

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I



BÀI BÁO CÁO THỰC HÀNH SỐ 5

Môn học: PHÂN TÍCH MÃ ĐỘC

Biến môi trường và chương trình Set-UID

Tên sinh viên: Ninh Chí Hường

Mã sinh viên: B20DCAT094

Nhóm lớp : 02

Giảng viên hướng dẫn: PGS.TS Đỗ Xuân Chợt

HÀ NỘI, THÁNG 9/2023

Table of Contents

1. Giới thiệu:.....	3
2. Khái quát chung:	3
3. Các bài tập:	3
Nhiệm vụ 1: Thao tác với các biến môi trường:.....	3
Nhiệm vụ 2: Kế thừa biến môi trường từ tiến trình cha:.....	5
Nhiệm vụ 3: các biến môi trường và hàm <code>execve()</code>	6
Nhiệm vụ 4: các biến môi trường và hàm <code>system()</code>	7
Nhiệm vụ 5: các biến môi trường và các chương trình Set-UID	8
Nhiệm vụ 6: hàm <code>system()</code> và các chương trình Set-UID	9
Nhiệm vụ 7: biến môi trường <code>LD_PRELOAD</code> và các chương trình Set-UID.....	11
Nhiệm vụ 8: Khả năng rò rỉ.....	12
II. Checkwork	13

1. Giới thiệu:

- Biến môi trường là giá trị động ảnh hưởng đến phần mềm và tiến trình hoạt động trên server. Biến môi trường – environment variable có trên mọi hệ điều hành và có nhiều loại khác nhau. Biến môi trường có thể được tạo, chỉnh sửa, lưu hay xóa.
- Biến môi trường của linux chứa thông tin hệ thống, mà sẽ chuyển dữ liệu đó đi cho phần mềm trong shells hoặc sub-shells.
- Trên hệ điều hành Linux, chương trình Set-UID (Set User ID) là một loại chương trình được thiết lập với quyền thực thi của chủ sở hữu của chương trình thay vì quyền của người dùng hiện tại khi nó được thực thi. Điều này cho phép người dùng thực thi chương trình với quyền đặc biệt, như quyền quản trị hệ thống.
- Khi một tệp thực thi trên Linux được thiết lập Set-UID, nó có một bit Set-UID đặt là 1 trong quyền thực thi của nó. Khi một người dùng thực thi chương trình Set-UID, quyền của chương trình được tạm thời thay đổi thành quyền của chủ sở hữu của chương trình. Điều này cho phép người dùng thực thi các hoạt động đặc biệt mà họ không có thể lực thông thường.
- Chương trình Set-UID thường được sử dụng để thực hiện các tác vụ hệ thống quan trọng hoặc nhạy cảm, như quản lý người dùng, cấu hình hệ thống, và giao tiếp với phần cứng. Một số chương trình Set-UID phổ biến trên Linux bao gồm passwd (để thay đổi mật khẩu người dùng), su (để chuyển đổi sang tài khoản người dùng khác), và sudo (để thực thi lệnh với quyền đặc biệt).

2. Khái quát chung:

- Mục tiêu của bài thực hành này là giúp sinh viên hiểu được cách biến môi trường ảnh hưởng đến hành vi của các chương trình và hệ thống. Biến môi trường là một tập hợp các giá trị được gán tên và có tính động, có thể ảnh hưởng đến cách mà các tiến trình đang chạy sẽ hoạt động trên máy tính. Chúng được sử dụng bởi hầu hết các hệ điều hành, kể từ khi chúng được giới thiệu vào Unix vào năm 1979. Mặc dù biến môi trường ảnh hưởng đến hành vi của chương trình, cách chúng hoạt động thường không được hiểu rõ bởi nhiều lập trình viên. Kết quả là, nếu một chương trình sử dụng biến môi trường, nhưng nhà lập trình không biết rằng chúng được sử dụng, chương trình có thể có lỗ hổng bảo mật. Trong bài thực hành này, sinh viên sẽ hiểu cách biến môi trường hoạt động, cách chúng được truyền từ tiến trình cha đến tiến trình con và cách chúng ảnh hưởng đến hành vi của hệ thống/ chương trình. Chúng tôi đặc biệt quan tâm đến cách biến môi trường ảnh hưởng đến hành vi của các chương trình Set-UID (thường được sử dụng bởi các chương trình đặc quyền).

