69

C语言程序

```c
//69.c
#include<unistd.h>
int main()
{

execvp("/challenge/embryoio_level69",NULL);

}
```

shell脚本

```shell
#69.sh
/home/hacker/69
```

70

> WELCOME! This challenge makes the following asks of you:
>
> - the challenge checks for a specific parent process : shellscript

> - the challenge will check that the environment is empty (except LC_CTYPE, which is impossible to get rid of in some cases)
> - the challenge will check that env[KEY] holds value VALUE (listed to the right as KEY:VALUE) : 172:tcjsryvtjg

题目描述多少沾点矛盾。

```
//70.c
#include<unistd.h>
int main()
{
    char *const envp[]=
{"172=tcjsryvtjg",NULL};

execvpe("/challenge/embryoio_level70",NULL,en
vp);
}
```

sh:

```
#70.sh

#!/bin/bash
/home/hacker/70
```

71

> WELCOME! This challenge makes the following asks of you:
>
> - the challenge checks for a specific parent process : shellscript
> - the challenge will check that the environment is empty (except LC_CTYPE, which is impossible to get rid of in some cases)
> - the challenge will check that argv[NUM] holds value VALUE (listed to the right as NUM:VALUE) : 267:dklpmoxism
> - the challenge will check that env[KEY] holds value VALUE (listed to the right

> as KEY:VALUE) :
> 297:qasnbmvqzq

要求脚本，空环境，参数，环境变量参数

shellscript就直接执行程序。

71.c:

非配内存就OK了，不必向里面填充一些内容。

```
//71.c



#include <stdio.h>
#include <string.h>
#define _GNU_SOURCE
#include<unistd.h>
#include<stdlib.h>
int main()
{
    char **argv;
    argv=(char**)malloc(300*sizeof(char*));
    for(int i=0;i<300;i++)
    {
        argv[i]=
(char*)malloc(20*sizeof(char));
        const char source[10]="ILoveYou";
        //char* strcpy(char* destination,
const char* source);
        //strcpy(argv[i],source);
```

```
    }
    char pwd[20]="dklpmoxism";
    argv[267]=pwd;

    char *const envp[]=
{"297=qasnbmvqzq",NULL};

execvpe("/challenge/embryoio_level71",argv,en
vp);
}
```

72重定向输入

> WELCOME! This challenge makes the following asks of you:
>
> - the challenge checks for a specific parent process : shellscript
> - the challenge will check that input is redirected from a specific file path : iuqgdo
> - the challenge will check that it is running in a specific current working directory : /tmp/hqqonw

C语言版本：

在/tmp/hqqonw文件夹下执行操作。

这个可以正常运行getflag。

```
//72.c

#define _GNU_SOURCE
//for no warning of execvpe
 #include <fcntl.h>
//open
#include<unistd.h>
//dup2

int main()
{
    int fd=open("iuqgdo",O_RDONLY);
    dup2(fd,0);

execvpe("/challenge/embryoio_level72",NULL,NU
LL);
}
```

试着修改PWD，在~下执行程序。

```
#define _GNU_SOURCE
//for no warning of execvpe
 #include <fcntl.h>
```

```
//open
#include<unistd.h>
//dup2

int main()
{
    //int
fd=open("/tmp/hqqonw/iuqgdo",O_RDONLY);
    //这样写的话，会提示重定向的文件不对。

    //int fd=open("iuqgdo",O_RDONLY);
    //这样写的话，找不到文件，重定向失败

    char *const envp[]=
{"PWD=/tmp/hqqonw",NULL};
    dup2(fd,0);

execvpe("/challenge/embryoio_level72",NULL,en
vp);
}
```

截图如下：

```
[ADVICE] File descriptors are inherited from the parent, unless the FD_CLOEXEC is set by the parent on the file
descriptor.
[ADVICE] For security reasons, some programs, such as python, do this by default in certain cases. Be careful if
 you are
[ADVICE] creating and trying to pass in FDs in python.
[FAIL] You did not satisfy all the execution requirements.
[FAIL] Specifically, you must fix the following issue:
[FAIL]    You have redirected the wrong file for stdin (/tmp/hqqonw/iuqgdo instead of iuqgdo).
```

python版本：

```python
from pwn import *
fd=open("iuqgdo","r")
#print(fd)
p=process(['/challenge/embryoio_level72'],std
in=fd)
p.interactive()
#print(p.readall().decode())
```

