

# XCTF reverse maze

原创

YenKoc 于 2020-02-05 15:43:11 发布 264 收藏 1

分类专栏: [XCTF](#)

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/YenKoc/article/details/104183346>

版权

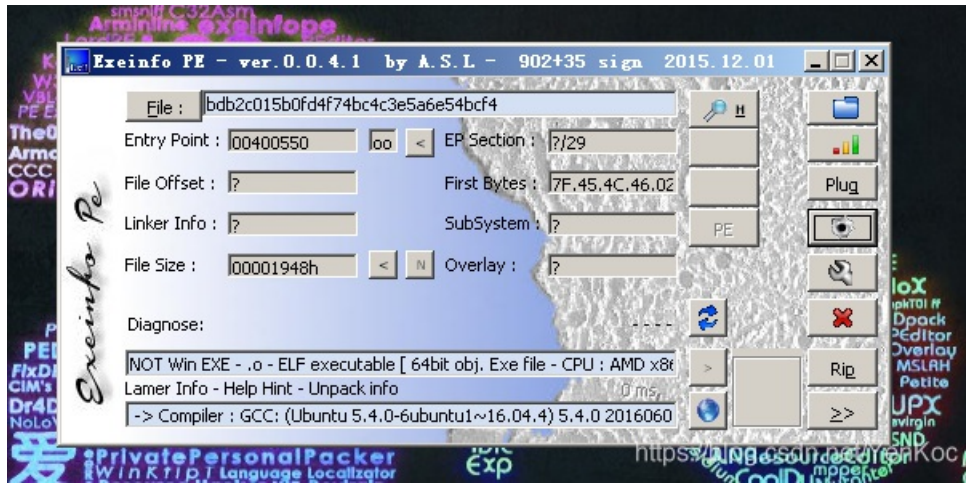


[XCTF 专栏收录该内容](#)

26 篇文章 2 订阅

订阅专栏

## 一.查壳



## 二.拖入ida64, 静态调试, 找到主函数F5反编译

```

4 signed int v4; // eax@5
000C 5 bool v5; // bp@5
0000 6 bool v6; // al@8
0000 7 const char *v7; // rdi@19
0000 8 __int64 v9; // [sp+0h] [bp-28h]@1
0000 9
0000 10 v9 = 0LL;
0000 11 puts("input flag:");
0000 12 scanf("%s", &s1, 0LL);
0000 13 if ( strlen(&s1) != 24 || strncmp(&s1, "nctf{", 5uLL) || *(&byte_6010BF + 24) != 125 )
0000 14 {
0000 15 LABEL_22:
0000 16     puts("Wrong flag!");
0000 17     exit(-1);
0000 18 }
0000 19 v3 = 5LL;
0000 20 if ( strlen(&s1) - 1 > 5 )
0000 21 {
0000 22     while ( 1 )
0000 23     {
0000 24         v4 = *(&s1 + v3);
0000 25         v5 = 0;
0000 26         if ( v4 > 'N' )
0000 27         {
0000 28             v4 = v4;
0000 29             if ( v4 == '0' )
0000 30             {
0000 31                 v6 = sub_400650(&v9 + 1);
0000 32                 goto LABEL_14;
0000 33             }
0000 34             if ( v4 == 'o' )
0000 35             {
0000 36                 v6 = sub_400660(&v9 + 1);
0000 37                 goto LABEL_14;
0000 38             }
0000 39         }
0000 40     else
0000 41     {
0000 42         v4 = v4;
0000 43         if ( v4 == '.' )
0000 44         {
0000 45             v6 = sub_400670(&v9);
0000 46             goto LABEL_14;
0000 47         }
0000 48         if ( v4 == '0' )
0000 49         {
0000 50             v6 = sub_400680(&v9);

```

<https://blog.csdn.net/YenKoc>

二.1 思路分析（逆向是真的费时间，每个函数都要分析过去）：

## 1.发现每个if最终都会进入LABEL-15

```
000| 55 | }
    | 56 | LABEL_15:
    | 57 | if ( !sub_400690(asc_601060, SHIDWORD(v9), v9) )
    | 58 | goto LABEL_22;
    | 59 | if ( ++v3 >= strlen(&s1) - 1 )
    | 60 | {
    | 61 |     if ( v5 )
    | 62 |         break;
```

点进去，看看这个函数是干啥的。

```
1 | __int64 __fastcall sub_400690(__int64 a1, int a2, int a3)
2 | {
3 |     __int64 result; // rax@1
4 |
5 |     result = *(a1 + a2 + 8LL * a3);
6 |     LOBYTE(result) = result == '.' || result == '#';
7 |     return result;
8 | }
```

这里基本可以推理出a3是行，a2是列了。

那么我们的迷宫一定是一个8列的矩阵。

得知这点，退出去，再分析。

```
EL_15:
    if ( !sub_400690(asc_601060, SHIDWORD(v9), v9) )
        goto LABEL_22;
```

v9是行，SHIDWORD函数是指的下一个字节，得知这点，从上面分析可知：

就将四个函数点进去就可以判断方向了。

举个例子：

```
1 | bool __fastcall sub_400650(_DWORD *a1)
2 | {
3 |     int v1; // eax@1
4 |
5 |     v1 = (*a1)--;
6 |     return v1 > 0;
7 | }
```

相对应，其他方向，O:下,o:上,0:左,.:右。

2.将迷宫打印出来，python脚本如下。

```
maze=" ***** * **** * **** * *** *# *** ** * ** *****"
res=""
count=0
for str in maze:
    res+=str
    count+=1
    if count%8==0:
        print(res)
        res=""
```



3.由于我们是从左上角开始的，开始我们的路线  
个人感觉答案不唯一的。。。没办法。》-《。。。  
根据迷宫最后得到的flag: `nctf{o0oo00O000oooo...OO}`