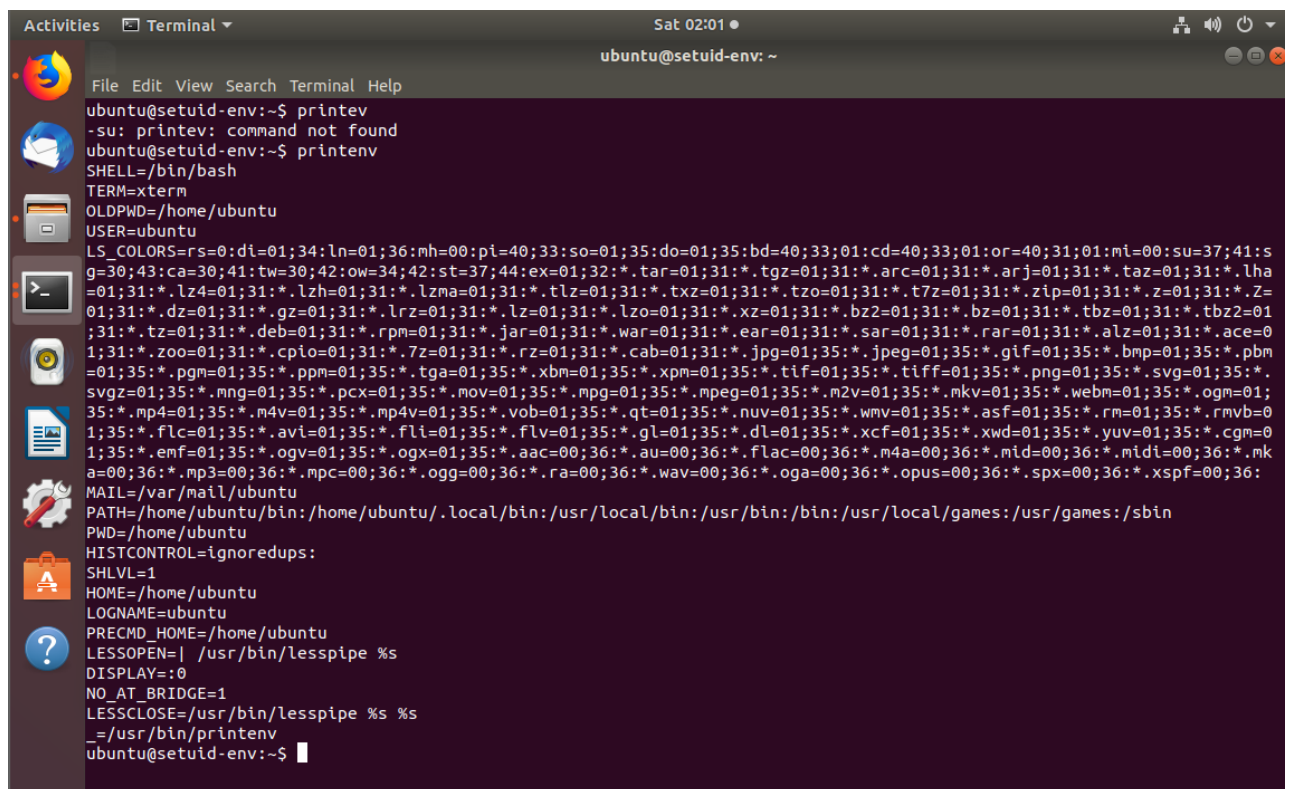
3. Các bài tập:

Nhiệm vụ 1: Thao tác với các biến môi trường:

- Trong nhiệm vụ này, ta sẽ tìm hiểu các lệnh có thể được sử dụng để thiết lập và

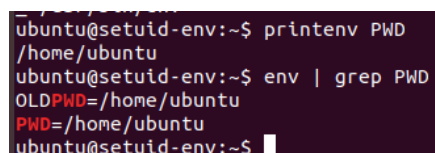
hủy bỏ các biến môi trường trên hệ thống Unix. Sử dụng Bash trong bài thực hành này. Shell mặc định mà người dùng sử dụng được thiết lập trong tệp `/etc/passwd` (trường cuối cùng của mỗi mục). Ta có thể thay đổi điều này sang một chương trình shell khác bằng lệnh `chsh` (không thực hiện điều này trong bài thực hành này). Ta cần thực hiện các nhiệm vụ sau đây:

- Sử dụng lệnh `printenv` hoặc `env` để in ra danh sách các biến môi trường:



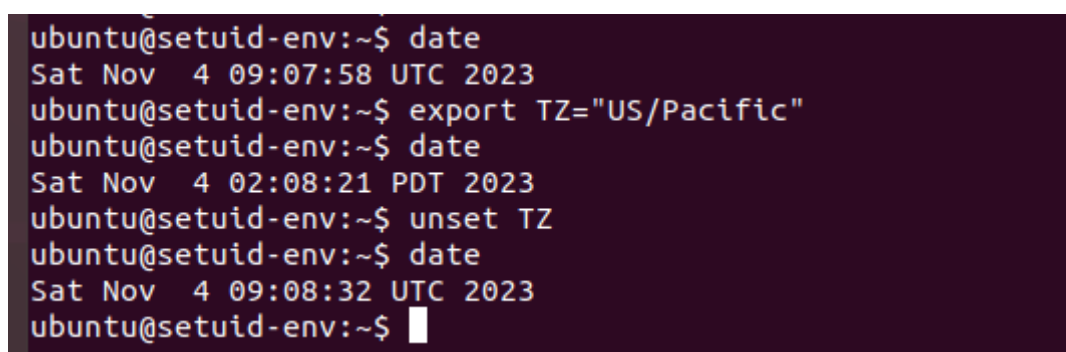
```
ubuntu@setuid-env:~$ printenv
-su: printenv: command not found
ubuntu@setuid-env:~$ printenv
SHELL=/bin/bash
TERM=xterm
OLDPWD=/home/ubuntu
USER=ubuntu
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.taz=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:*.cab=01;31:*.jpg=01;35:*.jpeg=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=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:
MAIL=/var/mail/ubuntu
PATH=/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/sbin
PWD=/home/ubuntu
HISTCONTROL=ignoredups:
SHLVL=1
HOME=/home/ubuntu
LOGNAME=ubuntu
PRECMD_HOME=/home/ubuntu
LESSOPEN=| /usr/bin/lesspipe %s
DISPLAY=:0
NO_AT_BRIDGE=1
LESSCLOSE=/usr/bin/lesspipe %s %s
_=/usr/bin/printenv
ubuntu@setuid-env:~$
```

