CLASS 1

Environment Variable and Set-UID Program Lab

吴瑞欣-E41614059

Task 1: Manipulating environment variables

Use printenv or env command to print out the environment variables. If you are interested in some particular environment variables, such as PWD, you can use "printenv PWD" or "env |grep PWD".

```
[09/11/2018 19:37] seed@ubuntu:~$ pwd
/home/seed
[09/11/2018 19:37] seed@ubuntu:~$ printenv
SSH_AGENT_PID=2404
GPG_AGENT_INFO=/tmp/keyring-yjqTL6/gpg:0:1
TERM=xterm
SHELL=/bin/bash
XDG_SESSION_COOKIE=6da3e071019f67095bc4c5e900
```

Use export and unset to set or unset environment variables. It should be noted that these two commands are not seperate programs; they are two of the Bash's internal commands (you will not be able to find them outside of Bash).

export：拓展局部变量为全局变量

```
[09/11/2018 19:48] seed@ubuntu:~$ my_variable="Hello WORLD"
[09/11/2018 19:53] seed@ubuntu:~$ export my_variable
[09/11/2018 19:54] seed@ubuntu:~$ bash
[09/11/2018 19:54] seed@ubuntu:~$ echo $my_variable
Hello WORLD
[09/11/2018 19:55] seed@ubuntu:~$ unset my_variable
[09/11/2018 19:55] seed@ubuntu:~$ echo $my_variable

[09/11/2018 19:55] seed@ubuntu:~$ exit
exit
[09/11/2018 19:56] seed@ubuntu:~$ echo $my_variable
Hello WORLD
[09/11/2018 19:56] seed@ubuntu:~$
```

Task 2: Inheriting environment variables from parents

Man fork 查看、调取了 fork 函数

NAME

fork - create a child process

SYNOPSIS

#include <unistd.h>

pid_t fork(void);

Step 1. Please compile and run the following program, and describe your observation. Because the output contains many strings, you should save the output into a file, such as using a.out > child (assuming that a.out is your executable file name).

解: 在这题中由于代码是复制粘贴/* */注释中'/'与'*'有空格, 导致编译出错。正确代码如下:

```
[09/11/2018 20:58] seed@ubuntu:~/Desktop$ gcc first.c>child
[09/11/2018 20:58] seed@ubuntu:~/Desktop$
```

Step 2. Now comment out the printenv() statement in the child process case, and uncomment the printenv() statement in the parent process case. Compile and run the code, and describe your observation. Save the output in another file.

```
[09/11/2018 21:02] seed@ubuntu:~/Desktop$ a.out
SSH_AGENT_PID=2404
GPG_AGENT_INFO=/tmp/keyring-yjqTL6/gpg:0:1
TERM=xterm
SHELL=/bin/bash
```

Step 3. Compare the difference of these two files using the diff command. Please draw your conclusion.

```
[09/12/2018 00:42] seed@ubuntu:~/Desktop$ gcc first.c
[09/12/2018 00:43] seed@ubuntu:~/Desktop$ a.out>child
[09/12/2018 00:43] seed@ubuntu:~/Desktop$ gcc first.c
[09/12/2018 00:43] seed@ubuntu:~/Desktop$ a.out>parent
[09/12/2018 00:43] seed@ubuntu:~/Desktop$ diff child parent
[09/12/2018 00:44] seed@ubuntu:~/Desktop$
```

C 语言程序分析:

一、pid_t 类似一个类型，就像 int 型一样，int 型定义的变量都是整型的，pid_t 定义的类型都是进程号类型

pid_t 与 int 又有什么区别呢?

pid_t 是 typedef 定义的类型，表示进程的 id

在 sys/types.h 中定义:

typedef short pid_t;

所以说 pid_t 就是一个 short 类型的变量，实际表示的是内核中进程表的索引

二、其中 fork(void)为创建子进程，有趣的是他有三种不同的返回值

1、在父进程中，fork 返回新创建的子进程的 PID

2.、在子进程中，fork 返回 0

3、如果出现错误，fork 返回一个负值

三、Linux C 中 environ 变量是一个 char ** 类型，存储着系统的环境变量。

四、return 与 exit 的区别

用 exit()函数可以退出程序并将控制权返回给操作系统，

而用 return 语句可以从一个函数中返回并将控制权返回给调用该函数的函数。如果在 main()函数中加入 return 语句，那么在执行这条语句后将退出 main()函数并将控制权返回给操作系统，

