东南大學 SOUTHEAST UNIVERSITY | 网络空间安全学院 School of Cyber Science and Engineering

网络空间安全综合课程设计

实验报告（一）

Environment Variable and Set-UID Program

学号：57117137 姓名：刘康亮

东南大学网络空间安全学院

2020 年 9 月 3 日

Task 1: Manipulating Environment Variables

1、使用 env 打印出环境变量，得到如下结果：

```
[09/03/20]seed@VM:~$ env
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:210df9a5-9519-4dfb-9189-f407d3681b32
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
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/see
d/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_syst
em.so.1.64.0
WINDOWID=27262980
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1533
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
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:*.Z=01;31:*.dz=01;31:*.gz=01;3
1:*.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;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:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.
```

2、PWD：

```
[09/03/20]seed@VM:~$ env | grep PWD
PWD=/home/seed
```

3、使用 export 和 unset 设置或删除环境变量

```
[09/03/20]seed@VM:~$ env | grep PWD
PWD=/home/seed
[09/03/20]seed@VM:~$ export lkl=1203
[09/03/20]seed@VM:~$ env | grep lkl
lkl=1203
[09/03/20]seed@VM:~$ unset lkl
[09/03/20]seed@VM:~$ env | grep lkl
[09/03/20]seed@VM:~$
```

Task 2: Passing Environment Variables from Parent Process to Child Process

1、将所给的程序编译并执行，生成 a.out 文件，查看其内容，为各个环境变量的值

```
[09/03/20]seed@VM:~$ cd Desktop
[09/03/20]seed@VM:~/Desktop$ gcc test.c
[09/03/20]seed@VM:~/Desktop$ a.out
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:210df9a5-9519-4dfb-9189-f407d3681b32
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
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/see
d/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_syst
em.so.1.64.0
WINDOWID=27262980
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1533
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
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
```

2、将程序中 child process 下的 printenv()注释，将 parent process 下的 printenv()取消注释

```
void main()
{
  pid_t childPid;

  switch(childPid = fork()) {
    case 0: /* child process */
        //printenv();
        exit(0);
    default: /* parent process */
        printenv();
        exit(0);
  }
}
```

重新编译并执行，结果保存为 b.out 文件，查看其内容，与 a.out 很相似

3、分别将 a.out、b.out 文件的结果分别保存为 child 和 parent，再使

用 diff 命令比较，两者的内容是完全相同的。

```
seed@VM:~/Desktop$ a.out > child
seed@VM:~/Desktop$ b.out > parent
seed@VM:~/Desktop$ diff a.out b.out
es a.out and b.out differ
```

这说明子进程环境变量会继承父环境变量。

Task 3: Environment Variables and execve()

1、将所给程序编译并运行，该程序调用了/usr/bin/env，可以打印出当前进程的环境变量。运行后，发现执行结果为空。

```
[09/03/20]seed@VM:~/Desktop$ gcc -o c.out 1-3.c
1-3.c: In function ‘main’:
1-3.c:9:1: warning: implicit declaration of function ‘execve’ [-Wimplicit-fu
nction-declaration]
 execve("/usr/bin/env", argv, NULL);
```

2、把 execve()的调用改为以下内容

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

因为 execve 第三个参数为传递给执行文件的新环境变量数，第一步我们赋予新进程的环境变量为空，所以印出环境变量结果为空，现在修改参数后，可以看到结果不为空：

```
[09/03/20]seed@VM:~/Desktop$ gcc -o d.out 1-3.c
1-3.c: In function ‘main’:
1-3.c:9:1: warning: implicit declaration of function ‘execve’ [-Wimplicit-fu
nction-declaration]
 execve("/usr/bin/env", argv, environ);
 ^
[09/03/20]seed@VM:~/Desktop$ ./d.out
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:210df9a5-9519-4dfb-9189-f407d3681b32
```

Task 4: Environment Variables and system()

编译并运行所给程序，执行结果：

```
[09/03/20]seed@VM:~/Desktop$ gcc -o 4.out 1-4.c
[09/03/20]seed@VM:~/Desktop$ ./4.out
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
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/b
oost_1_64_0/stage/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
```

Task 5: Environment Variable and Set-UID Programs

1、根据所给程序, 该程序可以在当前进程中打印出所有的环境变量。

然后切换为 root 权限，将上述程序的所有权改为 root，并使它成为一个 Set-UID 程序

2、切换为普通用户，使用 export 命令设置环境变量：PATH、LD_LIBRARY_PATH、ANY_NAME

```
[09/03/20]seed@VM:~/Desktop$ su
Password:
root@VM:/home/seed/Desktop# chown root 1-5.c
root@VM:/home/seed/Desktop# chmod 4755 1-5.c
root@VM:/home/seed/Desktop# exit
exit
[09/03/20]seed@VM:~/Desktop$ export PATH="$PATH:/usr/local/"
[09/03/20]seed@VM:~/Desktop$ export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/l
ocal/"
[09/03/20]seed@VM:~/Desktop$ export Liukangliang="/usr/local"
```

