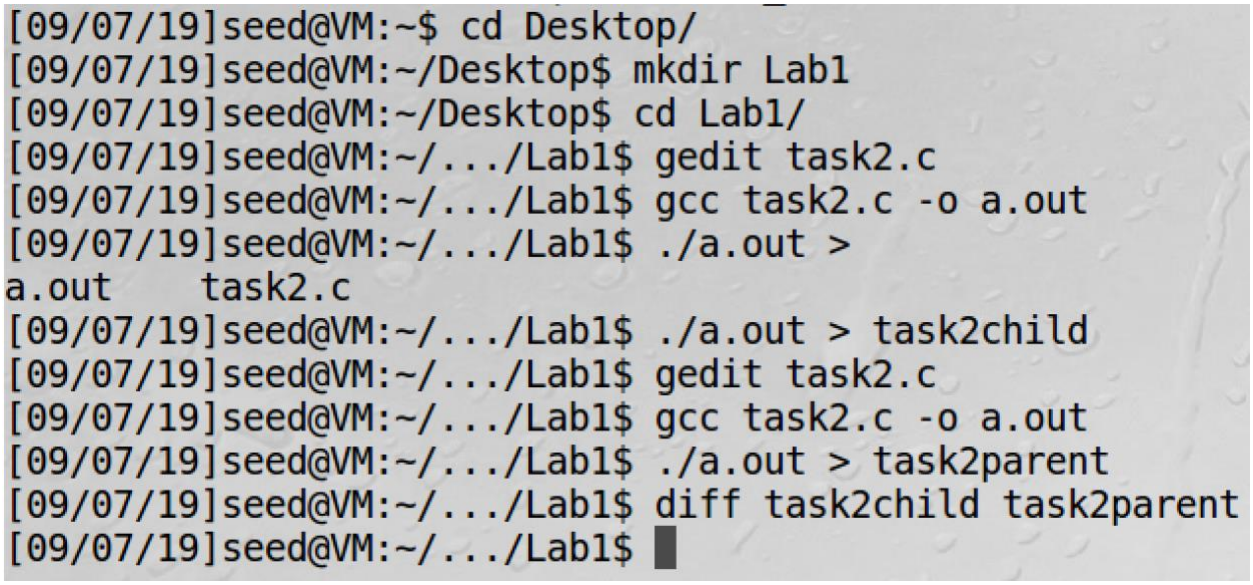
In the above, we are trying to create a new environment variable called mas_env and we are trying to show that it exists in the list of the environment variables. Then, we are deleting mas_env from the set of environment variables. Then we run the grep command after deleting mas_env, the bash shell is returned, indicating that no such variable exists.

Explanation:

'Export' command is used to set environment variables and 'unset' command is used to unset/delete environment variables.

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

Observation:



In this task, we are creating a program task2.c. The output of this program is stored in task2child. Then the program is modified and the output is stored in task2parent. Then we compare the two files using 'diff' command and the shell is returned indicating that there are no differences between the files.

Explanation:

A copy of parent's environment is inherited by the child processes. Hence, there is no difference when we run the 'diff' command on parent's environment variables and child's environment variables.

Task 3: Environment variables and execve()

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task3.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task3.c -o a.out
task3.c: In function ‘main’:
task3.c:10:2: warning: implicit declaration of function ‘execve’ [
-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, NULL);
  ^
[09/07/19]seed@VM:~/.../Lab1$ ./a.out
[09/07/19]seed@VM:~/.../Lab1$
```

