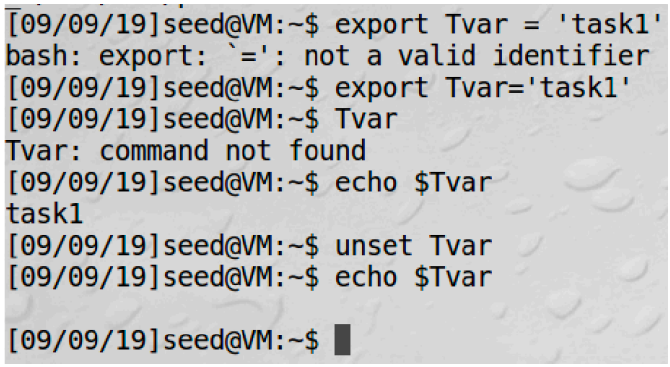
2019/09/09 Marco Lin

Task 1: Manipulating environment variables

Use Printenv or command to print out the environment variable:

Use export and unset to set or unset environment variables:



Task 2: Inheriting environment variables from parents

Step2:

Explanation:

Child output sceenshot: the variables are the same as the parent.

```
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
TERMINATOR_UUID=urn:uuid:b101ba4c-7f82-4b8d-a37d-1ee142d1
b642
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=2218
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.
so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.6
4.0:/home/seed/lib/boost/libboost_system.so.1.64.0
WINDOWID=62914564
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session
/1000/1379
"child" [readonly] 75L, 4223C          1,1           Top
```

Use diff command:

```
[09/10/19]seed@VM:~/Documents$ diff child parent
5c5
< TERMINATOR_UUID=urn:uuid:b101ba4c-7f82-4b8d-a37d-1ee142
d1b642
---
> TERMINATOR_UUID=urn:uuid:78fe7c45-c5ee-4dfb-bd9e-60cbe3
830b9e
9d8
< GIO_LAUNCHED_DESKTOP_FILE_PID=2218
17c16
< WINDOWID=62914564
---
> WINDOWID=69206020
29d27
< GIO_LAUNCHED_DESKTOP_FILE=/usr/share/applications/termi
nator.desktop
[09/10/19]seed@VM:~/Documents$
```

Step3:

Explanation :

The output from child and parent file are totally same. This is because when we use fork() function, it will create a new space for child process. All environment variables in child will come from parents and they will be the same as well.

Task 3: Environment variables and execve()

When using execve ("/usr/bin/env", argv, NULL):

```
[09/10/19]seed@VM:~/Documents$ ./task3
[09/10/19]seed@VM:~/Documents$
```

Explanation

When the parameter is NULL, the output will be blank. When the parameter is environ, the output will print the environment variables:

```
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
TERMINATOR_UUID=urn:uuid:c281f222-9203-418a-8d62-d2ef828b
2330
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=2257
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.
so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.6
4.0:/home/seed/lib/boost/libboost_system.so.1.64.0
WINDOWID=29360132
OLDPWD=/home/seed
```

Task 4: Environment variables and system()

```
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu
ORBIT_SOCKETDIR=/tmp/orbit-seed
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:
/home/seed/source/boost_1_64_0/stage/lib:
SHLVL=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
OLDPWD=/home/seed
HOME=/home/seed
QT4_IM_MODULE=xim
DESKTOP_SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE=/usr/share/applications/termina
tor.desktop
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
```

Task 5: Environment variable and Set-UID Programs

```
[09/10/19]seed@VM:~/Documents$ ls
task5  task5.c
[09/10/19]seed@VM:~/Documents$ ls -l task5
-rwxrwxr-x 1 seed seed 7396 Sep 10 23:13 task5
[09/10/19]seed@VM:~/Documents$ sudo chown root task5
[09/10/19]seed@VM:~/Documents$ ls -l task5
-rwxrwxr-x 1 root seed 7396 Sep 10 23:13 task5
[09/10/19]seed@VM:~/Documents$ sudo chmod 4755 task5
[09/10/19]seed@VM:~/Documents$ ls
task5  task5.c
[09/10/19]seed@VM:~/Documents$ ls -l task5
-rwsr-xr-x 1 root seed 7396 Sep 10 23:13 task5
[09/10/19]seed@VM:~/Documents$
```

```
PATH=task5
QT_IM_MODULE=ibus
```

```
Marco=task5
TERM=xterm-256color
VTE_VERSION=4205
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=62914570
OLDPWD=/home/seed
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1247
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
@
@
@
@
                                                              6,1           Top
```

Explanation

In output, we only saw the variables PATH and Marco. There is no LD LIBRARY PATH variable in output because it is protected by root.