- Một số biến môi trường cụ thể, chẳng hạn như `PWD`, có thể sử dụng “`printenv PWD`” hoặc “`env | grep PWD`”.



```
ubuntu@setuid-env:~$ printenv PWD
/home/ubuntu
ubuntu@setuid-env:~$ env | grep PWD
OLDPWD=/home/ubuntu
PWD=/home/ubuntu
ubuntu@setuid-env:~$
```

- Sử dụng lệnh `export` và `unset` để thiết lập hoặc hủy bỏ các biến môi trường



```
ubuntu@setuid-env:~$ date
Sat Nov  4 09:07:58 UTC 2023
ubuntu@setuid-env:~$ export TZ="US/Pacific"
ubuntu@setuid-env:~$ date
Sat Nov  4 02:08:21 PDT 2023
ubuntu@setuid-env:~$ unset TZ
ubuntu@setuid-env:~$ date
Sat Nov  4 09:08:32 UTC 2023
ubuntu@setuid-env:~$
```

Nhiệm vụ 2: Kế thừa biến môi trường từ tiến trình cha:

- Trong nhiệm vụ này, ta sẽ tìm hiểu cách các biến môi trường được kế thừa bởi các tiến trình con từ tiến trình cha trên hệ thống Unix. Trong Unix, hàm `fork()` tạo ra một tiến trình mới bằng cách sao chép tiến trình gọi. Tiến trình mới, được gọi là tiến trình con, là một bản sao chính xác của tiến trình gọi, được gọi là tiến trình cha; tuy nhiên, một số thứ không được kế thừa bởi tiến trình con (xem mô tả của hàm `fork()` bằng cách nhập lệnh sau: `man fork`). Trong nhiệm vụ này, cần tìm hiểu xem các biến môi trường của tiến trình cha có được kế thừa bởi tiến trình con hay không?
 - Bước 1. Thực hiện biên dịch và chạy chương trình `printenv.c` (nội dung được liệt kê bên dưới) và mô tả những gì quan sát được. Bởi vì đầu ra chứa nhiều chuỗi, sinh viên nên lưu đầu ra vào một tệp, ví dụ bằng cách sử dụng `a.out > child` (giả sử `a.out` là tên tệp thực thi của sinh viên).



```
ubuntu@setuid-env: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: printenv.c Modified

#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
extern char ** environ;
void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
void main()
{
    pid_t childPid;
    printf("Environment variables for fork:\n");
    switch(childPid = fork()) {
        case 0: /* child process */
            // printenv();
            exit(0);
        default: /* parent process */
            printenv();
            exit(0);
    }
}
```

```
GNU nano 2.5.3 File: printenv.c
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
extern char ** environ;
void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
void main()
{
    pid_t childPid;
    printf("Environment variables for fork:\n");
    switch(childPid = fork()) {
        case 0: /* child process */
            printenv();
            exit(0);
        default: /* parent process */
            //printenv();
            exit(0);
    }
}
```

Bước 3. So sánh sự khác biệt giữa hai tệp bằng lệnh diff.

```
ubuntu@setuid-env:~$ ./f1 > f1.txt
ubuntu@setuid-env:~$ ./f2 > f2.txt
ubuntu@setuid-env:~$ diff f1.txt f2.txt
18c18
< _=./f1
---
> _=./f2
ubuntu@setuid-env:~$
```

Kết luận: kết quả diff không hiển thị bất kỳ sự khác biệt nào giữa hai tệp đầu ra. Điều này cho thấy rằng các biến môi trường của tiến trình cha được kế thừa bởi tiến trình con trong hệ thống Unix.

