Cybersecurity

Environment Variable and Set-UID Program Lab

Task 1: Manipulating Environment Variables

printenv/env command to print out the environment variables.
export and unset to set or unset environment variables.

```
fabst@fab:~$ printenv PWD
/home/fabst
fabst@fab:~$ env | grep PWD
PWD=/home/fabst
fabst@fab:~$ unset PWD
fabst@fab:~$ env | grep PWD
fabst@fab:~$ export PWD=/home/fabst
fabst@fab:~$ env | grep PWD
PWD=/home/fabst
fabst@fab:~$
```

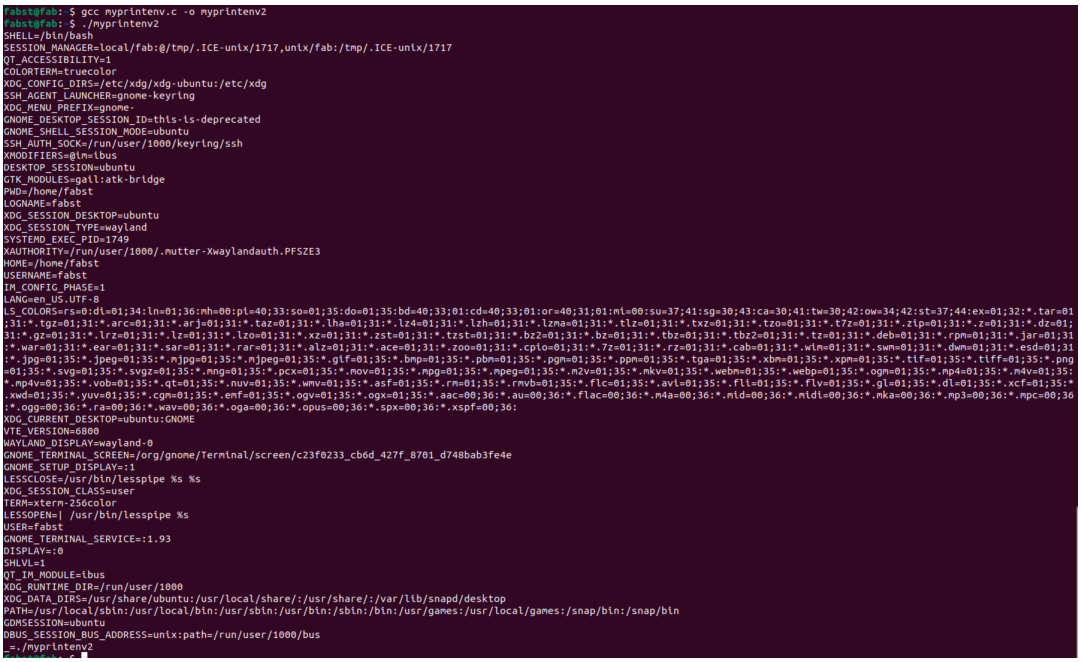
Task 2: Passing Environment Variables from Parent Process to Child Process

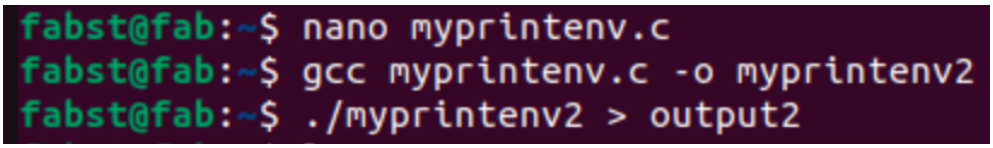
The compilation and running of the myprintenv.c program containing the child process printenv gives the output below. This output shows all the environment variables of the child process.

```
fabst@fab: $ gcc myprintenv.c -o myprintenv
fabst@fab: $ ./myprintenv
SHELL=/bin/bash
SESSION_MANAGER=local/fab:@/tmp/.ICE-unix/1717,unix/fab:/tmp/.ICE-unix/1717
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
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
GTK_MODULES=gail:atk-bridge
PWD=/home/fabst
LOGNAME=fabst
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1749
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.PFSZE3
HOME=/home/fabst
USERNAME=fabst
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
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:*.webp=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=01;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=6800
WAYLAND_DISPLAY=wayland-0
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/c23f0233_cb6d_427f_8701_d748bab3fe4e
GNOME_SETUP_DISPLAY=:1
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=fabst
GNOME_TERMINAL_SERVICE=:1.93
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
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:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=./myprintenv
```