```
task3.c:10:2: warning: implicit declaration of function ‘execve’ [
-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, environ);
  ^
[09/07/19]seed@VM:~/.../Lab1$ ./a.out
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:39e96c09-8cb5-4b14-8a20-54368647afe8
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=2080
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0
:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib
/boost/libboost_system.so.1.64.0
WINDOWID=35651588
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/115
0
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;
41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.a
```

In this task, we are creating a program task3.c, in which, the environment variables are set to 'NULL' in the 3rd argument of the 'execve' command. Since only the shell is returned, we do not obtain an output. We then modify the 3rd argument of the 'execve' command and set the environment variables. In output of this program we obtain the environment variables.

Explanation:

When the environment variables argument of the 'execve' command is set to 'NULL', it is not stored in the environment and argument memory and so the new program does not inherit the environment variables of the calling process. But the argument takes the environment variables, they are stored in memory, then the new program inherits the environment variables of the calling process.

Task 4: Environment variables and system()

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task4.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task4.c -o task4.out
[09/07/19]seed@VM:~/.../Lab1$ cat task4.c
#include <stdio.h>
#include <stdlib.h>

int main()
{
        system("/usr/bin/env");
        system("sleep 1000");
        return 0 ;
}
[09/07/19]seed@VM:~/.../Lab1$ ./task4.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
ORBIT_SOCKETDIR=/tmp/orbit-seed
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/see
d/source/boost_1_64_0/stage/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
OLDPWD=/home/seed/Desktop
```

```
rmvb=01;35:*.flc=01;35:*.avi=01;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:*.e
mf=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:*.opus=00
;36:*.spx=00;36:*.xspf=00;36:
XMODIFIERS=@im=ibus
XDG_SESSION_DESKTOP=ubuntu
XAUTHORITY=/home/seed/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
TERMINATOR_UUID=urn:uuid:39e96c09-8cb5-4b14-8a20-54368647afe8
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/115
0
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Desktop/Lab1
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xd
g
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
```

In this task, we are creating a program task4.c with a sleep function within a system call. This is used to show, a chain of processes called by the system call. When we execute this program, the program sleeps for 1000s.

New twerminal to see process active:

```
[09/07/19]seed@VM:~$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 18:13 ?        00:00:01 /sbin/init splash
root         2     0  0 18:13 ?        00:00:00 [kthreadd]
root         3     2  0 18:13 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 18:13 ?        00:00:00 [kworker/0:0H]
root         7     2  0 18:13 ?        00:00:00 [rcu_sched]
root         8     2  0 18:13 ?        00:00:00 [rcu_bh]
root         9     2  0 18:13 ?        00:00:00 [migration/0]
root        10     2  0 18:13 ?        00:00:00 [lru-add-drain]
root        11     2  0 18:13 ?        00:00:00 [watchdog/0]
root        12     2  0 18:13 ?        00:00:00 [cpuhp/0]
root        13     2  0 18:13 ?        00:00:00 [kdevtmpfs]
root        14     2  0 18:13 ?        00:00:00 [netns]
root        15     2  0 18:13 ?        00:00:00 [khungtaskd]
root        16     2  0 18:13 ?        00:00:00 [oom_reaper]
root        17     2  0 18:13 ?        00:00:00 [writeback]
root        18     2  0 18:13 ?        00:00:00 [kcompactd0]
root        19     2  0 18:13 ?        00:00:00 [ksmd]
root        20     2  0 18:13 ?        00:00:00 [khugepaged]
root        21     2  0 18:13 ?        00:00:00 [crypto]
root        22     2  0 18:13 ?        00:00:00 [kintegrityd]
root        23     2  0 18:13 ?        00:00:00 [bioset]
root        24     2  0 18:13 ?        00:00:00 [kblockd]
root        25     2  0 18:13 ?        00:00:00 [ata_sff]
root        26     2  0 18:13 ?        00:00:00 [md]
root        27     2  0 18:13 ?        00:00:00 [devfreq_wq]
root        28     2  0 18:13 ?        00:00:00 [watchdogd]
root        29     2  0 18:13 ?        00:00:00 [kworker/u2:1]
root        32     2  0 18:13 ?        00:00:00 [kswapd0]
root        33     2  0 18:13 ?        00:00:00 [vmstat]
root        34     2  0 18:13 ?        00:00:00 [ecryptfs-kthrea]
root        73     2  0 18:13 ?        00:00:00 [kthrotld]
```