这样的一条 return 语句和 exit()函数的作用是相同的。

总结：说实话，这段 C 语言我没看懂，对应实验结果没能对应的上，不懂为什么会两段的答案相同，我测试了一下，添加了改变后与改变前输出。以下面两段代码，我就不是很懂，为什么 printevn 和 printf（“%d”，childPid）都会运行。想了好久还是没有想通，真的气，先下一题。

```
void main()
{
pid_t childPid;
switch(childPid = fork()) {
        case 0: /* child process*/
//              printf("%d\n",childPid);
                printenv();
                exit(0);
        default: /* parent process*/
                printf("%d\n",childPid);
//              printenv();
                exit(0);
        }
}
```

```
[09/12/2018 01:24] seed@ubuntu:~/Desktop$ a.out
13523
SSH_AGENT_PID=2404
GPG_AGENT_INFO=/tmp/keyring-yjqTL6/gpg:0:1
TERM=xterm
SHELL=/bin/bash
XDG_SESSION_COOKIE=6da3e071019f67095bc4c5e9000000
WINDOWID=67108869
```

啊啊啊啊，想懂了，case（0）和 default 是两个进程，第一个进程结束后，下一个进程依然执行。

然后又修改代码如下：

```
[09/12/2018 02:24] seed@ubuntu:~/Desktop$ gcc first.c
[09/12/2018 02:25] seed@ubuntu:~/Desktop$ a.out
13841  13840  13713
SSH_AGENT_PID=2404
GPG_AGENT_INFO=/tmp/keyring-yjqTL6/gpg:0:1
TERM=xterm
```

```
_=./a.out
0  13841  1
[09/12/2018 02:25] seed@ubuntu:~/Desktop$
```

```
switch(childPid = fork()) {
        case 0: /* child process*/
                printf("%d  %d  %d\n",childPid,getpid(),getppid());
//              printenv();
                exit(0);
        default: /* parent process*/
                printf("%d  %d  %d\n",childPid,getpid(),getppid());
                printenv();
                exit(0);
        }
}
```

不断变换注释，最终得出结论，该 switch 语句在 fork（）后，创建子进程，然后先运行父进程【从 childPid=13841 得出该结论】，随后执行子进程【由 childPid=0 得出结论】，故清楚得知 fork 的用法，创建子进程与父进程，先运行父进程，后运行子进程。

注意：1、对子进程来说，fork 返回给它 0,但它的 pid 不是 0【由图片可知】；虽然 fork 返回 0 给它，但是它可以调用 getpid()来获取自己的 pid

2、通过查阅资料得知，父进程还是子进程先运行是由操作系统调度决定的，他们都存在，2 个进程一直同时运行，在 fork 之后，他们分别作不同的工作，谁先运行具有随机性。‘’

Task 3: Environment variables and execve()

Step 1. Please compile and run the following program, and describe

your observation. This program Simply execute a program called /usr/bin/env, which prints out the environment variables of the current process.

execve()用来执行

参数 1 是 filename 字符串所代表的文件路径,

参数 2 是利用指针数组来传递给执行文件，并且需要以空指针结束,

参数 3 是传递给执行文件的新环境变量数组。

成功不会返回，失败返回-1。

```
[09/12/2018 05:29] seed@ubuntu:~/Desktop/lab1$ gcc test1.c
[09/12/2018 05:29] seed@ubuntu:~/Desktop/lab1$ a.out
[09/12/2018 05:29] seed@ubuntu:~/Desktop/lab1$
```

Step 2. Now, change the invocation of execve() to the following, and describe your observation.

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

```
[09/12/2018 06:43] seed@ubuntu:~/Desktop/lab1$ gcc test1.c
[09/12/2018 06:44] seed@ubuntu:~/Desktop/lab1$ a.out
SSH_AGENT_PID=2404
GPG_AGENT_INFO=/tmp/keyring-yjqTL6/gpg:0:1
TERM=xterm
SHELL=/bin/bash
```

Step 3. Please draw your conclusion regarding how the new program gets its environment variables.