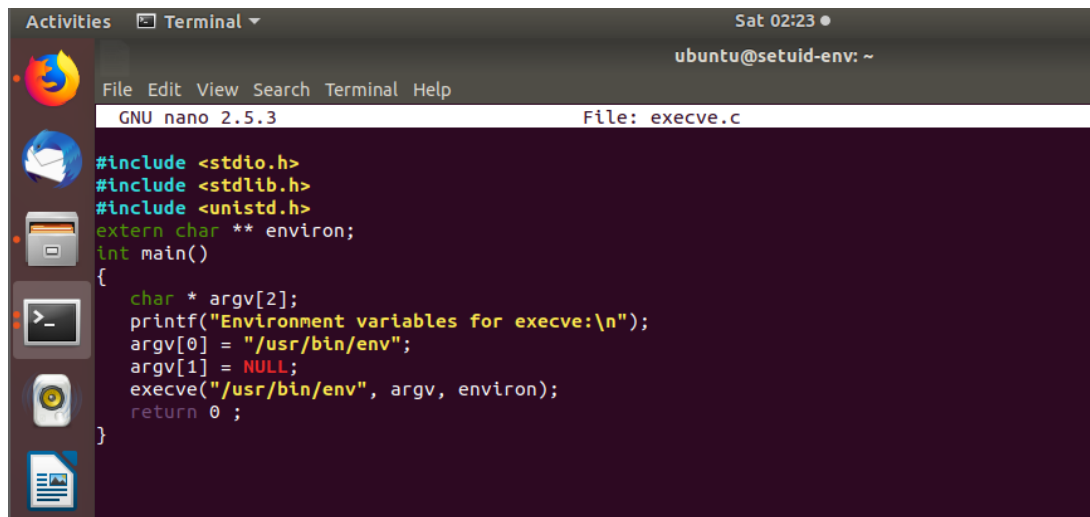
Nhiệm vụ 3: các biến môi trường và hàm execve()

Nhiệm vụ 3: Các biến môi trường và hàm execve()

Bước 1. Thực hiện biên dịch và chạy chương trình execve.c

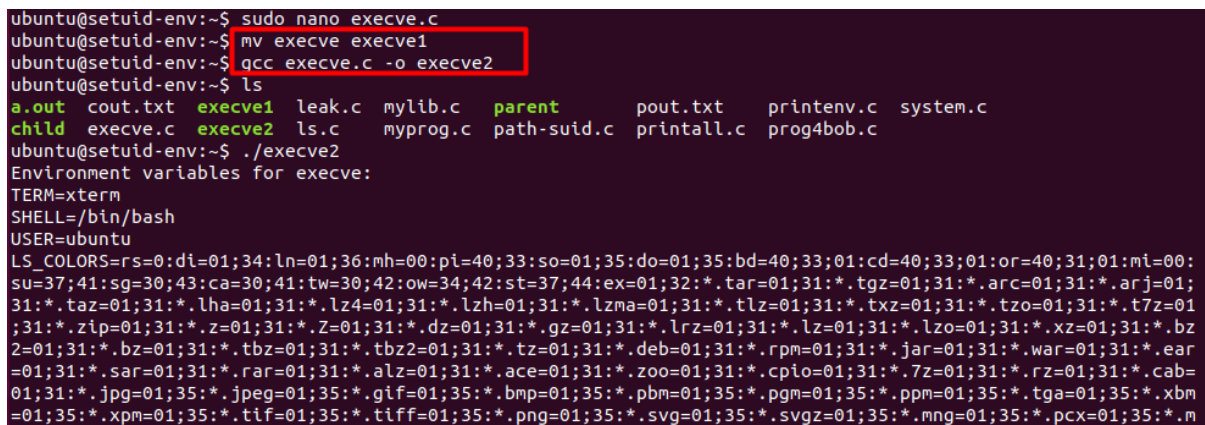
```
ubuntu@setuid-env:~$ gcc execve.c -o execve
ubuntu@setuid-env:~$ nano execve.c
ubuntu@setuid-env:~$ nano execve.c
```

Bước 2. Thay đổi đoạn mã execve("/usr/bin/env", argv, environ);



```
Activities Terminal Sat 02:23
ubuntu@setuid-env: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: execve.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
extern char ** environ;
int main()
{
    char * argv[2];
    printf("Environment variables for execve:\n");
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, environ);
    return 0 ;
}
```