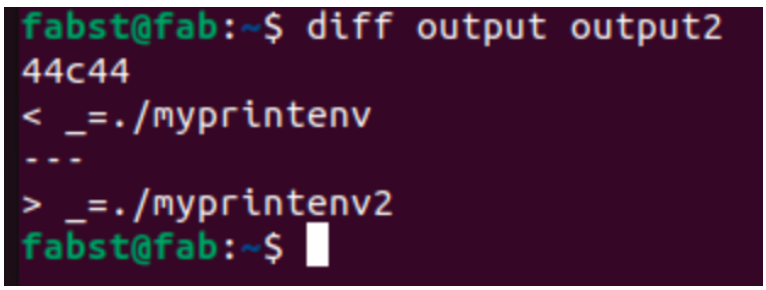
```
fabst@fab:~$ man fork
fabst@fab:~$ nano myprintenv.c
fabst@fab:~$ gcc myprintenv.c -o myprintenv
fabst@fab:~$ ./myprintenv > output
```

A similar output displays the environment variables of the parent process from the same program.





Comparing the two output files using the diff command.



The difference of the two files using the diff command is interpreted as: 44c44 means that the 44th line in the left file is changed (c) to the 44th line in the right file. The < stands for the left file and > stands for the right file showing the changed content.

Task 3: Environment variables and execve()

The images below show the output from compiling and running myenv.c program with NULL in the execve line (myenv) and environ in the execve line (myenv2).

```
fabst@fabjona:~$ nano myenv.c
fabst@fabjona:~$ gcc myenv.c -o myenv
fabst@fabjona:~$ ./myenv
fabst@fabjona:~$
```

```
fabst@fab:~$ nano myenv.c
fabst@fab:~$ gcc myenv.c -o myenv2
fabst@fab:~$ ./myenv2
SHELL=/bin/bash
SESSION_MANAGER=local/fab:@/tmp/.ICE-unix/1717,unix/fab:/tmp/.ICE-unix/1717
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
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
GTK_MODULES=gail:atk-bridge
PWD=/home/fabst
LOGNAME=fabst
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1749
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.PFSZE3
HOME=/home/fabst
USERNAME=fabst
IM_CONFIG_PHASE=1
```

The difference in output between the two changes in the file is that myenv did not output anything, while myenv2 did have an output. This happened because the first file (myenv) had NULL as an argument in the print line, but the second file (myenv2) has environ as an argument. Considering that both arguments are the third argument means that, that argument specifies the environment variable of the current process. In conclusion, the third argument of the print line gets the environment variable of that process.

Task 4: Environment variables and system()

The following shows the output of the compilation and execution of the systemfunction.c program, which includes the system() function, which passes the environment variables to the called function /bin/sh. By using the system() function, we do not need to pass any environment variables to the program.

```
fabst@fab:~$ gcc systemfunction.c -o systemfunction
fabst@fab:~$ ./systemfunction
LESSOPEN=| /usr/bin/lesspipe %s
USER=fabst
XDG_SESSION_TYPE=wayland
SHLVL=1
HOME=/home/fabst
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
SYSTEMD_EXEC_PID=1749
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
WAYLAND_DISPLAY=wayland-0
LOGNAME=fabst
_=./systemfunction
XDG_SESSION_CLASS=user
USERNAME=fabst
TERM=xterm-256color
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
SESSION_MANAGER=local/fab:@/tmp/.ICE-unix/1717,unix/fab:/tmp/.ICE-unix/1717
XDG_MENU_PREFIX=gnome-
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/caba73c4_a881_48c1_ac58_298468bc9d1e
GNOME_SETUP_DISPLAY=:1
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/.mutter-Xwaylandauth.PFSZE3
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:*.s
wm=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:*
.webp=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=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.f
li=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=:1.142
SSH_AGENT_LAUNCHER=gnome-keyring
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
QT_IM_MODULE=ibus
PWD=/home/fabst
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
VTE_VERSION=6800
```