当 execve()的第三个参数, 新的环境变量数组, 设置为 NULL 的时候, 打印信息为空。当设置为 environ 时，打印出环境变量信息。

Task 4: Environment variables and system()

system()会调用 fork()产生子进程，由子进程来调用/bin/sh 来执行参数 string 字符串所代表的命令，

```
[09/12/2018 05:49] seed@ubuntu:~/Desktop/lab1$ a.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=2229
USER=seed
SSH_AGENT_PID=2404
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed/Desktop
```

Task 5: Environment variable and Set-UID Programs

Step 1. We are going to write a program that can print out all the environment variables in the current process.

```
[09/12/2018 06:44] seed@ubuntu:~/Desktop/lab1$ gcc test5.c
[09/12/2018 06:48] seed@ubuntu:~/Desktop/lab1$ a.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=2229
USER=seed
SSH_AGENT_PID=2404
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed/Desktop
```

Step 2. Compile the above program, change its ownership to root, and make it a Set-UID program.

```
[09/12/2018 07:02] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s test5.c
[09/12/2018 07:03] root@ubuntu:/home/seed/Desktop/lab1# ls -l
total 44
-rwxrwxr-x 1 seed seed 7161 Sep 12 06:48 a.out
-rw-rw-r-- 1 seed seed 2637 Sep 12 00:43 child
-rw-rw-r-- 1 seed seed  505 Sep 12 02:25 first.c
-rw-rw-r-- 1 seed seed 2637 Sep 12 00:43 parent
-rw-rw-r-- 1 seed seed  259 Sep 12 06:43 test1
-rw-rw-r-- 1 seed seed  261 Sep 12 06:43 test1~
-rw-rw-r-- 1 seed seed  187 Sep 12 06:44 test1.c
-rw-rw-r-- 1 seed seed  184 Sep 12 05:17 test1.c~
-rwSrw-r-- 1 seed seed   89 Sep 12 05:49 test5.c
-rw-rw-r-- 1 seed seed    0 Sep 12 05:48 test5.c~
drwxrwxr-x 2 seed seed 4096 Sep 12 05:48 Untitled Folder
```

Step 3. In your Bash shell (you need to be in a normal user account,

not the root account), use the export command to set the following environment variables (they may have already exist):

- PATH
- LD LIBRARY PATH
- ANY NAME (this is an environment variable defined by you, so pick whatever name you want).These environment variables are set in the user's shell process. Now, run the Set-UID program from Step 2 in your shell. After you type the name of the program in your shell, the shell forks a child process, and uses the child process to run the program. Please check whether all the environment variables you set in the shell process (parent) get into the Set-UID child process. Describe your observation. If there are surprises to you, describe them.

运行前：

```
[09/12/2018 07:12] seed@ubuntu:~/Desktop/lab1$ echo $PATH
.:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sb
in:/bin:/usr/games
[09/12/2018 07:12] seed@ubuntu:~/Desktop/lab1$ echo $LD_LIBRARY_PATH
/home/seed/Desktop/lab1
[09/12/2018 07:13] seed@ubuntu:~/Desktop/lab1$ echo $HELLO
/home/seed/Desktop/lab1
[09/12/2018 07:13] seed@ubuntu:~/Desktop/lab1$
```

运行后：

```
HELLO=/home/seed/Desktop/lab1
```

```
PATH=.:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/u
n:/sbin:/bin:/usr/games
```

```
LD_LIBRARY_PATH=/home/seed/Desktop/lab1
```

通过运行 suid 程序，成功将用户 bash 内设置的环境变量写入总系统中。

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

Please compile the above program, and change its owner to root, and make it a Set-UID program.Can you let this Set-UID program run your code instead of /bin/ls? If you can, is your code running with the root privilege? Describe and explain your observations.

```
[09/12/2018 07:29] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s test6.c
[09/12/2018 07:30] root@ubuntu:/home/seed/Desktop/lab1# ls -l
total 48
-rwxrwxr-x 1 seed seed 7161 Sep 12 07:17 a.out
-rw-rw-r-- 1 seed seed 2637 Sep 12 00:43 child
-rw-rw-r-- 1 seed seed  505 Sep 12 02:25 first.c
-rw-rw-r-- 1 seed seed 2637 Sep 12 00:43 parent
-rw-rw-r-- 1 seed seed  259 Sep 12 06:43 test1
-rw-rw-r-- 1 seed seed  261 Sep 12 06:43 test1~
-rw-rw-r-- 1 seed seed  187 Sep 12 06:44 test1.c
-rw-rw-r-- 1 seed seed  184 Sep 12 05:17 test1.c~
-rwSrw-r-- 1 seed seed   89 Sep 12 05:49 test5.c
-rw-rw-r-- 1 seed seed    0 Sep 12 05:48 test5.c~
-rwSrw-r-- 1 seed seed   39 Sep 12 07:29 test6.c
-rw-rw-r-- 1 seed seed    0 Sep 12 07:28 test6.c~
```