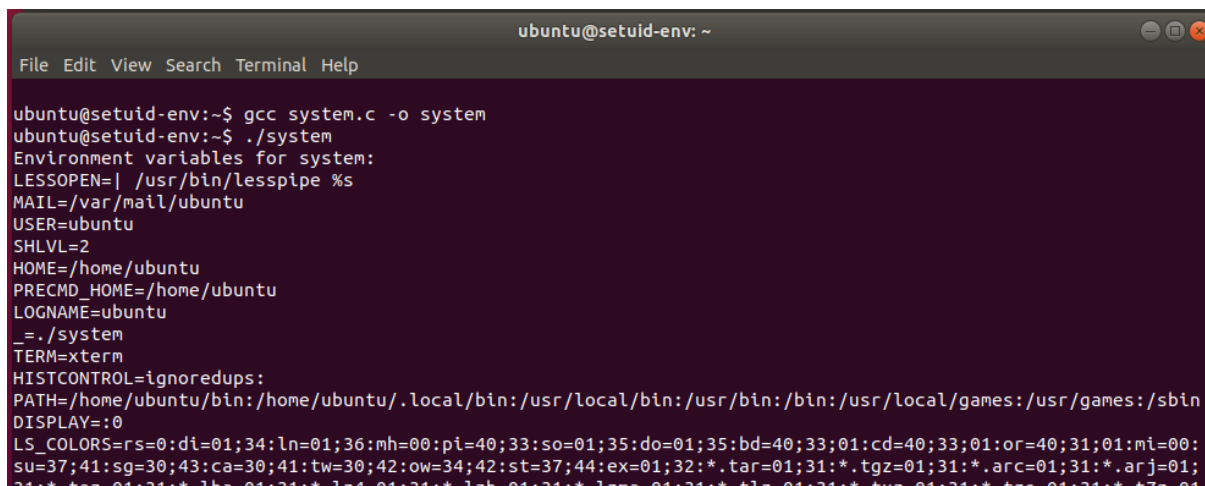
Bước 3. Cuối cùng rút ra kết luận về cách mà chương trình mới lấy các biến môi trường của nó.



```
ubuntu@setuid-env:~$ sudo nano execve.c
ubuntu@setuid-env:~$ mv execve execve1
ubuntu@setuid-env:~$ gcc execve.c -o execve2
ubuntu@setuid-env:~$ ls
a.out  cout.txt  execve1  leak.c  mylib.c  parent  pout.txt  printenv.c  system.c
child  execve.c  execve2  ls.c    myprog.c  path-suid.c  printall.c  prog4bob.c
ubuntu@setuid-env:~$ ./execve2
Environment variables for execve:
TERM=xterm
SHELL=/bin/bash
USER=ubuntu
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:
su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;
31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01
;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz
2=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:*.jpg=01;35:*.jpeg=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:*.m
```

Nhiệm vụ 4: các biến môi trường và hàm system()

Thực hiện biên dịch và chạy chương trình system.c



```
ubuntu@setuid-env: ~
File Edit View Search Terminal Help
ubuntu@setuid-env:~$ gcc system.c -o system
ubuntu@setuid-env:~$ ./system
Environment variables for system:
LESSOPEN=| /usr/bin/lesspipe %s
MAIL=/var/mail/ubuntu
USER=ubuntu
SHLVL=2
HOME=/home/ubuntu
PRECMD_HOME=/home/ubuntu
LOGNAME=ubuntu
_=./system
TERM=xterm
HISTCONTROL=ignoredups:
PATH=/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/sbin
DISPLAY=:0
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
```

```
ubuntu@setuid-env: ~  
File Edit View Search Terminal Help  
GNU nano 2.5.3 File: system.c  
#include <stdio.h>  
#include <stdlib.h>  
int main()  
{  
    printf("Environment variables for system:\n");  
    system("/usr/bin/env");  
    return 0 ;  
}  
  
ubuntu@setuid-env:~$ ls -l /bin/sh  
lrwxrwxrwx 1 root root 4 Feb 17 2016 /bin/sh -> dash  
ubuntu@setuid-env:~$ sudo ln -sf /bin/bash /bin/sh  
ubuntu@setuid-env:~$ ls -l /bin/sh  
lrwxrwxrwx 1 root root 9 Oct 23 13:19 /bin/sh -> /bin/bash  
ubuntu@setuid-env:~$
```

Khi hàm hệ thống thực thi, nó không trực tiếp thực thi mà thay vào đó nó gọi shell và shell thực thi lệnh. Shell gọi nội bộ hệ điều hành và các biến môi trường của quá trình gọi được thông qua tới shell và shell sẽ chuyển nó tới

Nhiệm vụ 5: các biến môi trường và các chương trình Set-UID

Bước 1. Sinh viên cần sử dụng chương trình printall.c

```
ubuntu@setuid-env: ~  
File Edit View Search Terminal Help  
ubuntu@setuid-env:~$ gcc printall.c -o printall  
ubuntu@setuid-env:~$ ls  
a.out  cout.txt  execve1  leak.c  mylib.c  parent  pout.txt  printall.c  prog4bob.c  system.c  
child  execve.c  execve2  ls.c    myprog.c  path-suid.c  printall  printenv.c  system  
ubuntu@setuid-env:~$ ./printall  
Environment variables from printall.  
TERM=xterm  
SHELL=/bin/bash  
USER=ubuntu  
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;  
  
ubuntu@setuid-env: ~  
File Edit View Search Terminal Help  
GNU nano 2.5.3 File: printall.c  
#include <stdio.h>  
#include <stdlib.h>  
extern char ** environ;  
void main()  
{  
    printf("Environment variables from printall.\n");  
    int i = 0;  
    while (environ[i] != NULL) {  
        printf("%s\n", environ[i]);  
        i++;  
    }  
}
```

Bước 2. Thực hiện biên dịch chương trình printall.c và thay đổi chủ sở hữu của chương trình sang root và đặt Set-UID cho nó.

sudo chown root:root a.out

sudo chmod a+s a.out


```
ubuntu@setuid-env:~$ sudo chown root:root printall
ubuntu@setuid-env:~$ sudo chmod a+s printall
ubuntu@setuid-env:~$ ls -l printall
-rwsrwsr-x 1 root root 8680 Oct 23 13:20 printall
ubuntu@setuid-env:~$
```