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

```
root@VM:/home/seed/Documents# chmod 4755 task6
root@VM:/home/seed/Documents# ls
task5  task5.c  task6  task6.c
root@VM:/home/seed/Documents# ls -l
total 24
-rwsr-xr-x 1 root seed 7396 Sep 10 23:13 task5
-rw-rw-r-- 1 seed seed  166 Sep 10 23:13 task5.c
-rwsr-xr-x 1 root seed 7348 Sep 10 23:33 task6
-rw-rw-r-- 1 seed seed   39 Sep 10 23:40 task6.c
root@VM:/home/seed/Documents# exit
exit
[09/10/19]seed@VM:~/Documents$ cp /bin/sh /tmp/ls
[09/10/19]seed@VM:~/Documents$ ./task6
task5  task5.c  task6  task6.c
[09/10/19]seed@VM:~/Documents$ ls
task5  task5.c  task6  task6.c
[09/10/19]seed@VM:~/Documents$ PATH=~:$PATH
[09/10/19]seed@VM:~/Documents$ vim task6.c
[09/10/19]seed@VM:~/Documents$ ./task6
task5  task5.c  task6  task6.c
```

```
[09/10/19]seed@VM:~/Documents$ cp /bin/sh ~/ls
[09/10/19]seed@VM:~/Documents$
[09/10/19]seed@VM:~/Documents$ ./task6
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip
),46(plugdev),113(lpadmin),128(sambashare)
$
```

Explanation

It can have root previlege, copy /bin/sh to /document with new name ls. Then set PATH to current directory, compile and run system program and we will get root previlege.

Task 7: The LD PRELOAD environment variable and Set-UID Programs

1. Make myprog a regular program, and run it as a normal user.

```
[09/11/19]seed@VM:~/Documents$ gcc -fPIC -g -c mylib.c
[09/11/19]seed@VM:~/Documents$ ls
mylib.c  mylib.o
[09/11/19]seed@VM:~/Documents$ gcc -shared -o libmylib.so.1.0.1 mylib.o
[09/11/19]seed@VM:~/Documents$ ls
libmylib.so.1.0.1  mylib.c  mylib.o
[09/11/19]seed@VM:~/Documents$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/11/19]seed@VM:~/Documents$ vim myprog.c
[09/11/19]seed@VM:~/Documents$ gcc myprog.c -o myprog
myprog.c: In function ‘main’:
myprog.c:2:1: warning: implicit declaration of function ‘sleep’ [-Wimpli
tion-declaration]
 sleep(1);
 ^
[09/11/19]seed@VM:~/Documents$ echo LD_PRELOAD
LD_PRELOAD
[09/11/19]seed@VM:~/Documents$ echo $LD_PRELOAD
./libmylib.so.1.0.1
[09/11/19]seed@VM:~/Documents$ ./myprog
i am not sleeping
```

2. Make myprog a Set-UID root program, and run it as a normal user.

```
-rwxrwxr-x 1 seed seed 7924 Sep 11 00:02 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed   79 Sep 11 00:00 mylib.c
-rw-rw-r-- 1 seed seed 2584 Sep 11 00:02 mylib.o
-rwsr-xr-x 1 root root 7348 Sep 11 00:14 myprog
-rw-rw-r-- 1 seed seed   37 Sep 11 00:14 myprog.c
root@VM:/home/seed/Documents# exit
exit
[09/11/19]seed@VM:~/Documents$ ls
libmylib.so.1.0.1  mylib.c  mylib.o  myprog  myprog.c
[09/11/19]seed@VM:~/Documents$ ./myprog
[09/11/19]seed@VM:~/Documents$
```

Explanation

In this situation, it will ignore LD_PRELOAD environment variable and use the system's default sleep() function. So sleep() function will not be overrided.

3. Make myprog a Set-UID root program, export the LD PRELOAD environment variable again in the root account and run it.

```
[09/11/19]seed@VM:~/Documents$ sudo su
root@VM:/home/seed/Documents# ./myprog
root@VM:/home/seed/Documents# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Documents# ./myprog
i am not sleeping
root@VM:/home/seed/Documents#
```

Explanation

In this situation, it will use LD_PRELOAD environment variable and override sleep() function.

4. Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the LD PRELOAD environment variable again in a different user’s

account (not-root user) and run it.

```
root@VM:/home/seed/Documents# useradd -d /usr/user1 -m user1
root@VM:/home/seed/Documents# su user1
user1@VM:/home/seed/Documents$ export LD_PRELOAD=./libmylib.so.1.0.1
user1@VM:/home/seed/Documents$ ls
libmylib.so.1.0.1  mylib.c  mylib.o  myprog  myprog.c
user1@VM:/home/seed/Documents$ gcc -o myprog myprog.c
myprog.c: In function ‘main’:
myprog.c:3:1: warning: implicit declaration of function ‘sleep’ [-Wimplicit-func
tion-declaration]
 sleep(1);
 ^
/usr/bin/ld: cannot open output file myprog: Permission denied
collect2: error: ld returned 1 exit status
user1@VM:/home/seed/Documents$ chmod u+s myprog
chmod: changing permissions of 'myprog': Operation not permitted
user1@VM:/home/seed/Documents$ ls -l myprog
-rwsr-xr-x 1 root root 7348 Sep 11 00:14 myprog
user1@VM:/home/seed/Documents$ ./myprog
user1@VM:/home/seed/Documents$
```

Explanation

In this situation, it will not override sleep() function.

