CTF中的迷宫问题

原创

dusk！ 于 2021-04-20 19:43:51 发布 446 收藏 1

分类专栏： reverse

本文链接：https://blog.csdn.net/qq_51999322/article/details/115872229

reverse 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

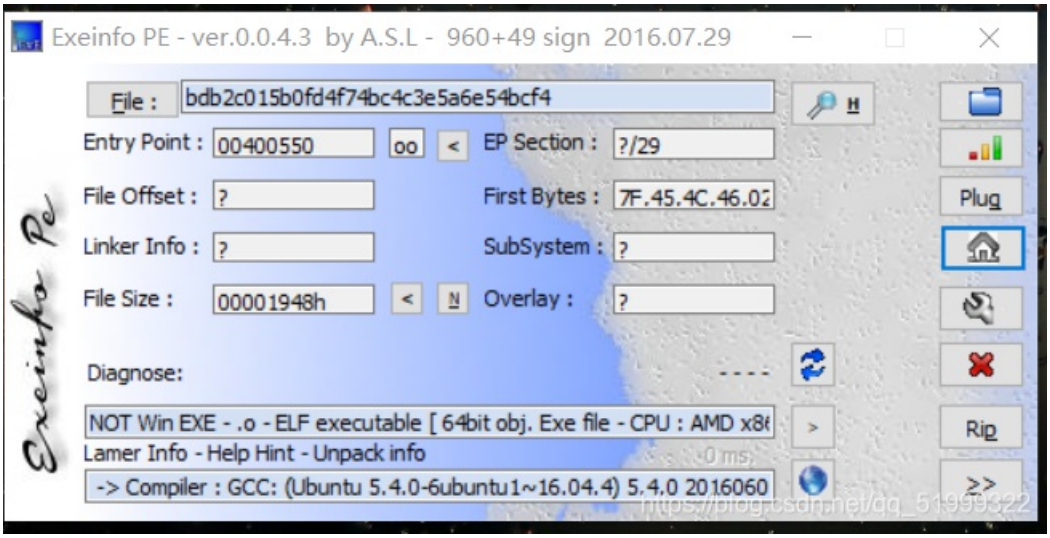
这是拿的WIKI的迷宫图片，扫眼一看确实没有什么思路，再加上密密麻麻的字符就会让人感觉头大，还必须再这里找到迷宫的路线，好家伙，真的是给做题人不留点空间想象，不过我做的这道题确实迷宫不算太大，至少是8x8的宽度。

我是看了这个链才有所顿悟，如果我下面说的还不算太明白可以看看这个链接https://ctf-wiki.org/reverse/maze/maze/.