Task 5: Environment variables and Set-UID Programs

After compiling the given program, we change the ownership to root and make it a Set-UID program. Then we use the export command to set the following environment variables:

- PATH
- LD_LIBRARY_PATH
- ANY_NAME (environment variable defined by me, in this case called CHOC)

```
fabst@fabjona:~$ gcc pev.c -o pevinprocess
fabst@fabjona:~$ sudo chown root pevinprocess
fabst@fabjona:~$ sudo chmod 4755 pevinprocess
fabst@fabjona:~$ export PATH
fabst@fabjona:~$ export LD_LIBRARY_PATH
fabst@fabjona:~$ export CHOC
fabst@fabjona:~$
```

After running the above program, it can be seen that the child process inherits the PATH and CHOC environment variable, but there is no LD environment variable.

```
CHOC=/home/fabst
USER=fabst
GNOME_TERMINAL_SERVICE=:1.110
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
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:/snap/bin
```

This shows that the child process might not inherit all the environment variables of the parent process. This is a security mechanism implemented by the dynamic linker. The LD_LIBRARY_PATH is ignored because the real id and the effective id are different.

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

First we change the PATH environment variable. Then we compile the given program and change ownership to root and make it a Set-UID program.

```
fabst@fabjona:~$ export PATH=/home/fabst:$PATH
fabst@fabjona:~$ nano task6.c
fabst@fabjona:~$ gcc task6.c -o task6
task6.c: In function ‘main’:
task6.c:2:3: warning: implicit declaration of function ‘system’ [-Wimplicit-function-declaration]
    2 |   system("ls");
      |   ^~~~~~
fabst@fabjona:~$ sudo chown root task6
[sudo] password for fabst:
fabst@fabjona:~$ sudo chmod 4755 task6
fabst@fabjona:~$ ./task6
Desktop  Documents  Downloads  Music  pev.c  pevinprocess  Pictures  Public  task6  task6.c  Templates  Videos
fabst@fabjona:~$ ls -l task6
-rwsr-xr-x 1 root fabst 8880 Oct 31 17:14 task6
fabst@fabjona:~$
```

The following shows the contents of my ls program:

```
 GNU nano 6.2                                    ls.c
#include <stdio.h>

int main(){
  printf("It's FRIDAY!!!!!!!\n");
}
```

The following shows that I have compiled and executed my program, instead of the regular ls program.

```
fabst@fabjona:~$ gcc ls.c -o ls
fabst@fabjona:~$ ./ls
It's FRIDAY!!!!!!!
fabst@fabjona:~$
```

This shows that the PATH environment variables can be changed to point to a desired folder and execute the user-defined programs, which could be malicious. By changing the PATH value to a folder containing a file with the same name as the one specified in the program, the attacker or another person can run a malicious code with root privileges because the file it’s a root-owned Set-UID program. Because of this using the system function and relative path could lead to attacks.

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

First we create a program called mylib.c, which overrides the sleep() function in libc.
Then we compile the program, set the LD_PRELOAD environment variable. Finally we create and compile the program myprog. First, we run myprog as a normal user.

```
fabst@fabjona:~$ nano mylib.c
fabst@fabjona:~$ gcc -fPIC -g -c mylib.c
fabst@fabjona:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
fabst@fabjona:~$ export LD_PRELOAD=./libmylib.so.1.0.1
fabst@fabjona:~$ nano myprog.c
fabst@fabjona:~$ gcc myprog.c -o program7
fabst@fabjona:~$ ./program7
I am not sleeping!
fabst@fabjona:~$
```

In the above output we can see that the program called the sleep function defined by us.