From the four formal situation, we know that only a user run the program created by himself, LD_PRELOAD environment valiable can be used and sleep() function can be overrided.

Task 8: Invoking external programs using system() versus execve()

```
[09/11/19]seed@VM:~/Documents$ sudo chown root bob
[09/11/19]seed@VM:~/Documents$ sudo chmod 4755 bob
[09/11/19]seed@VM:~/Documents$ ./bob
Please type a file name.
[09/11/19]seed@VM:~/Documents$ ls
bob  bob.c  libmylib.so.1.0.1  mylib.c  mylib.o  myprog  myprog.c
[09/11/19]seed@VM:~/Documents$ ./bob
Please type a file name.
[09/11/19]seed@VM:~/Documents$ bob "/etc/hostname;/bin/sh"
VM
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip
),46(plugdev),113(lpadmin),128(sambashare)
$ rm /etc/xxxx
rm: cannot remove '/etc/xxxx': No such file or directory
$
```

Explanation

Q.If you were Bob, can you compromise the integrity of the system? For example, can you remove a file that is not writable to you?

The file is not safe, Bob can read, write or move files which only root user can run. We can use this program to get root. However, if we comment system() and uncomment execve, we will fail:

```
[09/11/19]seed@VM:~/Documents$ sudo chown root bob
[09/11/19]seed@VM:~/Documents$ sudo chmod 4755 bob
[09/11/19]seed@VM:~/Documents$ ./bob
Please type a file name.
[09/11/19]seed@VM:~/Documents$ bob
Please type a file name.
[09/11/19]seed@VM:~/Documents$ bob "/etc/hostname;/bin/sh"
/bin/cat: '/etc/hostname;/bin/sh': No such file or directory
[09/11/19]seed@VM:~/Documents$
```

Explanation

Q. Do your attacks in Step 1 still work? Please describe and explain your observations.
When modify q to 1, the attack can't make sense. The reason why the before attack effectively is because system() call /bin/sh, which links /sh. After running cat file with root privilege, it runs mv file file_new. But when q = 1, execve() will regard file;mv file file_new2 as a folder name, so system will prompt there have no the file.

Task 9: Capability Leaking

```
[09/11/19]seed@VM:~/Documents$ sudo chown root task9
[09/11/19]seed@VM:~/Documents$ sudo chmod 4755 task9
[09/11/19]seed@VM:~/Documents$ touch /etc/zzz
touch: cannot touch '/etc/zzz': Permission denied
[09/11/19]seed@VM:~/Documents$ sudo touch /etc/zzz
[09/11/19]seed@VM:~/Documents$ ./task9
[09/11/19]seed@VM:~/Documents$ cat /etc/zzz
Malicious Data
[09/11/19]seed@VM:~/Documents$
```

Explanation

Will the file /etc/zzz be modified?
Yes, this is because file zzz is opened before set uid.

3.1 Emma runs a Set-UID program that is owned by Dan. The program tries to read from /tmp/x, which is readable to Emma, but not to anybody else. Can this program successfully read from the file?

No. The Set-UID program owned by Dan will try to read /tmp/x as Dan, and not Emma. And since only Emma can access that file, the program cannot.

3.2 In our code, when we use execve() to execute an external program xyz , we pass NULL in the third argument. How many environment variables will the process running xyz has?
Zero, this is because when we set the third argument to be NULL, the function will

return nothing on success.

3.3 We are trying to turn a program prog owned by the seed user into a Set-UID program that is owned by root. Can running the following commands achieve the goal?

No. Upon chown, the Set-UID bit is reset. So running chmod 4755 should always be the last step in the set up.

3.4 A program abc invokes an external program xyz using system(), which is affected by the PATH environment variable. When we invoke abc from a shell prompt, how does the shell variable PATH in the current shell end up affecting the behavior of the system() function?
With system(command) in shell, if we make the program be a set_UID program, it will be execute with root. And it can do whatever user want to do.

3.5 When a program takes an input from users, we can redirect the input device, so the program can take the input from a file. For example, we can use prog < myfile to provide the data from myfile as input to the prog program. Now, if prog is a root-owned Set-UID program, can we use the following method to to get this privileged program to read from the /etc/shadow file?

No. The fact that prog is a Set-UID program with root privileges has nothing to do with the file that's input after. Since a normal user can't access /etc/shadow, user can't use it to redirect the input device. Trying to do see will result in a permission denied error.

3.6 Assume that you have a file that you would allow other users to read, only if a user’s ID is smaller than 1000. Please describe how you can actually achieve this.

Assuming you are root, create a new group with all users that have an id less than 1000.
$ cat /etc/passwd | cut -d : -f1,3 and, it will list all current users and their userid.