Trong Bash shell, sinh viên có thể sử dụng lệnh export để đặt các biến môi trường sau (chúng có thể đã tồn tại):

PATH

LD_LIBRARY_PATH

ANY NAME

```
ubuntu@setuid-env:~$ nano printall.c
ubuntu@setuid-env:~$ gcc -o setuid printall.c
ubuntu@setuid-env:~$ sudo chown root setuid
ubuntu@setuid-env:~$ sudo chmod 4755 setuid
ubuntu@setuid-env:~$ export PATH=$PATH:task5
ubuntu@setuid-env:~$ export LD_LIBRARY_PATH=cannotedit
ubuntu@setuid-env:~$ export myenv=test
```

Hãy kiểm tra xem tất cả các biến môi trường sinh viên đặt trong tiến trình cha có được truyền đến tiến trình con của Set-UID hay không.

```
ubuntu@setuid-env:~$ ./setuid grep | PATH
/usr/sbin/exec_wrap.sh: line 16: PATH: command not found
ubuntu@setuid-env:~$ ./setuid | grep PATH
LD_LIBRARY_PATH=cannotedit
PATH=/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/sbin:task5
ubuntu@setuid-env:~$ ./setuid | grep myenv
myenv=test
ubuntu@setuid-env:~$ ./setuid | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=cannotedit
```

Nhiệm vụ 6: hàm system() và các chương trình Set-UID

Chương trình setuid path-suid.c được liệt kê dưới đây được thiết kế để thực thi lệnh /bin/ls

```
ubuntu@setuid-env: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: path-suid.c

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main()
{
    uid_t euid = geteuid();
    printf("euid is %d\n", euid);
    system("ls");
    return 0;
}

ubuntu@setuid-env:~$ printenv MYK
ubuntu@setuid-env:~$ printenv MYK=my variable
ubuntu@setuid-env:~$ export MYK=my variable
ubuntu@setuid-env:~$ printenv MYK
my
ubuntu@setuid-env:~$ export MYK=myvariable
ubuntu@setuid-env:~$ printenv MYK
myvariable
ubuntu@setuid-env:~$
```

Thay đổi đường dẫn export PATH='.':\$PATH

```
ubuntu@setuid-env:~$ ls
a.out  cout.txt  execeve1  leak.c  mylib.c  parent  pout.txt  printall.c  prog4bob.c  system.c
child  execeve.c  execeve2  ls.c    mvprog.c  path-suid.c  printall  printenv.c  system
ubuntu@setuid-env:~$ export PATH='.':$PATH
ubuntu@setuid-env:~$ echo $PATH
.: /home/ubuntu/bin: /home/ubuntu/.local/bin: /usr/local/bin: /usr/bin: /bin: /usr/local/games: /usr/games: /sbin
ubuntu@setuid-env:~$
```

```
ubuntu@setuid-env: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: ls.c
/*
ls program
*/
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(){
    uid_t euid = geteuid();
    printf("my ls prog, euid is %d\n", euid);
    return 0;
}
```

Chạy và biên dịch ls.c ta thấy được euid là 1000

```
ubuntu@setuid-env:~$ gcc ls.c -o ls
ubuntu@setuid-env:~$ ls
my ls prog, euid is 1000
ubuntu@setuid-env:~$
```

thay đổi chủ sở hữu của chương trình sang root và đặt Set-UID cho nó.

Sau đó chạy lại file ls và thấy euid là 0

```
ubuntu@setuid-env:~$ sudo chown root:root ls
ubuntu@setuid-env:~$ sudo chmod 4755 ls
ubuntu@setuid-env:~$ ls -l ls
my ls prog, euid is 0
ubuntu@setuid-env:~$
```

```
ubuntu@setuid-env:~$ cp ls ls1
ubuntu@setuid-env:~$ sudo chmod 4755 ls1
ubuntu@setuid-env:~$ sudo chown root:root ls1
ubuntu@setuid-env:~$ ls -l ls1
my ls prog, euid is 0
ubuntu@setuid-env:~$ ls -l
my ls prog, euid is 0
ubuntu@setuid-env:~$ ls -ls
my ls prog, euid is 0
ubuntu@setuid-env:~$ ./ls
my ls prog, euid is 0
ubuntu@setuid-env:~$
```

Chúng ta đã thay đổi quyền sở hữu thành root, quan sát biến môi trường đường dẫn tìm kiếm lệnh ls trong thư mục hiện tại trước tiên vì nó được chỉ định. Khi phát hiện thấy ls tồn tại, nó sẽ chạy chương trình đó thay vì lệnh shell ls, điều này chứng minh cho chúng ta thấy chương trình set uid có thể chạy các tệp độc hại có quyền root nếu biến đường dẫn bị thay đổi.