In this new terminal, we execute the ‘ps -ef’ command. This command shows all the active process.

```
seed      1952  1150  0 18:13 ?        00:00:00 /usr/lib/evolution
seed      1963  1150  0 18:13 ?        00:00:00 /usr/lib/gvfs/gvfs
seed      1971  1150  0 18:13 ?        00:00:00 /usr/lib/gvfs/gvfs
seed      1983  1150  0 18:13 ?        00:00:00 /usr/lib/gvfs/gvfs
seed      1987  1952  0 18:13 ?        00:00:00 /usr/lib/evolution
root      1992     1  0 18:13 ?        00:00:00 /usr/lib/i386-linu
seed      2032  1150  0 18:13 ?        00:00:00 /usr/lib/gvfs/gvfs
seed      2080  1150  0 18:13 ?        00:00:01 /usr/bin/python /u
seed      2090  1150  0 18:13 ?        00:00:00 /usr/lib/i386-linu
seed      2094  2080  0 18:13 ?        00:00:00 gnome-pty-helper
seed      2095  2080  0 18:13 pts/17   00:00:00 /bin/bash
seed      2108  1646  0 18:13 ?        00:00:00 zeitgeist-datahub
seed      2115  1150  0 18:13 ?        00:00:00 /bin/sh -c /usr/li
seed      2119  2115  0 18:13 ?        00:00:00 /usr/bin/zeitgeist
seed      2127  1150  0 18:13 ?        00:00:00 /usr/lib/i386-linu
seed      2154  1646  0 18:14 ?        00:00:00 update-notifier
seed      2175  1150  0 18:14 ?        00:00:02 /usr/bin/python3 /
seed      2199  1646  0 18:15 ?        00:00:00 /usr/lib/i386-linu
seed      2239  1150  0 18:18 ?        00:00:00 /usr/lib/gvfs/gvfs
root      2256     2  0 18:19 ?        00:00:00 [kworker/0:0]
seed      2268  1150  0 18:19 ?        00:00:00 /usr/lib/gvfs/gvfs
seed      2286  1150  0 18:19 ?        00:00:00 /usr/lib/gvfs/gvfs
root      2397     2  0 18:27 ?        00:00:00 [kworker/u2:0]
root      2402     2  0 18:28 ?        00:00:00 [kworker/0:1]
root      2480     2  0 18:33 ?        00:00:00 [kworker/u2:2]
seed      2511  2095  0 18:37 pts/17   00:00:00 ./task4.out
seed      2514  2511  0 18:37 pts/17   00:00:00 sh -c sleep 1000
seed      2515  2514  0 18:37 pts/17   00:00:00 sleep 1000
seed      2517  1150  3 18:37 ?        00:00:00 /usr/bin/python /u
seed      2531  2517  0 18:37 ?        00:00:00 gnome-pty-helper
seed      2532  2517  0 18:37 pts/0    00:00:00 /bin/bash
seed      2545  2532  0 18:37 pts/0    00:00:00 ps -ef
[09/07/19]seed@VM:~$
```

We observe that the shell calls the executable program, which then calls the sh -c command , which then internally calls the ‘execve’ command.

Explanation:

When the system function executes, it does not execute the command directly. It calls the shell instead and the shell executes the command. The shell internally calls the ‘execve’ command, and the environment variables of the calling process are passed to the shell, which in turn passes to ‘execve’ command.

Task 5: Environment Variable and Set-UID Programs

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task5.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task5.c -o task5
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root task5
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 task5
[09/07/19]seed@VM:~/.../Lab1$ ls -l task5
-rwsr-xr-x 1 root seed 7396 Sep  7 18:42 task5
[09/07/19]seed@VM:~/.../Lab1$ export PATH=/home/seed/:$PATH
[09/07/19]seed@VM:~/.../Lab1$ export LD_LIBRARY_PATH=mrudhu
[09/07/19]seed@VM:~/.../Lab1$ export mas_env=Mrudhu
[09/07/19]seed@VM:~/.../Lab1$ ./task5 > task5out.txt
[09/07/19]seed@VM:~/.../Lab1$ grep PATH task5out.txt
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/us
r/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:.:/snap/bin
:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:
/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-
linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/h
ome/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[09/07/19]seed@VM:~/.../Lab1$ grep LA_LIBRARY_PATH task5out.txt
[09/07/19]seed@VM:~/.../Lab1$ grep mas_env task5out.txt
mas_env=Mrudhu
[09/07/19]seed@VM:~/.../Lab1$
```

In this task, we are creating a program and compiling it. Then we change the ownership of the program using the 'chown' command. Then we make this program into a Set-UID program using the 'chmod' command. To know if the program is a Set-UID program it is denoted by 's' when we execute the 'ls -l' command. Then, we set 3 environment variables using the 'export' command and then run the program. The environment variables 'PATH' and 'mas_env' are inherited into the Set-UID program. But the environment variable 'LD_LIBRARY_PATH' is not inherited.

Explanation:

'LD_LIBRARY_PATH' is a path from which shared libraries are accessed. Is a privileged path and is automatically ignored if a Set-UID program accesses it. It is a mechanism to protect against malicious files being placed into shared libraries. There will be a predefined path from which the program accesses shared libraries which cannot be altered for the Set-UID programs.

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

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task6.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task6.c -o task6
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root task6
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 task6
[09/07/19]seed@VM:~/.../Lab1$ ls -l task6
-rwsr-xr-x 1 root seed 7348 Sep  7 18:57 task6
[09/07/19]seed@VM:~/.../Lab1$ gedit ls.c
[09/07/19]seed@VM:~/.../Lab1$ gcc ls.c -o ls
[09/07/19]seed@VM:~/.../Lab1$ export PATH=/home/seed:$PATH
[09/07/19]seed@VM:~/.../Lab1$ printenv PATH
/home/seed:/home/seed/:/home/seed/bin:/usr/local/sbin:/usr/local/b
in:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:.:/sn
ap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/d
b/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/androi
d-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-to
ols:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.loc
al/bin
[09/07/19]seed@VM:~/.../Lab1$ ./task6
a.out  task2.c      task3.c    task5         task6
ls     task2child   task4.c    task5.c       task6.c
ls.c   task2parent  task4.out  task5out.txt
[09/07/19]seed@VM:~/.../Lab1$
```

