

# XCTF 攻防世界 Reverse新手题（getit）

原创

[sherlockjobs](#) 于 2020-12-13 16:24:05 发布 651 收藏 1

分类专栏: [CTF](#) [网络安全](#) [渗透测试](#) 文章标签: [反汇编](#) [反编译](#) [网络安全](#) [渗透测试](#)

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/weixin\\_43456810/article/details/111127072](https://blog.csdn.net/weixin_43456810/article/details/111127072)

版权



[CTF](#) 同时被 3 个专栏收录

19 篇文章 0 订阅

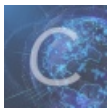
订阅专栏



[网络安全](#)

27 篇文章 0 订阅

订阅专栏



[渗透测试](#)

32 篇文章 0 订阅

订阅专栏

## XCTF 攻防世界 Reverse新手题（getit）

首先, 判断文件有没有加壳, 是多少位的, 利用exeinfoPE可以得知, 程序并没有加壳, 并且是64位的elf程序  
直接用IDA64打开, 找到main函数, F5反编译一下:

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v3; // a1
    __int64 v5; // [rsp+0h] [rbp-40h]
    int i; // [rsp+4h] [rbp-3Ch]
    FILE *stream; // [rsp+8h] [rbp-38h]
    char filename[8]; // [rsp+10h] [rbp-30h]
    unsigned __int64 v9; // [rsp+28h] [rbp-18h]

    v9 = __readfsqword(0x28u);
    LODWORD(v5) = 0;
    while ( (signed int)v5 < strlen(s) )
    {
        if ( v5 & 1 )
            v3 = 1;
        else
            v3 = -1;
        *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
        LODWORD(v5) = v5 + 1;
    }
    strcpy(filename, "/tmp/flag.txt");
    stream = fopen(filename, "w");
    fprintf(stream, "%s\n", u, v5);
    for ( i = 0; i < strlen(&t); ++i )
    {
        fseek(stream, p[i], 0);
        fputc(*(&t + p[i]), stream);
        fseek(stream, 0LL, 0);
        fprintf(stream, "%s\n", u);
    }
    fclose(stream);
    remove(filename);
    return 0;
}

```

简单分析一下代码，看到最后的for循环，猜测很有可能是输出flag的，所以先找到strlen()函数的地址400824，之后去汇编代码的界面，找到400824，确实是strlen函数：

```

.text:0000000000400824      call     _strlen
.text:0000000000400829      cmp     rbx, rax
.text:000000000040082C      jnb     loc_4008B5
.text:0000000000400832      mov     eax, [rbp+var_3C]
.text:0000000000400835      cdqe

```

回到伪代码，for循环的第一个函数是fseek()，因此jnb loc\_4008B5很有可能就是fseek()，这一点确定之后，接下来便可以动态调试把400832设为断点进行尝试

```

pwndbg 文件名 //进入动态调试界面
b *0x400832 //设置断点的地址
r //运行

```

可以看到，运行一下直接就可以看到flag信息了：

```
Breakpoint 1, 0x00000000400832 in main ()
Building dependency tree
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
-----[ REGISTERS ]-----
*RAX 0x2b
*RBX 0x0
*RCX 0x0
*RDX 0x6010e0 (t) ← 'SharifCTF{b70c59275fcfa8aebf2d5911223c6589}'
*RDI 0x601100 (t+32) ← '11223c6589}'
*RSI 0x400978 ← optional, byte ptr [rax] /* '\n' */
*R8 0x602480 ← '*****\n'
*R9 0x2c
*R10 0x6f
*R11 0x0
*R12 0x400660 (_start) ← xor ebp, ebp
*R13 0x0
*R14 0x0
*R15 0x0
*RBP 0x7fffffffdfc0 → 0x4008f0 (__libc_csu_init) ← push r15
*RSP 0x7fffffffdfc0 ← 0x20 /* ' ' */
*RIP 0x400832 (main+220) ← mov eax, dword ptr [rbp - 0x3c]
```

[https://blog.csdn.net/weixin\\_43456810](https://blog.csdn.net/weixin_43456810)



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)