Nhiệm vụ 7: biến môi trường LD_PRELOAD và các chương trình Set-UID

Bước 1. Đầu tiên, chúng ta sẽ xem xét cách các biến môi trường này ảnh hưởng đến hành vi của trình tải/liên kết động khi chạy một chương trình bình thường. Thực hiện theo các bước sau:

Đầu tiên, xây dựng một thư viện liên kết động. Đoạn mã chương trình sau đây được lưu trong tệp mylib.c.

```
ubuntu@setuid-env: ~  
File Edit View Search Terminal Help  
GNU nano 2.5.3 File: mylib.c  
#include <stdio.h>  
void sleep (int s)  
{  
    /* If this is invoked by a privileged program, you can do damage here! */  
    printf("I am not sleeping!\n");  
}
```

Chúng ta có thể biên dịch chương trình mylib.c bằng các lệnh sau

```
% gcc -fPIC -g -c mylib.c
```

```
% gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
```

Tiếp theo cần thiết lập biến môi trường LD_PRELOAD

```
% export LD_PRELOAD=./libmylib.so.1.0.1
```

```
ubuntu@setuid-env: ~  
File Edit View Search Terminal Help  
ubuntu@setuid-env:~$ gcc -fPIC -g -c mylib.c  
ubuntu@setuid-env:~$ gcc -shared -o libmylib.so.1 mylib.o -lc  
ubuntu@setuid-env:~$ export LD_PRELOAD=./libmylib.so.1  
ubuntu@setuid-env:~$ echo $LD_PRELOAD  
./libmylib.so.1  
ubuntu@setuid-env:~$  
  
ubuntu@setuid-env:~$ gcc myprog.c -o myprog  
ubuntu@setuid-env:~$ sudo ls  
a.out      execve1      ls          mylib.o     path-suid.c  printenv.c  
child      execve2      ls.c        myprog      pout.txt     prog4bob.c  
cout.txt   leak.c       ls1         myprog.c    printall     system  
execve.c   libmylib.so.1 mylib.c     parent      printall.c   system.c  
ubuntu@setuid-env:~$ ./myprog  
I am not sleeping!
```

Cuối cùng, biên dịch chương trình myprog.c trong cùng thư mục với thư viện liên kết động libmylib.so.1.0.1 như sau:

```
ubuntu@setuid-env:~$ gcc -o myprog myprog.c  
ubuntu@setuid-env:~$
```

Bước 2. chạy myprog dưới các điều kiện sau đây, và quan sát điều gì xảy ra.

Biến myprog thành một chương trình bình thường và chạy nó dưới dạng một người dùng bình thường.

Biến myprog thành một chương trình Set-UID root và chạy nó dưới dạng một người dùng bình thường.

Trở thành người dùng root với lệnh sudo su, xuất biến môi trường LD_PRELOAD và chạy lại chương trình myprog.

Biến myprog thành một chương trình Set-UID user1 (tức là chủ sở hữu là user1, là một tài khoản người dùng khác), xuất biến môi trường LD_PRELOAD lại dưới tên người dùng user1 và chạy nó.

```
ubuntu@setuid-env:~$ sudo chown root myprog
ubuntu@setuid-env:~$ sudo chmod 4755 myprog

ubuntu@setuid-env:~$ sudo su
root@setuid-env:/home/ubuntu# echo $LD_PRELOAD

root@setuid-env:/home/ubuntu# export LD_PRELOAD=./libmylib.so.1
root@setuid-env:/home/ubuntu# echo $LD_PRELOAD
./libmylib.so.1
root@setuid-env:/home/ubuntu# id
uid=0(root) gid=0(root) groups=0(root)
root@setuid-env:/home/ubuntu# ./myprog
I am not sleeping!
```

Nhiệm vụ 8: Khả năng rò rỉ

```
ubuntu@setuid-env:~$ sudo nano /etc/zzz
ubuntu@setuid-env:~$ sudo ls -l /etc/zzz
-rw-rw-r-- 1 root root 29 Oct 23 13:48 /etc/zzz
ubuntu@setuid-env:~$ cat /etc/zzz
important stuff, not really.
```

Chạy và biên dịch file leak.c, sau đó thay đổi chủ sở hữu của nó thành root và biến nó thành một chương trình Set-UID.