Code inside the ‘ls’ program.

```
[09/07/19]seed@VM:~/.../Lab1$ cat ls.c
#include<stdio.h>
#include<stdlib.h>

int main()
{
        system("pwd");
        return 0;
}
[09/07/19]seed@VM:~/.../Lab1$
```

In this task, we are creating a program task6.c with the system function containing 'ls' and 'compiled'. Then we change the ownership of the file using 'chown' command, then make it a Set-UID program using the 'chmod' command. When we run the 'ls -l' command, it shows that the given file is a Set-UID program. We then change the value of the 'PATH' environment variable. We then create a new program 'ls' and compile it. This is the malicious file that we are placing in the path. This file is going to replace the functionality of the 'ls' command. Hence, the current working directory is printed as specified by the modified 'ls' program.

Explanation:

The 'PATH' environment variable looks for the command 'ls' in the current directory first since it is specified. Since it finds, that ls exists, it runs that program instead of the 'shell ls' command. Hence, we know that Set-UID programs may run malicious files with root privileges if the PATH variable is altered. To avoid attacks like this, we can always run commands using the absolute path as specified in the screenshot above.

Task 7: The LD PRELOAD environment variable and Set-UID Programs

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit mylib.c
[09/07/19]seed@VM:~/.../Lab1$ gcc -fPIC -g -c mylib.c
[09/07/19]seed@VM:~/.../Lab1$ gcc -shared -o libmylib.so.1.0.1 myl
ib.o -lc
[09/07/19]seed@VM:~/.../Lab1$ export LD_PRELOAD=./libmylib.so.1.0.
1
[09/07/19]seed@VM:~/.../Lab1$ gedit myprog.c
[09/07/19]seed@VM:~/.../Lab1$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:5:2: warning: implicit declaration of function 'sleep' [-
Wimplicit-function-declaration]
  sleep(1);
  ^
[09/07/19]seed@VM:~/.../Lab1$ ./myprog
I am not sleeping!
[09/07/19]seed@VM:~/.../Lab1$
```

In this task, we are creating a program mylib.c, we compile and make it a dynamic link library. We set the 'LD_PRELOAD' environment variable using the 'export' command to point to the DLL we just created. We are then creating a program myprog.c and compile it. When we run the 'myprog' program, the output is shown as above. It means that this program calls the 'mylib' DLL that we just created instead of the lib.c DLL.

```
[09/11/18]seed@VM:~/.../Lab1$ sudo chown root myprog
[sudo] password for seed:
[09/11/18]seed@VM:~/.../Lab1$ sudo chmod 4755 myprog
[09/11/18]seed@VM:~/.../Lab1$ /b
bin/  boot/
[09/11/18]seed@VM:~/.../Lab1$ /bin/ls -l myprog
-rwsr-xr-x 1 root seed 7348 Sep 11 19:45 myprog
[09/11/18]seed@VM:~/.../Lab1$ ./myprog
[09/11/18]seed@VM:~/.../Lab1$ sudo su root
root@VM:/home/seed/Desktop/compsec/Lab1#
```

In the above screenshot, we are making the 'myprog' program as a Set-UID program owned by root. When we run the program, the program sleeps for some time and the shell prompt is returned indicating that the program doesn't invoke the 'mylib' DLL that we created.

