攻防世界 maze

原创

别害怕我在 于 2021-08-06 15:07:36 发布 128 收藏

分类专栏： CTF逆向reverse新手

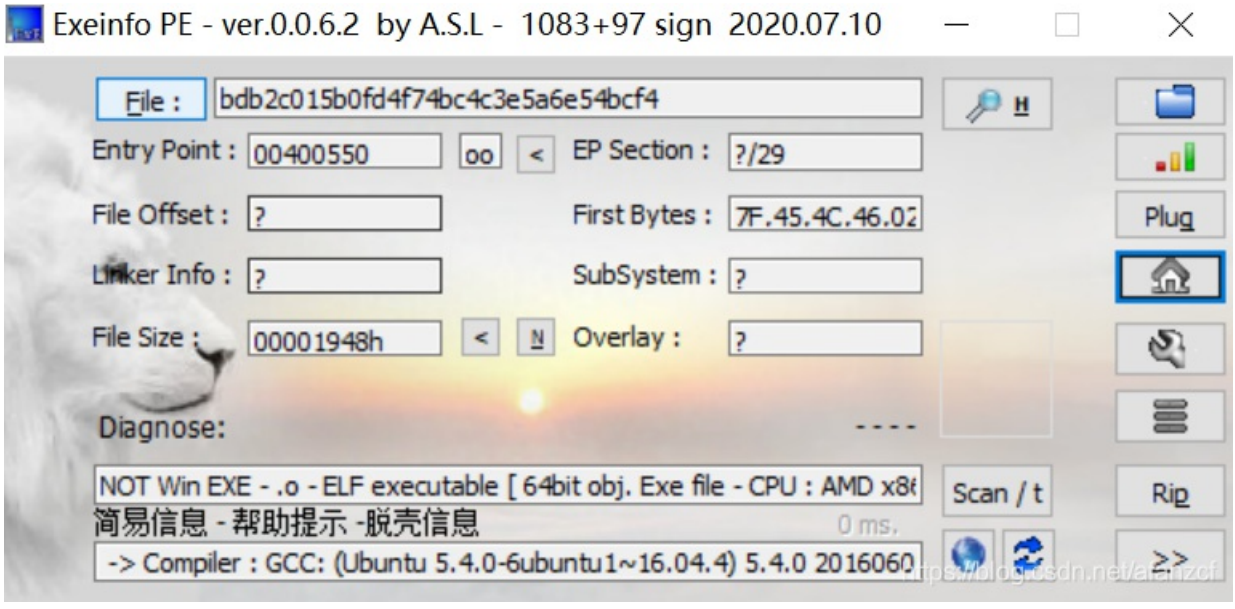
本文链接：https://blog.csdn.net/afanzcf/article/details/119455357

---

title: 攻防世界
date: 2021年8月6日 09点48分
tags: 攻防世界
categories: 攻防世界

在做了Bugku love 之后，对reverse题有了一点新的感悟，于是又对攻防世界 maze发起了进攻。这个题一开始看到题目，地图？迷宫？跟我之前做的一道题有点像，BUUCTF 不一样的flag。