可以看到以上三个被定义的环境变量全部被包括在 shell 中

```
[09/03/20]seed@VM:~/Desktop$ echo $Liukangliang
/usr/local
```

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

1、将所给程序编译，将执行文件设置为 set-UID 文件，使用 ls -l 命令可以查看设置成功

```
[09/03/20]seed@VM:~/Desktop$ gcc -o 6.out 1-6.c
1-6.c: In function ‘main’:
1-6.c:3:1: warning: implicit declaration of function ‘system’ [-Wimplicit-fu
nction-declaration]
 system("ls");
 ^
[09/03/20]seed@VM:~/Desktop$ su
Password:
root@VM:/home/seed/Desktop# chown root 6.out
root@VM:/home/seed/Desktop# chmod 4755 6.out
root@VM:/home/seed/Desktop# ls -l 6.out
-rwsr-xr-x 1 root seed 7344 Sep  3 11:45 6.out
root@VM:/home/seed/Desktop#
```

2、将/bin/sh 复制到程序当前目录，命令为 ls，设置环境变量，

```
[09/03/20]seed@VM:~/Desktop$ cp /bin/sh ls
[09/03/20]seed@VM:~/Desktop$ PATH=~:$PATH
[09/03/20]seed@VM:~/Desktop$ t6
$ printenv PATH
/home/seed:/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/us
r/bin:/sbin:/bin:/usr/games:/usr/local/games:.:/home/seed/android/android-sdk-lin
ux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/a
ndroid-ndk/android-ndk-r8d:/home/seed/.local/bin
$ exit
```

PATH 环境变量的命令找寻顺序是先找寻当前目录，当前目录我们自己编造了一个 ls，所以程序就会直接执行伪造的 ls。sh 原本的作用是创建一个新 shell，在执行此命令后我们就会一直停留在子进程中，直到我们主动退出这个程序，我们才会回到原来的权限。

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

新建一个动态链接库，手册所给程序 mylib.c 内有 sleep 函数。按照所给命令编译 mylib.c，然后将 LD_PRELOAD 设置为我们编译的动态链接库。

在链接库的相同目录下编译 myprog 程序，并在不同的情形下运行

myprog 程序:

（1）普通用户运行程序

```
[09/03/20]seed@VM:~/Desktop$ ./myprog
I am not sleeping!
```

有输出 I am not sleeping! 说明 sleep 函数为链接库的

（2）普通用户运行 set -UID 的 root 程序

```
[09/03/20]seed@VM:~/Desktop$ sudo chown root ./myprog
[09/03/20]seed@VM:~/Desktop$ sudo chmod 4755 ./myprog
[09/03/20]seed@VM:~/Desktop$ ./myprog
[09/03/20]seed@VM:~/Desktop$
```

有 sleep，无输出 说明 sleep 函数为系统库的

（3）root 用户运行设置过 LD_PRELOAD 的 root 程序

```
root@VM:/home/seed/Desktop# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Desktop# ./myprog
I am not sleeping!
```

有输出

（4）用户 1 运行其他用户（我这里用户为 lkl）的程序

```
[09/03/20]seed@VM:~/Desktop$ sudo chown lkl ./myprog
[09/03/20]seed@VM:~/Desktop$ sudo chmod 4755 ./myprog
[09/03/20]seed@VM:~/Desktop$ ./myprog
[09/03/20]seed@VM:~/Desktop$
```

无输出

总结：只有用户自己创建的程序自己去运行，才会使用 LD_PRELOAD 环境变量，否则忽略 LD_PRELOAD 环境变量。

Task 8: Invoking External Programs Using system() versus execve()

将所给程序编译，将其设置为 set-UID 程序

```
[09/03/20]seed@VM:~/Desktop$ sudo chown root t8
[09/03/20]seed@VM:~/Desktop$ sudo chmod 4755 t8
[09/03/20]seed@VM:~/Desktop$ ls -l t8
-rwsr-xr-x 1 root seed 7548 Sep  1 03:39 t8
[09/03/20]seed@VM:~/Desktop$
```

新建一个文件 test8，将其权限设置为仅限 root 用户可读写执行，然后执行上述程序，发现要求 root 权限的 test8 文件已经被删除了，被更名为 mine

注释掉 system(command)语句，并取消 execve（）语句，则该程序将使用 execve（）来调用该命令。然后重新按照上述做法，发现 test8 文件不会被删除，原来的攻击方法已经失效了，因为 execve()函数会把 file; mv file file_new 看成是一个文件名，系统会提示不存在这个文件，system()则不会。

Task 9: Capability Leaking

先创建/etc/zzz 文件，并将其设置权限

```
root@VM:/home/seed/Desktop# chmod u+s /etc/zzz
root@VM:/home/seed/Desktop# ls -l /etc/zzz
-rwSr--r-- 1 root root 0 Sep  3 04:13 /etc/zzz
```

```
[09/03/20]seed@VM:~/Desktop$ t9
[09/03/20]seed@VM:~/Desktop$ cat /etc/zzz
Malicious Data
```

编译并执行之，再查看文件，发现被写入了信息