```
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root myprog
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 myprog
[09/07/19]seed@VM:~/.../Lab1$ /bin/ls -l myprog
-rwsr-xr-x 1 root seed 7348 Sep  7 19:06 myprog
[09/07/19]seed@VM:~/.../Lab1$ ./myprog
[09/07/19]seed@VM:~/.../Lab1$ sudo su root
root@VM:/home/seed/Desktop/Lab1#
```

```
root@VM:/home/seed/Desktop/Lab1# gcc -shared -o libmylib.so.1.0
mylib.o -lc
root@VM:/home/seed/Desktop/Lab1# export LD_PRELOAD=./libmylib.s
.0.1
root@VM:/home/seed/Desktop/Lab1# gcc myprog.c -o myprog
myprog.c: In function ‘main’:
myprog.c:5:2: warning: implicit declaration of function ‘sleep’
Wimplicit-function-declaration]
  sleep(1);
  ^
root@VM:/home/seed/Desktop/Lab1# ./myprog
I am not sleeping!
root@VM:/home/seed/Desktop/Lab1# sudo chown root myprog
root@VM:/home/seed/Desktop/Lab1# sudo chmod 4755 mypr
chmod: cannot access 'mypr': No such file or directory
root@VM:/home/seed/Desktop/Lab1# sudo chmod 4755 myprog
root@VM:/home/seed/Desktop/Lab1# /bin/ls -l myprog
-rwsr-xr-x 1 root root 7348 Sep  7 19:10 myprog
root@VM:/home/seed/Desktop/Lab1# ./myprog
I am not sleeping!
```

```
root@VM:/home/seed/Desktop/Lab1# export LD_PRELOAD=./libmylib.so.1
.0.1
root@VM:/home/seed/Desktop/Lab1# printenv LD_PRELOAD ./libmylib.so
.1.0.1
./libmylib.so.1.0.1
root@VM:/home/seed/Desktop/Lab1# ./myprog
I am not sleeping!
root@VM:/home/seed/Desktop/Lab1# sudo adduser seed1
Adding user `seed1' ...
Adding new group `seed1' (1001) ...
Adding new user `seed1' (1001) with group `seed1' ...
Creating home directory `/home/seed1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for seed1
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

```
root@VM:/home/seed/Desktop/Lab1# sudo chown seed1 myprog
root@VM:/home/seed/Desktop/Lab1# sudo chmod 4755 myprog
root@VM:/home/seed/Desktop/Lab1# /bin/ls -l
total 112
-rwxrwxr-x 1 seed  seed 7448 Sep  7 18:28 a.out
-rwxr-xr-x 1 root  root 7928 Sep  7 19:09 libmylib.so.1.0.1
-rwxrwxr-x 1 seed  seed 7344 Sep  7 18:59 ls
-rw-rw-r-- 1 seed  seed   86 Sep  7 18:59 ls.c
-rw-rw-r-- 1 seed  seed  162 Sep  7 19:03 mylib.c
-rw-rw-r-- 1 seed  seed 2588 Sep  7 19:03 mylib.o
-rwsr-xr-x 1 seed1 root 7348 Sep  7 19:10 myprog
-rw-rw-r-- 1 seed  seed   71 Sep  7 19:05 myprog.c
-rw-rw-r-- 1 seed  seed  391 Sep  7 18:23 task2.c
-rw-rw-r-- 1 seed  seed 4265 Sep  7 18:22 task2child
-rw-rw-r-- 1 seed  seed 4265 Sep  7 18:23 task2parent
-rw-rw-r-- 1 seed  seed  194 Sep  7 18:28 task3.c
-rw-rw-r-- 1 seed  seed  119 Sep  7 18:37 task4.c
-rwxrwxr-x 1 seed  seed 7348 Sep  7 18:37 task4.out
-rwsr-xr-x 1 root  seed 7396 Sep  7 18:42 task5
-rw-rw-r-- 1 seed  seed  176 Sep  7 18:42 task5.c
-rw-rw-r-- 1 seed  seed 4028 Sep  7 18:44 task5out.txt
-rwsr-xr-x 1 root  seed 7348 Sep  7 18:57 task6
-rw-rw-r-- 1 seed  seed   81 Sep  7 18:56 task6.c
root@VM:/home/seed/Desktop/Lab1# export LD_PRELOAD=./libmylib.so.1
.0.1
root@VM:/home/seed/Desktop/Lab1# ./myprog
root@VM:/home/seed/Desktop/Lab1#
```