执行失败，说是：

```
[*] Switching to interactive mode
WELCOME! This challenge makes the following asks of you:
- the challenge checks for a specific parent process : shellscript
- the challenge will check that input is redirected from a specific file path : iuqgdo
- the challenge will check that it is running in a specific current working directory : /tmp/hqqonw

ONWARDS TO GREATNESS!

[INFO] This challenge will now perform a bunch of checks.
[INFO] If you pass these checks, you will receive the flag.
[TEST] Performing checks on the parent process of this process.
[TEST] Checking to make sure the process is a non-interactive shell script.
[FAIL] You did not satisfy all the execution requirements.
[FAIL] Specifically, you must fix the following issue:
[FAIL]   Process interpreter must be 'sh' or 'bash'. Yours is: python3.8
Exception in thread Thread-2:
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
    self.run()
```

但我还是用的shell脚本执行python程序呀？

73PWD|CWD

考察父进程和子进程不在同一个folder里面，也就是有不同的cwd。

pwd作为二进制程序，打印当前的cwd

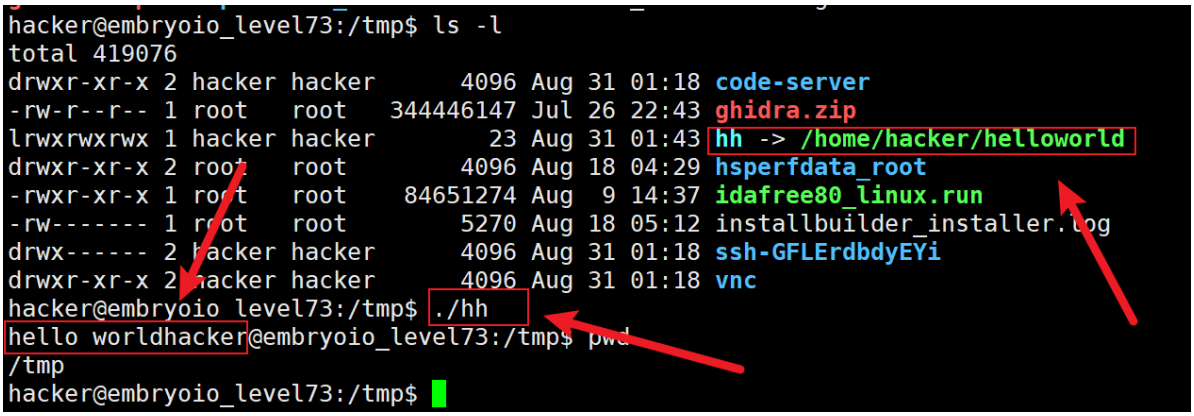
$PWD 是个系统变量（环境变量）

cwd是current working directory

> 一句话解释： 都指某个进程运行时所在的 目录.
>
> $PWD 是个系统变量
>
> pwd 是linux 自带的命令. 全称: pathname of the current working directory.
>
> cwd: 不是系统自带的命令，但是属于系统的属性 . 全称: current working directory . 不但在 /proc/{id} 这个目录下可以看到 cwd，在很多其他的编程语言中也可以看到(例如grunt)

软连接

是可以执行的。



寻找`pwd`：

```
/usr/local/sbin:无
/usr/local/bin:无
/usr/sbin:无
/usr/bin:有
/sbin:无
/bin:有
```

但是有一个奇怪的地方：

对于一个软链接：

```
lrwxrwxrwx 1 hacker hacker        13 Aug 31
01:47 fake -> /home/hacker/
```

我们进入这个软链接，如图所示。

执行两种pwd：

```
hacker@embryoio_level73:/tmp/fake$ pwd -P
/home/hacker
hacker@embryoio_level73:/tmp/fake$ pwd
/tmp/fake
```

