

Environment Variable and Set-UID Program Lab

Task 1: Manipulating Environment Variables

1. Use printenv or env command to print out the environment variables.

- 1) Use printenv.

```
[03/19/19]seed@VM:~$ printenv
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=52428810
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bzip2=01;31:*.bz2=01;31:*.thz=01;31:*.tbz2;XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/share:/var/lib/flatpak/exports:/usr/share:/var/lib/snapd/desktop:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-n8kB2NgftN
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
=/usr/bin/printenv
```

- 2) Use env.

```
[03/19/19]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
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=52428810
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bzip2=01;31:*.bz2=01;31:*.thz=01;31:*.tbz2;
```

```

XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/
usr/share:/var/lib/snapd/desktop:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-n8kB2NgftN
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
=/usr/bin/env

```

3) These two commands are almost the same except XAUTHORITY. It is changed according to the command used. All environment variables are printed out.

4) Use either of the following two commands to check a particular environment variable.

```

[03/19/19]seed@VM:~$ printenv LANGUAGE
en_US
[03/19/19]seed@VM:~$ env | grep LANGUAGE
LANGUAGE=en_US

```

2. Use export and unset to set or unset environment variables.

1) Use export to set environment variable. Then check whether the setting is successful with command printenv.

```

[03/19/19]seed@VM:~$ export TEST="test"
[03/19/19]seed@VM:~$ printenv TEST
test

```

2) Unset environment variable TEST. Try to print TEST again and nothing happens. Unset TEST successfully.

```

[03/19/19]seed@VM:~$ unset TEST
[03/19/19]seed@VM:~$ printenv TEST
[03/19/19]seed@VM:~$

```

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

Step 1: Compile and run the program on the guide PDF. Let its output be stored

in file child.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task2 task2.c
[03/19/19]seed@VM:~/.../3$ ./task2 > child
```

All environment variables of the child process is stored in file child.

```
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=50331658
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1582,unix/VM:/tmp/.ICE-unix/1582
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:./snap/bin
DESKTOP_SESSION=ubuntu
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
TERM=dtterm
```

Step 2: Comment out the `printenv()` statement in the child process case, and

uncomment the `printenv()` statement in the parent process case. Compile and run

the code again. Let its output be stored in file parent.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task2 task2.c
[03/19/19]seed@VM:~/.../3$ ./task2 > parent
```

All environment variables of the parent process is stored in file parent.

```
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=50331658
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1582,unix/VM:/tmp/.ICE-unix/1582
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:./snap/bin
DESKTOP_SESSION=ubuntu
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
TERM=dtterm
```

Step 3: Compare the difference of these two files using the `diff` command.


```
[03/19/19]seed@VM:~/.../3$ diff child parent
[03/19/19]seed@VM:~/.../3$
```

Nothing is outputted. File child and file parent are the same. That is, the parent's environment variables are inherited by the child process.

Task 3: Environment Variables and `execve()`

Step 1: Compile and run the program on the guide PDF. Let its output be stored in file before.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task3 task3.c
task3.c: In function 'main':
task3.c:13:5: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
    execve("/usr/bin/env", argv, NULL);
    ^
[03/19/19]seed@VM:~/.../3$ ./task3 > before
[03/19/19]seed@VM:~/.../3$
```

The file before is empty. Nothing has been printed out.

Step 2: Change the third parameter of `execve()` to `environ`. Compile and run the program again. Let its output be stored in file after.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task3 task3.c
task3.c: In function 'main':
task3.c:13:5: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
    execve("/usr/bin/env", argv, environ);
    ^
[03/19/19]seed@VM:~/.../3$ ./task3 > after
[03/19/19]seed@VM:~/.../3$
```

File after stores all the environment variables of the current process.

```
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=52428810
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XDG_CURRENT_DESKTOP=Unity
XDG_VARIANT_PLATFORM=ubuntu
XDG_VARIANT_NAME=ubuntu:seed
```

Step 3: The first parameter of `execve()` represents the program needing to be executed, the second passing parameters to the program and the third passing new environment variables to the program. If the environment variables are not sent to the executed program, the program cannot get them.