In the above screenshot, we are executing the ‘myprog’ program from root account. It is a set-UID program owned by root. We set the ‘LD_PRELOAD’ environment variable pointing to the DLL we created. When we run the program, the program calls the ‘mylib’ DLL we created. We are creating a new user ‘seed1’. We make the ‘myprog’ program a set-UID program owned by seed1 user. We then set the LD_PRELOAD environment variable pointing to the DLL we created. When we execute the program from another user account like seed, the program sleeps for some time. This means that the program doesn’t invoke the DLL we created.

Explanation:

The LD_PRELOAD environment variable is always ignored if a Set-UID program accesses it. It is a protection mechanism in UNIX.
In the first case: ‘myprog’ is a regular program and run by a normal user. Hence, LD_PRELOAD is not ignored and the new malicious DLL file created by us is accessed.
In the second case: ‘myprog’ is a Set-UID root program and run by a normal user. Since it is a Set-UID program, LD_PRELOAD is ignored and the DLL file created by us isn’t accessed, instead the default library file is accessed.

In the third case: 'myprog' is a Set-UID program and run by root. Here it checks for effective UID and real UID and since both are related to root, it trusts the DLL file and runs the DLL we created.
In the last case: 'myprog' is a Set-UID program owned by a user and run by another user. Hence, LD_PRELOAD is ignored again, since is it a Set-UID program.

Task8: Invoking external programs using system() versus execve()

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task8.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task8.c -o task8
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root task8
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 task8
[09/07/19]seed@VM:~/.../Lab1$ gedit empty.txt
```

```
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root:root empty.txt
[09/07/19]seed@VM:~/.../Lab1$ /bin/ls -l empty.txt
-rw-rw-r-- 1 root root 0 Sep  7 19:25 empty.txt
[09/07/19]seed@VM:~/.../Lab1$ su seed1
Password:
seed1@VM:/home/seed/Desktop/Lab1$ ./task8 "empty.txt;rm empty.txt"

rm: remove write-protected regular empty file 'empty.txt'? y
rm: cannot remove 'empty.txt': Permission denied
seed1@VM:/home/seed/Desktop/Lab1$
```

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task8.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task8.c -o task8
task8.c: In function ‘main’:
task8.c:18:2: warning: implicit declaration of function ‘execve’ [
-Wimplicit-function-declaration]
  execve(v[0], v, NULL);
  ^
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root task8
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 task8
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root:root empty.txt
[09/07/19]seed@VM:~/.../Lab1$ /bin/ls -l empty.txt
-rw-rw-r-- 1 root root 0 Sep  7 19:25 empty.txt
[09/07/19]seed@VM:~/.../Lab1$ su seed1
Password:
seed1@VM:/home/seed/Desktop/Lab1$ ./task8 "empty.txt;rm empty.txt"

/bin/cat: 'empty.txt;rm empty.txt': No such file or directory
seed1@VM:/home/seed/Desktop/Lab1$ ./task8 empty.txt;rm empty.txt
rm: remove write-protected regular empty file 'empty.txt'? y
rm: cannot remove 'empty.txt': Permission denied
seed1@VM:/home/seed/Desktop/Lab1$ gedit task8.c
No protocol specified
Failed to connect to Mir: Failed to connect to server socket: Perm
ission denied
Unable to init server: Could not connect: Connection refused

(gedit:3204): Gtk-WARNING **: cannot open display: :0
seed1@VM:/home/seed/Desktop/Lab1$
```

In this task, we are creating a program with system command first and making it a Set-UID program. We then create a file that is owned by root and is an important file with no ‘write’ privileges to other users. Now, we login into seed1 user account and execute the program. The program displays the contents of the file and also deletes the file because of the ‘rm’ command which is after the ‘;’. Now we modify the program to have the ‘execve’ command instead of the system command and the same steps are repeated. When we execute the program with “”, the program searches for the entire string. So, a file by that name wouldn’t exist. Now we try again without the “”, the program executes only till the ‘;’ and displays the contents of the file. The remaining part of the argument consisting of the ‘rm’ command is not executed since it does not have permissions.