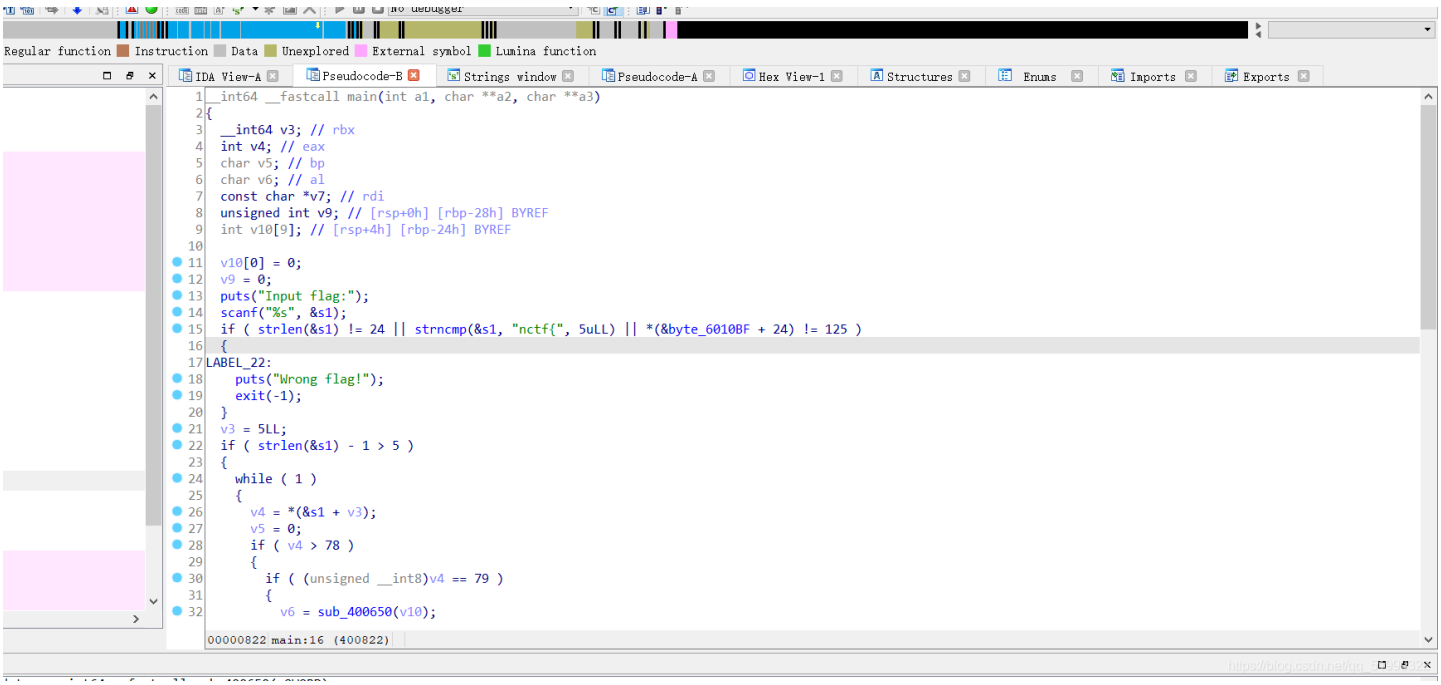
题目场景： 暂无

题目附件： 附件1

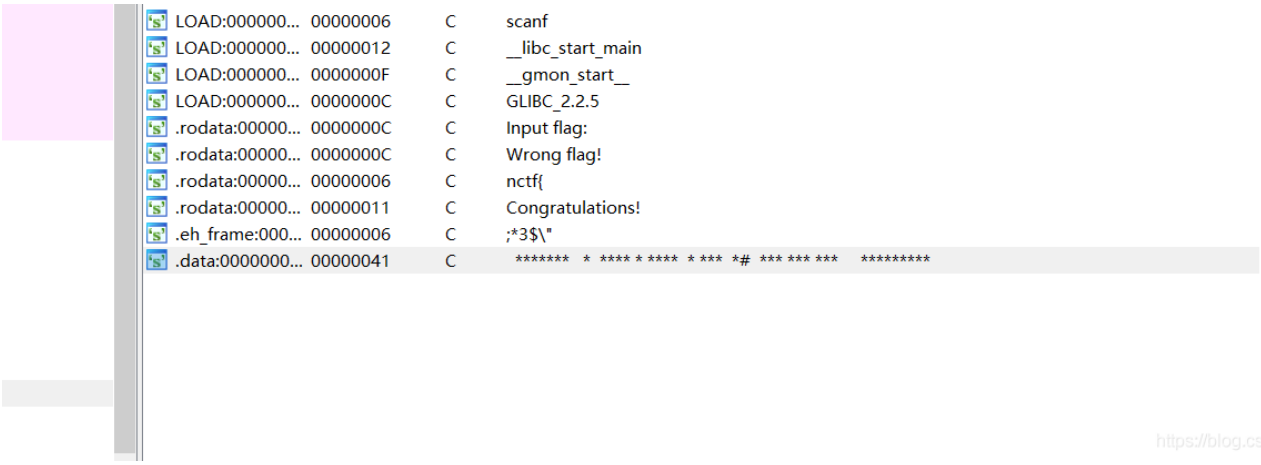
现在做reverse我的第一步都是成了查壳，习惯性的看它有没有加壳，然后拖到IDA中查看main函数



经过查看这是一个64位的ELF文件，我们在把他拖到64位IDA中查看main函数



我们再按shift+F12打开字符串窗口查看一下有没有可以的字符串



然后我们会看到一串可以的字符串

```
.data:0000000000601060 00000041 C   *******   *  **** * ****  * ***  *#  *** *** ***     *********
```

然后我们开始分析main函数

```
  v10[0] = 0;      //是一个数组，代表为列
  v9 = 0;          //代表为行
  puts("Input flag:");
  scanf("%s", &s1);
  if ( strlen(&s1) != 24 || strncmp(&s1, "nctf{", 5uLL) || *(&byte_6010BF + 24) != 125 )
  {
LABEL_22:
    puts("Wrong flag!");
    exit(-1);
  }//这里说不是括号里的东西就是错误答案，所以可得到len(s1) = 24, s1开头必须为nctf
```

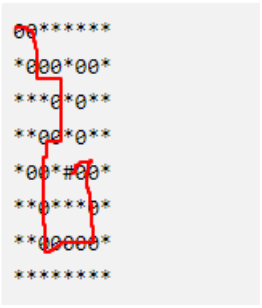
```
v3 = 5LL;
 if ( strlen(&s1) - 1 > 5 )
 {
   while ( 1 )
   {
     v4 = *(&s1 + v3);
     v5 = 0;
     if ( v4 > 78 )
     {
       if ( (unsigned __int8)v4 == 79 )//aciss中79为‘O’
       {
         v6 = sub_400650(v10);//跟进这个函数里面显示的是左移，因为函数中为--又因为是v10为列，所以左移
         goto LABEL_14;
       }
       if ( (unsigned __int8)v4 == 111 )//aciss中111为‘o’
       {
         v6 = sub_400660(v10);//跟进这个函数里面显示的是右移
         goto LABEL_14;
       }
     }
     else
     {
       if ( (unsigned __int8)v4 == 46 )//aciss中46为‘.’
       {
         v6 = sub_400670(&v9);//跟进这个函数里面显示的是下移，因为函数中为--，又因为是v9为行所以下移
         goto LABEL_14;
       }
       if ( (unsigned __int8)v4 == 48 )//aciss中48为‘0’
       {
         v6 = sub_400680(&v9);//跟进这个函数里面显示的是上移
LABEL_14:
         v5 = v6;
         goto LABEL_15;
       }
     }
```

然后就是我们走迷宫的时候了，因为这是个8x8迷宫，所以要把它弄成8x8表格

```
.data:0000000000601060 00000041 C   *******   *  **** * ****  * ***  *#  *** *** ***     *********
```

```
00******
*000*00*
***0*0**
**00*0**
*00*#00*
**0***0*
**00000*
********
```

从左上脚出发走到#号，上下左右就是前面分析的0oO.这四个来控制



因为我们前面分析获取正确的flag前要开头必须为nctf
所以flag就是nctf{o0oo00O000oooo…OO}