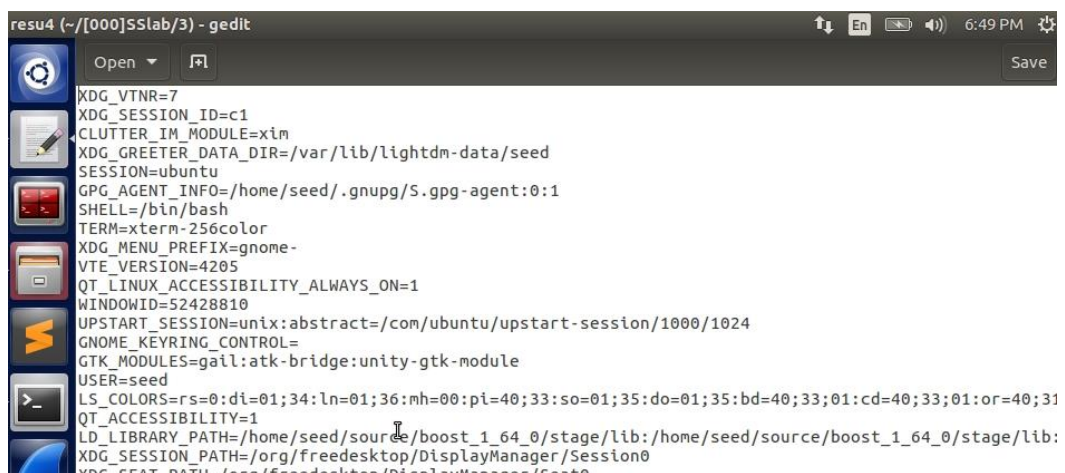
That is, the new program gets its environment variables by getting what the old program passes to it.

Task 4: Environment Variables and `system()`

Compile and run the program on the guide PDF. Let its output be stored in file `resu4`.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task4 task4.c
[03/19/19]seed@VM:~/.../3$ ./task4 > resu4
[03/19/19]seed@VM:~/.../3$
```

File `resu4` stores the environment variables of the calling process.



```
resu4 (~/[000]SSlab/3) - gedit
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=52428810
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1024
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
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
```

That is, using `system()`, the environment variables of the calling process is actually passed to the new program `/bin/sh`.

Task 5: Environment Variable and Set-UID Programs

Step 1: Use the program on the guide PDF that can print out all the environment variables in the current process.

```

#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void 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.

```

[03/19/19]seed@VM:~/.../3$ gcc -o task5 task5.c
[03/19/19]seed@VM:~/.../3$ sudo chown root task5
[03/19/19]seed@VM:~/.../3$ sudo chmod 4755 task5
[03/19/19]seed@VM:~/.../3$

```

Step 3: Set PATH, LD_LIBRARY_PATH and ANY_NAME as below.

```

[03/19/19]seed@VM:~/.../3$ export TASK5TEST="testtask5"
[03/19/19]seed@VM:~/.../3$ export LD_LIBRARY_PATH=/home/seed
[03/19/19]seed@VM:~/.../3$ export PATH=/home/seed
Command 'date' is available in '/bin/date'
The command could not be located because '/bin' is not included in
the PATH environment variable.
date: command not found
[03/19/19]seed@VM:~/.../3$

```

Run the Set-UID program from Step 2.

```

[ ]seed@VM:~/.../3$ ./task5
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205

```

Not all three variables get into the Set-UID child process. PATH is changed and TASK5TEST is added while LD_LIBRARY_PATH cannot be found.

```

XDG_CONFIG_DIRS=/etc/xdg
PATH=/home/seed
DESKTOP_SESSION=ubuntu
WINDOWID=52428810
TASK5TEST=testtask5
UPSTART_SESSION=univarshe

```

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

1. Use the following commands to link /bin/sh to zsh in order to see how our

attack works without the countermeasure in /bin/dash.

```
[03/19/19]seed@VM:~/.../3$ sudo rm /bin/sh
[sudo] password for seed:
[03/19/19]seed@VM:~/.../3$ sudo ln -s /bin/zsh /bin/sh
[03/19/19]seed@VM:~/.../3$
```

2. Use the program on the guide PDF.

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    system("ls");
    return 0;
}
```

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

program.

```
[03/19/19]seed@VM:~/.../3$ gcc -o task6 task6.c
[03/19/19]seed@VM:~/.../3$ sudo chown root task6
[03/19/19]seed@VM:~/.../3$ sudo chmod 4755 task6
[03/19/19]seed@VM:~/.../3$
```

Now run task6. It performs the function of command ls now.