想要做下面的实验就必须对 zsh，bash 有一个初步的了解，为此，我做了将 zsh,bash 赋予 suid 的实验（seed 与 root 下分别实验），发现 root 下可以直接进入管理员权限，而 seed 则不行，这是防止 set-uid 机制被滥用所设置的权限管理，sh 与 zsh 连接的实验，图示如下：

ln –s zsh sh

```
[09/12/2018 19:45] seed@ubuntu:/bin$ sh
$ sudo sh
# exit
$ exit
[09/12/2018 19:46] seed@ubuntu:/bin$ sudo sh
#
```

下面进入正文：

实验六我做了 4 个小时，在学习原理之余就是被网上的的很多前辈误

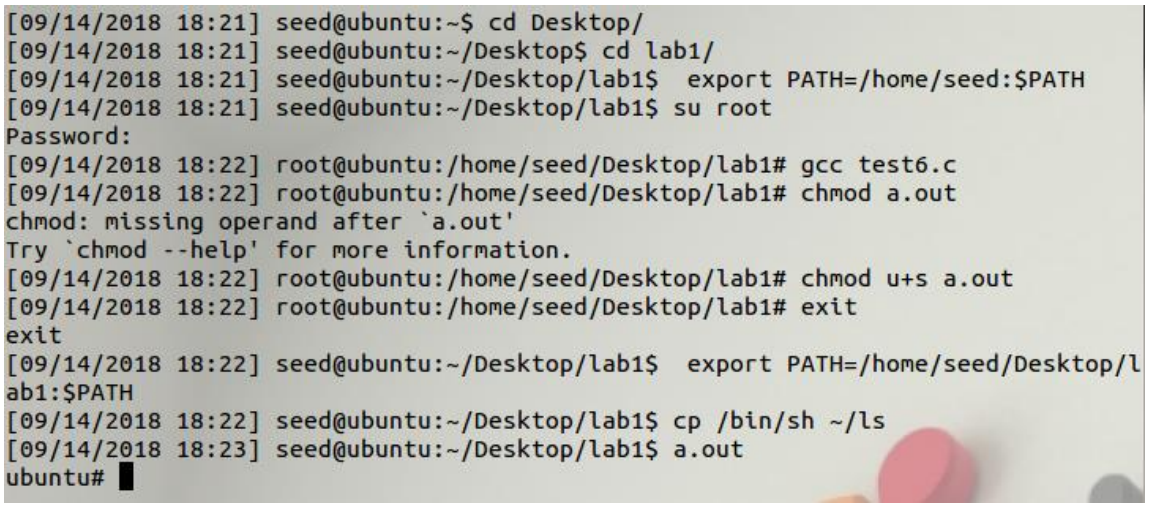
导了，他们的经验有问题，我做的第一次成功如下，我赋予了你们较多的权限 test6.c 的 suid，a.out 的 suid，root 权限的 a.out

```
[09/14/2018 18:02] seed@ubuntu:~$ export PATH=/home/seed:$PATH
[09/14/2018 18:02] seed@ubuntu:~$ su root
Password:
[09/14/2018 18:02] root@ubuntu:/home/seed# cd Desktop/
[09/14/2018 18:03] root@ubuntu:/home/seed/Desktop# cd lab1/
[09/14/2018 18:03] root@ubuntu:/home/seed/Desktop/lab1# gcc test6.c
[09/14/2018 18:03] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s test6.c
[09/14/2018 18:03] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s a.out
[09/14/2018 18:03] root@ubuntu:/home/seed/Desktop/lab1# exit
exit
[09/14/2018 18:03] seed@ubuntu:~$  export PATH=/home/seed/Desktop/lab1:$PATH
[09/14/2018 18:04] seed@ubuntu:~$ cp /bin/sh ~/ls
[09/14/2018 18:04] seed@ubuntu:~$ a.out
ubuntu#
```