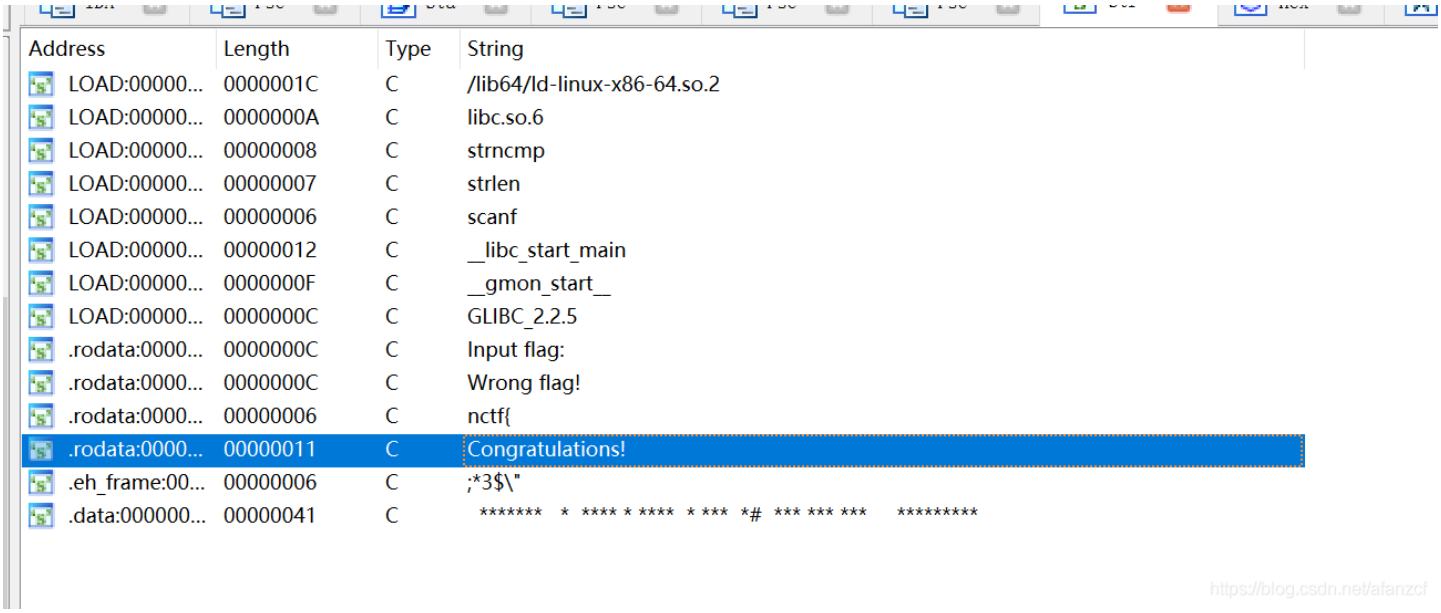
1、PE分析



不是exe，而是ELF，也就是linux的。64位，直接拖到IDA64打开。

2、IDApro分析

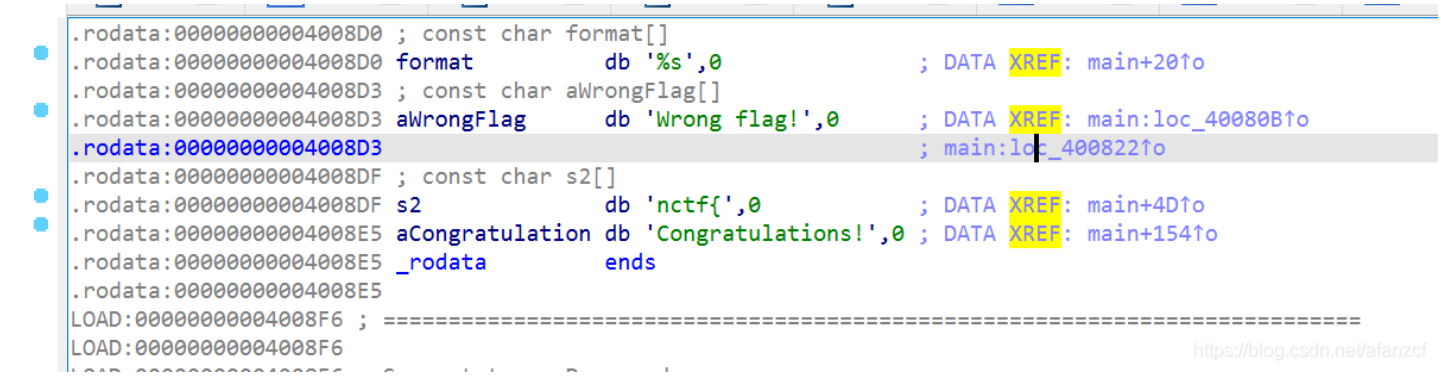
1、shift + F12查看字符串窗口



Address	Length	Type	String
LOAD:00000...	0000001C	C	/lib64/ld-linux-x86-64.so.2
LOAD:00000...	0000000A	C	libc.so.6
LOAD:00000...	00000008	C	strncmp
LOAD:00000...	00000007	C	strlen
LOAD:00000...	00000006	C	scanf
LOAD:00000...	00000012	C	__libc_start_main
LOAD:00000...	0000000F	C	__gmon_start__
LOAD:00000...	0000000C	C	GLIBC_2.2.5
.rodata:0000...	0000000C	C	Input flag:
.rodata:0000...	0000000C	C	Wrong flag!
.rodata:0000...	00000006	C	nctf{
.rodata:0000...	00000011	C	Congratulations!
.eh_frame:00...	00000006	C	;*3$\"
.data:000000...	00000041	C	******* * **** * **** * *** *# *** *** *** *********