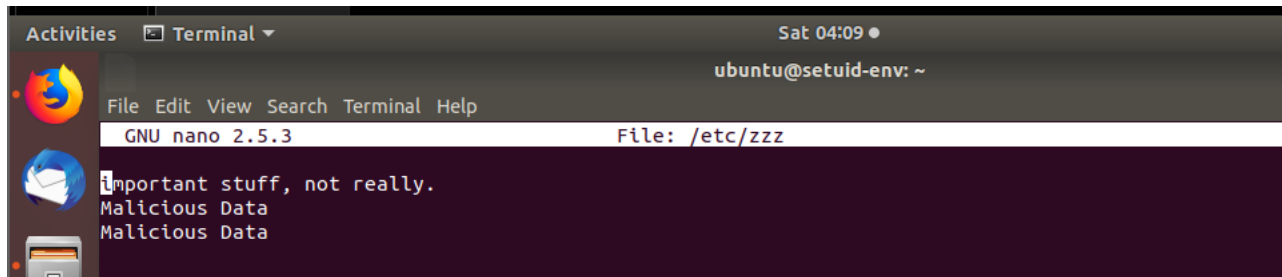
```
ubuntu@setuid-env:~$ gcc -o leak leak.c
ubuntu@setuid-env:~$ sudo chown root leak
ubuntu@setuid-env:~$ sudo chmod 4755 leak
```

```
ubuntu@setuid-env:~$ sudo chmod 0644 /etc/zzz
ubuntu@setuid-env:~$ sudo ls -l /etc/zz
ls: cannot access '/etc/zz': No such file or directory
ubuntu@setuid-env:~$ sudo ls -l /etc/zzz
-rw-r--r-- 1 root root 29 Oct 23 13:48 /etc/zzz
ubuntu@setuid-env:~$ gcc leak.c -o leak
ubuntu@setuid-env:~$ ./leak
Cannot open /etc/zzz
ubuntu@setuid-env:~$ sudo chown root:root leak
ubuntu@setuid-env:~$ sudo chmod 0755 leak
ubuntu@setuid-env:~$ sudo ls -l leak
-rwxr-xr-x 1 root root 9008 Oct 23 13:52 leak
```

```
ubuntu@setuid-env:~$ stat leak
  File: 'leak'
  Size: 9008          Blocks: 24          IO Block: 4096   regular file
Device: 31h/49d Inode: 2115397      Links: 1
Access: (4755/-rwsr-xr-x)  Uid: (  0/   root)   Gid: (1000/  ubuntu)
Access: 2023-09-30 10:32:20.551503434 +0000
Modify: 2023-09-30 10:30:13.308589513 +0000
Change: 2023-09-30 10:31:23.676497387 +0000
```

Ta mở tệp `zzz` với quyền root trong `/etc`, nó sẽ hiện ra dòng chữ “important stuff, not really.

```
ubuntu@setuid-env:~$ gcc leak.c -o leak
ubuntu@setuid-env:~$ ls
my ls prog, euid is 0
ubuntu@setuid-env:~$ ./leak
Cannot open /etc/zzz
ubuntu@setuid-env:~$ sudo chown root:root leak
ubuntu@setuid-env:~$ sudo chmod 4755 leak
ubuntu@setuid-env:~$ sudo ls -l leak
-rwsr-xr-x 1 root root 9008 Oct 23 14:04 leak
ubuntu@setuid-env:~$ ./leak
ubuntu@setuid-env:~$ sudo chmod +s leak
ubuntu@setuid-env:~$ sudo ls -l leak
-rwsr-sr-x 1 root root 9008 Oct 23 14:04 leak
ubuntu@setuid-env:~$
```



- Khi chạy chương trình dưới dạng người dùng bình thường, nó sẽ thực hiện các thao tác sau:
 - Mở tệp `/etc/zzz` để ghi dữ liệu vào nó (nếu tệp tồn tại).
 - Tiến hành một số tác vụ giả lập (sử dụng `sleep(1)` để giữ tiến trình chạy trong 1 giây).
 - Sau khi hoàn thành các tác vụ, tiến trình sử dụng `setuid(getuid())` để thu hồi đặc quyền root và chuyển về đặc quyền của người dùng gốc (real uid).
 - Tiến trình tạo một tiến trình con bằng cách sử dụng `fork()`. Tiến trình cha sẽ tiếp tục chạy và thoát ra.
 - Tiến trình con, trong ví dụ này, được giả định đã bị tấn công và thực hiện các thao tác độc hại. Nó ghi dữ liệu "Malicious Data" vào tệp `/etc/zzz` và sau đó đóng tệp lại.
- Kết quả: Tiến trình con có thể ghi dữ liệu độc hại vào tệp `/etc/zzz` sau khi đặc quyền root đã bị thu hồi bởi tiến trình cha. Vì tiến trình con thực hiện các tác vụ với đặc quyền gốc của người dùng và không còn đặc quyền root, việc ghi dữ liệu vào tệp `/etc/zzz` vẫn được thực hiện mà không cần đặc quyền root.
- Lý do: Đặc quyền root đã bị thu hồi bởi tiến trình cha bằng cách sử dụng `setuid(getuid())`, nhưng tiến trình con vẫn có quyền truy cập vào tệp `/etc/zzz` bởi vì nó không thể thu hồi các quyền đối với các tài nguyên đã được mở hoặc sở hữu trước đó. Điều này có thể gây ra một lỗ hổng bảo mật, vì tiến trình con có thể tiếp tục tương tác với tài nguyên mà nó không nên được truy cập

II. Checkwork

```

student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/setuid-env
Labname setuid-env

Student | cmp_printenv | did_diff | ran_execve | ran_mylib | ran_printall | ran_ls | ran_leak |
=====|=====|=====|=====|=====|=====|=====|=====|
B20DCAT094 | 2 | 3 | 3 | 2 | Y | Y | Y |

What is automatically assessed for this lab:
ran_printall: Ran the printall program
ran_ls: Got setuid program to run bogus ls command
ran_leak: Ran the leak program
cmp_printenv: Count of compilations of printenv.c program
did_diff: Count of use of diff command
ran_execve: Count of use running the execve program
ran_mylib: Count of use running the myprog LD program

```