我不在 root 权限下编译 test6.c 发现失败，得出结论，root 权限的 a.out 是必须条件

```
[09/14/2018 18:15] seed@ubuntu:~$ cd Desktop/
[09/14/2018 18:16] seed@ubuntu:~/Desktop$ cd lab1/
[09/14/2018 18:16] seed@ubuntu:~/Desktop/lab1$ gcc test6.c
[09/14/2018 18:16] seed@ubuntu:~/Desktop/lab1$  export PATH=/home/seed:$PATH
[09/14/2018 18:17] seed@ubuntu:~/Desktop/lab1$ su root
Password:
buntu One
[09/14/2018 18:17] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s test6.c
[09/14/2018 18:17] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s a.out
[09/14/2018 18:17] root@ubuntu:/home/seed/Desktop/lab1# exit
exit
[09/14/2018 18:18] seed@ubuntu:~/Desktop/lab1$  export PATH=/home/seed/Desktop/
ab1:$PATH
[09/14/2018 18:18] seed@ubuntu:~/Desktop/lab1$ cp /bin/sh ~/ls
[09/14/2018 18:19] seed@ubuntu:~/Desktop/lab1$ a.out
This is the Z Shell configuration function for new users,
zsh-newuser-install.
You are seeing this message because you have no zsh startup files
(the files .zshenv, .zprofile, .zshrc, .zlogin in the directory
~).  This function can help you with a few settings that should
make your use of the shell easier.
```

加入 root 权限，减少 test6.c 的 suid 权限



得出结论想要利用 suid 的权限发起攻击需要两个条件，第一：该文件需要在 root 条件下编译，拥有 root 权限，第二：该文件拥有 suid 权限。

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

Step 1: Compile the above program, make root its owner, and change it to a Set-UID program. The program will use system() to invoke the command. 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?

```
[09/14/2018 18:53] seed@ubuntu:~/Desktop/lab1$ ls -l file
-rwxr--r-- 1 seed seed 0 Sep 14 18:42 file
[09/14/2018 18:54] seed@ubuntu:~/Desktop/lab1$ ls -l a.out
-rwsr-xr-x 1 root root 7274 Sep 14 18:39 a.out
[09/14/2018 18:54] seed@ubuntu:~/Desktop/lab1$ a.out "file;mv file newfile"
[09/14/2018 18:54] seed@ubuntu:~/Desktop/lab1$ ls file
ls: cannot access file: No such file or directory
[09/14/2018 18:54] seed@ubuntu:~/Desktop/lab1$ ls newfile
newfile
[09/14/2018 18:55] seed@ubuntu:~/Desktop/lab1$
```

Step 2: Comment out the system(command) statement, and uncomment the execve() statement;

the program will use execve() to invoke the command. Compile the program, and make it Set-UID(owned by root). Do your attacks in Step 1 still work? Please describe and explain your observations.

```
[09/14/2018 18:56] seed@ubuntu:~$ cd Desktop/
[09/14/2018 18:56] seed@ubuntu:~/Desktop$ cd lab1
[09/14/2018 18:57] seed@ubuntu:~/Desktop/lab1$ su root
Password:
[09/14/2018 18:57] root@ubuntu:/home/seed/Desktop/lab1# gcc test8.c
[09/14/2018 18:59] root@ubuntu:/home/seed/Desktop/lab1# chmod u+s a.out
[09/14/2018 18:59] root@ubuntu:/home/seed/Desktop/lab1# exit
exit
[09/14/2018 18:59] seed@ubuntu:~/Desktop/lab1$ a.out "file;mv file newfile"
/bin/cat: file;mv file newfile: No such file or directory
```

不会有效，在 step1 之所以有效，是具有 root 权限的 system 在执行了 cat file 文件后，还会接着执行 mv file file_new 命令。而 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。

Task 9: Capability Leaking

Compile the following program, change its owner to root, and make it a Set-UID program. Run the

program as a normal user, and describe what you have observed. Will the file /etc/zzz be modified?

Please explain your observation.