```
[03/19/19]seed@VM:~/.../3$ ./task6
after child  resu4 task2.c task3.c task4.c task5.c task6.c
before parent task2 task3 task4 task5 task6
```

3. As system() needs the environment variable SHELL, to execute my code such as task5.c rather than /bin/ls, the directory of my executable file should be added in the front of SHELL.

```
[03/19/19]seed@VM:~/.../3$ export SHELL=/home/[000]SSlab/3:$SHELL
```

Then modify the file name of which system() executes in task6.c.

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    system("task5");
    return 0;
}
```

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

program. Run it again.


```

[03/19/19]seed@VM:~/.../3$ gcc -o task6 task6.c
[03/19/19]seed@VM:~/.../3$ sudo chown root task6
[03/19/19]seed@VM:~/.../3$ sudo chmod 4755 task6
[03/19/19]seed@VM:~/.../3$ ./task6
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
SHELL=/home/[000]SSlab/3:/bin/bash
TERM=xterm-256color
XDG_MENU_PREFIX=gnome-
VTE_VERSION=4205

```

It performs the function of program task5 now. The code must be running with the root privilege.

4. Function `system()` executes with the environment variable `SHELL`, which points out the directory of the file this function would execute. If the value of `SHELL` and the name of according executable file is changed, `system()` will execute another file which it finds in the present `SHELL` directory.

If the vulnerable program cannot be changed, just create a new program with the same name as "ls" in the changed `SHELL` directory.

Task 7: The `LD_PRELOAD` Environment Variable and Set-UID Programs

Step 1: Examine how the environment variables influence the behavior of dynamic loader/linker when running a normal program.

Create `mylib.c` which overrides 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");
}

```

Compile the above program using the following commands.

```

[03/19/19]seed@VM:~/.../3$ gcc -fPIC -g -c mylib.c
[03/19/19]seed@VM:~/.../3$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc

```


Set the LD_PRELOAD environment variable as below.

```
[03/19/19]seed@VM:~/.../3$ export LD_PRELOAD=./libmylib.so.1.0.1
```

Compile the following program myprog, and in the same directory as the above dynamic link library libmylib.so.1.0.1.

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

```
[03/19/19]seed@VM:~/.../3$ gcc -o myprog myprog.c
myprog.c: In function 'main':
myprog.c:4:5: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
    ^
```

Step 2: Run myprog under the following conditions and observe what happens.

1. Make myprog a regular program, and run it as a normal user.

```
[03/20/19]seed@VM:~/.../3$ gcc -o myprog myprog.c
myprog.c: In function 'main':
myprog.c:4:5: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
    ^
[03/20/19]seed@VM:~/.../3$ ./myprog
I am not sleeping!
```

"I am not sleeping!" is shown. Thus, the sleep() function called by myprog is the function defined in mylib which overrides the system sleep() function.

2. Make myprog a Set-UID root program, and run it as a normal user.

```
[03/20/19]seed@VM:~/.../3$ sudo chown root myprog
[sudo] password for seed:
[03/20/19]seed@VM:~/.../3$ sudo chmod 4755 myprog
[03/20/19]seed@VM:~/.../3$ ./myprog
[03/20/19]seed@VM:~/.../3$
```

No words is shown, which means the called sleep() function at present is the system function without modification.

3. Make myprog a Set-UID root program, export the LD_PRELOAD

environment variable again in the root account and run it.

```
[03/20/19]seed@VM:~/.../3$ sudo chown root myprog
[03/20/19]seed@VM:~/.../3$ sudo chmod 4755 myprog
[03/20/19]seed@VM:~/.../3$ sudo su
root@VM:/home/seed/[000]SSlab/3# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/[000]SSlab/3# ./myprog
I am not sleeping!
root@VM:/home/seed/[000]SSlab/3# exit
exit
[03/20/19]seed@VM:~/.../3$
```

When running in the root account, the program outputs "I am not sleeping!" , the result of the overridden sleep() function.

4. Create user accounts "user1" .

```
/3$ sudo useradd -d /usr/user1 -m user1
```

Make myprog a Set-UID seed program.

```
[04/03/19]seed@VM:~/.../3$ sudo chown seed myprog
[04/03/19]seed@VM:~/.../3$ sudo chgrp seed myprog
```

Export the LD_PRELOAD environment variable again in user1 account1.