我们看一看pwd的来源：

```
hacker@embryoio_level73:/tmp/fake$ pwd
/tmp/fake
hacker@embryoio_level73:/tmp/fake$ which pwd
/usr/bin/pwd
```

pwd有两个来源，我们分别执行一下，再执行一下pwd：

```
hacker@embryoio_level73:/tmp/fake$ /usr/bin/pwd
/home/hacker
hacker@embryoio_level73:/tmp/fake$ /bin/pwd
/home/hacker
hacker@embryoio_level73:/tmp/fake$ pwd
/tmp/fake
```

好奇怪。

pwd，来自/usr/bin/pwd，得到表面地址。

但是执行/usr/bin/pwd，却得到真实地址。

/bin/pwd，得到真实地址。

exp

这个exp一行都不能少

```
#!/bin/bash
#注意创建文件夹

cd /tmp/lfiiwq
/challenge/$HOSTNAME &
cd /home/hacker
sleep 1
```

虽然不太理解，

&是放在后台运行，但是输入输出还是会在运行结束后输出出来。

这样做不行

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/wait.h>
#include<unistd.h>
```

```
int main() {
        char* const argv[]=
{"/challenge/embryoio_level73",NULL};
        char* const env[]=
{"PWD=/tmp/lfiiwq",NULL};


 execvpe("/challenge/embryoio_level73",argv,e
nv);
        return 0;

}
```

74

> list,Python 数据类型
>
> []

exp

```python
from pwn import *
new_argv=[]
for i in range(100):
    new_argv.append("new_argv")## 使用
append() 添加元素

new_argv[0]="/challenge/embryoio_level74"
new_argv[92]="mgzvmrsity"
print(new_argv)

p=process(new_argv)
p.interactive()
```

75

多少有点扯淡了。

先写个C文件：

```c
#include<unistd.h>

int main()
{

 execvpe("/challenge/embryoio_level75",NULL,N
ULL);
}
```

再写个python：

```python
from pwn import *
p=process("/home/hacker/75")
p.interactive()
```

然后执行py文件就可以了。

76

先写个C文件：

```c
#include<unistd.h>

int main()
{
    char *const envp[]=
{"277=gjvxigddql",NULL};
    //int execvpe(const char *file, char
*const argv[],char *const envp[]);

 execvpe("/challenge/embryoio_level76",NULL,e
nvp);
}
```

再写个python：

```python
from pwn import *
p=process("/home/hacker/76")
p.interactive()
```

然后执行py文件就可以了。

77

c:

```
#include<unistd.h>
#include<stdlib.h>

int main()
{
    char **argv;
    argv=(char**)malloc(300*sizeof(char*));
    for(int i=0;i<300;i++)
    {
        argv[i]=
(char*)malloc(20*sizeof(char));
    }
    argv[0]="/challenge/embryoio_level77";
    argv[111]="wzuodrmxzc";

    char *const envp[]=
{"37=lwkwvosuxm",NULL};
    //int execvpe(const char *file, char
*const argv[],char *const envp[]);

 execvpe("/challenge/embryoio_level77",argv,e
nvp);
}
```

py:

```
from pwn import *
p=process("/home/hacker/77")
p.interactive()
```

78

如图，在这个文件夹下创建一个py文件和一个新文件

```
hacker@embryoio_level78:/tmp/kbwrer$ ls -l
total 4
-rw-r--r-- 1 hacker hacker 105 Aug 31 13:23 78.py
-rw-r--r-- 1 hacker hacker   0 Aug 31 13:19 qqljfy
hacker@embryoio_level78:/tmp/kbwrer$
```

py:

```
from pwn import *
fd=open("qqljfy","r")
p=process("/challenge/embryoio_level78",stdin
=fd)
p.interactive()
```

79

exp:

直接修改cwd即可

```
from pwn import *
p=process(["/challenge/embryoio_level79"],cwd
="/tmp/fvhjep")
p.interactive()
```

错误示范：

```
#!/bin/bash
mkdir /tmp/fvhjep
cd /tmp/fvhjep
/challenge/$HOSTNAME &
cd /home/hacker
sleep 1
```

再写个py，执行这个shell脚本，但是不行。

80

随便改改代码：

```
#include<unistd.h>
#include<stdlib.h>

void pwncollege()
{
    char **argv;
    argv=(char**)malloc(300*sizeof(char*));
    for(int i=0;i<300;i++)
    {
        argv[i]=
(char*)malloc(20*sizeof(char));
    }
    argv[0]="/challenge/embryoio_level80";
    argv[148]="mucbnxuvor";
```