Then, we make this program a Set_UID root program and run it.

```
fabst@fabjona:~$ sudo chown root program7
fabst@fabjona:~$ sudo chmod 4755 program7
fabst@fabjona:~$ ./program7
fabst@fabjona:~$
```

In the output above, we can see that after changing the ownership and making it a Set-UID program, the program calls the sleep() function and not the function that we made.
After compiling the program with the different scenarios given in the lab, I noticed that in some cases the sleep() function was called and in other cases the sleepy function defined by us was called. This indicates that the LD_PRELOAD is present only when the real and effective ID are the same. This is due to the Set-UID program's security mechanism. In the first, third and fourth case the LD_PRELOAD was present since the owner and the account executing the file were the same. In the second case, the effective ID was root and the effective ID was of fabst, and because of this the LD_PRELOAD variable and dropped and the system-defined sleep() function was called.

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

We compile the given program and make it a Set-UID root program.

```
fabst@fabjona:~$ nano catall.c
fabst@fabjona:~$ gcc catall.c -o task8
fabst@fabjona:~$ sudo chown root task8
[sudo] password for fabst:
fabst@fabjona:~$ sudo chmod 7455 task8
fabst@fabjona:~$ sudo chmod 4755 task8
```

If I were Bob, I would be able to compromise the integrity of the system. The program will run normally if we provide the file to be read, but if we provide a malicious input, the program will first read the file and then it will run it as a command. So, if Bob runs the rm command, he is able to remove a file. Even though Bob did not have write permissions, he could easily remove a file by just assuming root privileges. This happens because the system() call does not separate the command and user input.

Then in the program we comment out the system() statement and uncomment execve() statement and compile it. We make this program a root-owned Set-UID program.

```
fabst@fabjona:~$ nano catall.c
fabst@fabjona:~$ gcc catall.c -o task8execve
fabst@fabjona:~$ ./task8execve
Please type a file name.
fabst@fabjona:~$ sudo chown root task8execve
fabst@fabjona:~$ sudo chmod 4755 task8execve
```

In the output above we can see that if we try to perform the attacks from the first part, they are not successful because the user input is considered a file name. The difference between the two statements is that the system() statement is constructed using strings inputted while executing. Meanwhile when using execve(), the input is directly entered as the second parameter of the function.

Task 9: Capability Leaking

We compile the given program and make it a Set-UID root-owned program. We also create the zzz file in the /etc folder. Then, we run the program as a normal user.

```
fabst@fabjona:~$ nano cap_leak.c
fabst@fabjona:~$ gcc cap_leak.c -o task9
fabst@fabjona:~$ sudo chown root task9
[sudo] password for fabst:
fabst@fabjona:~$ sudo chmod 4755 task9
fabst@fabjona:~$ ls -l task9
-rwsr-xr-x 1 root fabst 9232 Oct 31 19:02 task9
fabst@fabjona:~$ ./task9
Cannot open /etc/zzz
fabst@fabjona:~$ cd etc
bash: cd: etc: No such file or directory
fabst@fabjona:~$ cd /etc
fabst@fabjona:/etc$ ll zzz
ls: cannot access 'zzz': No such file or directory
fabst@fabjona:/etc$ touch zzz
touch: cannot touch 'zzz': Permission denied
fabst@fabjona:/etc$ sudo touch zzz
fabst@fabjona:/etc$ ll zzz
-rw-r--r-- 1 root root 0 Oct 31 19:06 zzz
```

As seen in the above image, to be able to write to the zzz file we need root privileges. We can write to it as a normal user but only if we have root privileges. When we run the program, the output prints out the file descriptor value.

```
fabst@fabjona:~$ ./task9
fd is 3
$
```

This dhows that there is a capability leak even though the program dropped the root privileges. The shell still has write access to the zzz file in the /etc because the file was opened while the process still had root privileges. The file descriptor is inherited from the child process.