```
[04/03/19]seed@VM:~/.../3$ sudo su user1
user1@VM:/home/seed/[000]SSlab/3$ printenv LD_PRELOAD
user1@VM:/home/seed/[000]SSlab/3$ export user1 LD_PRELOAD=./libmylib.so.1.0.1
```

Run myprog in seed account.

```
[04/03/19]seed@VM:~/.../3$ printenv LD_PRELOAD
[04/03/19]seed@VM:~/.../3$ ./myprog
```

The word "I am not sleeping!" is not shown. The sleep() function is not the newly defined function.

Step 3: Figure out the main causes of different phenomena.

These phenomena may result from different application of the environment variable LD_PRELOAD. Only when the owner of the program and the exporting place of LD_PRELOAD are the same can a user run the program with the overridden function.

This demonstration can be proved by the check of environment variable LD_PRELOAD in condition 4. Created by one user, it cannot be gotten by another.

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

Create file tsk8 with content "testing" and set its properties by command "chmod 444 tsk8" .

Step 1: Compile the program on the guide PDF and make it a root-owned Set-UID program.

```
.../3$ gcc -o task8 task8.c
.../3$ sudo chown root task8
.../3$ sudo chmod 4755 task8
```

Run the program to open file tsk8.

```
[04/03/19]seed@VM:~/.../3$ ./task8 "tsk8"
testing
```

To compromise the integrity of the system, or just remove a file that is not writable to the user, try following operations.

```
[04/03/19]seed@VM:~/.../3$ ./task8 "tsk8;rm tsk8 -rf"
testing
[04/03/19]seed@VM:~/.../3$ find tsk8
find: 'tsk8': No such file or directory
```

Add ";rm tsk8 -rf" to the tail of the file name "tsk8" , then this file will be deleted , as the root privilege is not revoked by the system which suggests the competence in file management.

Step 2: Comment out the system(command) statement, and uncomment the execve() statement; the program will use execve() to invoke the command. Compile the program, and make it a root-owned Set-UID. Run same commands as above.

```

[04/03/19]seed@VM:~/.../3$ gcc -o task8 task8.c
task8.c: In function 'main':
task8.c:21:6: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
    execve(v[0], v, NULL);
    ^
[04/03/19]seed@VM:~/.../3$ sudo chown root task8
[04/03/19]seed@VM:~/.../3$ sudo chmod 4755 task8
[04/03/19]seed@VM:~/.../3$ ./task8 "tsk8"
testing
[04/03/19]seed@VM:~/.../3$ ./task8 "tsk8;rm tsk8 -rf"
/bin/cat: 'tsk8;rm tsk8 -rf': No such file or directory

```

This file can be read but cannot be deleted in the same way.

The reason for this phenomenon can be that unlike `system()`, function `execve()` does not search for any other command. It just treats the input as a string of file name and try to find it, rather than try finding the command and then execute it with root privilege as `system()` does.

Task 9: Capability Leaking

Compile the program on the guide PDF, change its owner to root, and make it a Set-UID program.

```

[03/20/19]seed@VM:~/.../3$ gcc -o task9 task9.c
task9.c: In function 'main':
task9.c:20:5: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
    ^
task9.c:24:5: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
[03/20/19]seed@VM:~/.../3$ sudo chown root task9
[03/20/19]seed@VM:~/.../3$ sudo chmod 4755 task9

```

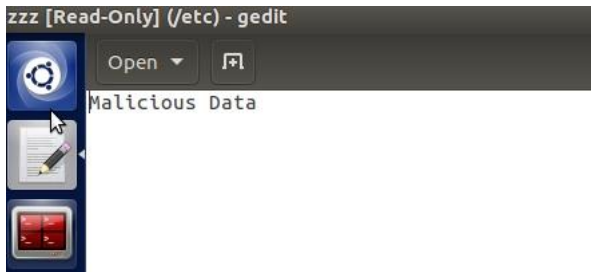
Before the attack, file `/etc/zzz` is empty and read-only.



Run the program as a normal user.

```
[03/20/19] seed@VM:~/.../3$ ./task9  
[03/20/19] seed@VM:~/.../3$
```

Without the affirmation of root privilege, "Malicious Data" is written into this file successfully.



This phenomenon may be caused by the insufficient privilege-downgrading. File zzz is opened under root privilege as the program calls sleep() later, which needs higher privilege. However, after the end of calling sleep(), the root privilege of modifying important system files is not revoked. Thus, the file zzz can be modified by this program.