```
    char *const envp[]={NULL};
    //int execvpe(const char *file, char
*const argv[],char *const envp[]);

 execvpe("/challenge/embryoio_level80",argv,e
nvp);
}

int main()
{
    int childPid;

    if(fork()==0)
    {
        pwncollege();
    }
    else
    {
        waitpid(childPid,NULL,NULL);
    }
}
```

81

没有意思，子进程空参数而已。

```
void pwncollege()
{
  char *const argv[]={NULL};
  char *const envp[]={NULL};
  //int execvpe(const char *file, char *const
argv[],
  //char *const envp[]);
  execvpe("/challenge/embryoio_level81",argv,
envp);
}
```

82

```
void pwncollege()
{
    char *const argv[]={NULL};
    char *const envp[]=
{"15=kqhcymhaqr",NULL};
    //int execvpe(const char *file, char
*const argv[],char *const envp[]);

 execvpe("/challenge/embryoio_level82",argv,e
nvp);
}
```

83

```
void pwncollege()
```

```
{

    char **argv;
    argv=(char**)malloc(300*sizeof(char*));
    for(int i=0;i<300;i++)
    {
        argv[i]=
(char*)malloc(20*sizeof(char));
    }
    argv[0]="/challenge/embryoio_level83";
    argv[175]="aziqyhetgq";

    //char *const argv[]={NULL};
    char *const envp[]=
{"314=uidurcakvz",NULL};
    //int execvpe(const char *file, char
*const argv[],char *const envp[]);

 execvpe("/challenge/embryoio_level83",argv,e
nvp);
}
```

84

Function:chdir()

```
        #include <unistd.h>
        int chdir(const char *path);
```

更改cwd，当然，在<unistd.h>中，显然是想配合着exec家族使用。

exp:

```
void pwncollege()
{
    childPid=getpid();
    chdir("/tmp/cymwyc");
    int fd=open("dsvmdj",O_RDONLY);
    char *const argv[]={NULL};
    char *const envp[]={NULL};
    dup2(fd,0);
    //int execvpe(const char *file, char
*const argv[],
    //char *const envp[]);

 execvpe("/challenge/embryoio_level84",argv,e
nvp);

}
```

85

```
void pwncollege()
{
    childPid=getpid();
    char *const argv[]={NULL};
    char *const envp[]={NULL};

    //int execvpe(const char *file, char
*const argv[],
    //char *const envp[]);
    chdir("/tmp/vjvjpy");

 execvpe("/challenge/embryoio_level85",argv,e
nvp);

}
```

86

真的不理解题意

> - the challenge checks for a specific parent process : shellscript
> - the challenge will force the parent process to solve a number of arithmetic problems : 1

> - the challenge will use the following arithmetic operations in its arithmetic problems : +*
> - the complexity (in terms of nested expressions) of the arithmetic problems : 1

exp

```
#!/bin/bash
/challenge/embryoio_level86
```

输入数字就OK了

87

做数学题？不理解

```
#!/bin/bash
/challenge/embryoio_level87
```

88|修改argv[0]

修改argv[0]，也就是修改进程名。

对一个程序创建一个软链接，然后执行这个软链接。

在tmp里面创建的软链接：

```
hacker@embryoio_level88:/tmp$ ls -l znfjiy
lrwxrwxrwx 1 hacker hacker 27 Sep 20 12:53 znfjiy -> /challenge/embryoio_level88
hacker@embryoio_level88:/tmp$
```

然后写一个.sh文件：

```
#!/bin/bash
/tmp/znfjiy
~
```

执行就OK了。

89

在88的基础上

有一个博主的做法，我跟是不理解：

```
export PATH=.:$PATH
#将当前目录作为首个PATH变量
#然后写脚本就可以了
```

错误exp：

```
#!/bin/bash
./tazjmc
#或者
./tazjmc
```

都会显示：

```
[FAIL]    argv[0] is not 'tazjmc' (it seems
to be './tazjmc', instead).
```

正确exp：

```
#!/bin/bash
tazjmc

#或者

tazjmc
```

90