Explanation:

When the system function executes, it does not execute the command directly. It calls the shell instead which in turn executes the command. So, if the program is a Set-UID program, the user will have temporary root privileges and can remove any file he wants with root privileges. Multiple commands can be passed together using the “” and ‘;’. System command calls the shell and the shell

parses the string and handles “”. Whereas, ‘execve’ does none of this. It replaces the program with the called program and passes the argument strings exactly as specified and will not interpret quotes. So when we pass the something after the ‘;’, it is treated as a new command and root privileges would have been lost by then. So, the ‘rm’ command is executed using seed1 privileges and hence wasn’t able to delete the file.

Task 9: Capability Leaking

Observation:

```
[09/07/19]seed@VM:~/.../Lab1$ gedit task9.c
[09/07/19]seed@VM:~/.../Lab1$ gcc task9.c -o task9
task9.c: In function ‘main’:
task9.c:18:2: warning: implicit declaration of function ‘sleep’ [-
Wimplicit-function-declaration]
  sleep(1);
  ^
task9.c:21:2: warning: implicit declaration of function ‘setuid’ [
-Wimplicit-function-declaration]
  setuid(getuid());  /* getuid() returns the real uid */
  ^
task9.c:21:9: warning: implicit declaration of function ‘getuid’ [
-Wimplicit-function-declaration]
  setuid(getuid());  /* getuid() returns the real uid */
         ^
task9.c:22:6: warning: implicit declaration of function ‘fork’ [-W
implicit-function-declaration]
  if (fork())
      ^
task9.c:24:5: warning: implicit declaration of function ‘close’ [-
Wimplicit-function-declaration]
     close (fd);
     ^
task9.c:32:5: warning: implicit declaration of function ‘write’ [-
Wimplicit-function-declaration]
     write (fd, "Malicious Data\n", 15);
     ^
```

```
[09/07/19]seed@VM:~/.../Lab1$ sudo chown root task9
[09/07/19]seed@VM:~/.../Lab1$ sudo chmod 4755 task9
[09/07/19]seed@VM:~/.../Lab1$ /bin/ls -l task9
-rwsr-xr-x 1 root seed 7640 Sep  7 19:35 task9
[09/07/19]seed@VM:~/.../Lab1$ sudo su root
root@VM:/home/seed/Desktop/Lab1# cd /etc
root@VM:/etc# gedit zzz

(gedit:3266): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:
org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.Sess
ionManager was not provided by any .service files

** (gedit:3266): WARNING **: Set document metadata failed: Setting
 attribute metadata::gedit-spell-enabled not supported

** (gedit:3266): WARNING **: Set document metadata failed: Setting
 attribute metadata::gedit-encoding not supported

** (gedit:3266): WARNING **: Set document metadata failed: Setting
 attribute metadata::gedit-position not supported
root@VM:/etc# cat zzz
This is an important file
root@VM:/etc# /bin/ls -l zzz
-rw-r--r-- 1 root root 26 Sep  7 19:37 zzz
root@VM:/etc# exit
exit
[09/07/19]seed@VM:~/.../Lab1$ ./task9
[09/07/19]seed@VM:~/.../Lab1$ cat /etc/zzz
This is an important file
Malicious Data
[09/07/19]seed@VM:~/.../Lab1$
```

Here in this task, we are creating a program task9.c, the we compile it and make it a Set-UID program owned by root. Then we login as root and create a program ‘zzz’ in the ‘etc’ directory. We exit from the root account and execute the program from seed account. We find that the file ‘/etc/zzz’ is modified by appending the content of the child process into the file.

Explanation:

During fork call copies of the parent’s set of open file descriptors is inherited be the child. Each file descriptor in the child refers to the same open file description as the corresponding file descriptor in the parent. So, the privileges that the parent gained was not downgraded and hence the child could also access the file ‘/etc/zzz’. To avoid such attacks, the file descriptor needs to be closed before the fork call.