我们可以看到有一个congratulations，还可以看到最下面的一串*号，根据之前做的那道迷宫题，可以得知，那一串就是地图。这里先跟踪congratulations。

2、CTRL + X 交叉引用



```
.rodata:00000000004008D0 ; const char format[]
.rodata:00000000004008D0 format          db '%s',0              ; DATA XREF: main+20↑o
.rodata:00000000004008D3 ; const char aWrongFlag[]
.rodata:00000000004008D3 aWrongFlag      db 'Wrong flag!',0     ; DATA XREF: main:loc_40080B↑o
.rodata:00000000004008D3                                        ; main:loc_400822↑o
.rodata:00000000004008DF ; const char s2[]
.rodata:00000000004008DF s2              db 'nctf{',0           ; DATA XREF: main+4D↑o
.rodata:00000000004008E5 aCongratulation db 'Congratulations!',0 ; DATA XREF: main+154↑o
.rodata:00000000004008E5 _rodata         ends
.rodata:00000000004008E5
LOAD:00000000004008F6 ; ===========================================================================
LOAD:00000000004008F6
```


3、F5查看伪代码

Pseudocode-D | IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A | Strings window

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  __int64 v3; // rbx
  int v4; // eax
  bool v5; // bp
  bool v6; // al
  const char *v7; // rdi
  unsigned int v9; // [rsp+0h] [rbp-28h] BYREF
  int v10[9]; // [rsp+4h] [rbp-24h] BYREF

  v10[0] = 0;
  v9 = 0;
  puts("Input flag:");
  scanf("%s", &s1);                             // 输入的flag是s1
  if ( strlen(&s1) != 24 || strncmp(&s1, "nctf{", 5uLL) || *(&byte_6010BF + 24) != '}' )
  {                                             // 计算&s1的长度如果不等于24，或者让nctf跟&s1对比，或者是那个地址+24，不等于125
LABEL_22:
    puts("Wrong flag!");                        // 输出错误的flag
    exit(-1);                                   // 以-1为返回值退出
  }
  v3 = 5LL;                                     // v3 = 5
  if ( strlen(&s1) - 1 > 5 )                    // 如果&s1的长度-1 > 5，则进入死循环
  {
    while ( 1 )
    {
      v4 = *(&s1 + v3);                         // v4 = &s1 + v3
      v5 = 0;
      if ( v4 > 'N' )                           // 如果v4的值 > 78(N)，则进入下一个if语句
      {
        if ( (unsigned __int8)v4 == 'O' )       // 如果v4 = 79（O），则跳转到LABEL_14这个地址去
        {
          v6 = sub_400650(v10);                 // v1 = (*a1)--;   v10--
          goto LABEL_14;
        }
        if ( (unsigned __int8)v4 == 'o' )       // 如果v4的值 = 111（o），则跳转到LABEL_14这个地址
        {
          v6 = sub_400660(v10);                 // v1 = *a1 + 1;
                                                //   *a1 = v1;   v10+1
          goto LABEL_14;
        }
      }
      else
      {
        if ( (unsigned __int8)v4 == '.' )       // 如果 = 46（.），则跳转LABEL_14
        {
          v6 = sub_400670(&v9);                 // v1 = (*a1)--;   v9--
          goto LABEL_14;
        }
        if ( (unsigned __int8)v4 == '0' )       // 如果 = 48（0），则跳转LABEL_14
        {
          v6 = sub_400680((int *)&v9);          // v1 = *a1 + 1;
                                                //   *a1 = v1;    v9+1
LABEL_14:
          v5 = v6;                              // 让v5 = v6，然后跳转去15的地址
          goto LABEL_15;
        }
      }
LABEL_15:
      if ( !(unsigned __int8)sub_400690((__int64)asc_601060, v10[0], v9) )// result = *(unsigned __int8 *)(a1 + a2 + 8LL * a3)
                                                //   LOBYTE(result) = (_DWORD)result == 32 || (_DWORD)result == 35;
        goto LABEL_22;
      if ( ++v3 >= strlen(&s1) - 1 )
      {
        if ( v5 )
          break;
LABEL_20:
        v7 = "Wrong flag!";
        goto LABEL_21;
      }
    }
  }
  if ( asc_601060[8 * v9 + v10[0]] != '#' )     //   *******   *  **** * ****  * ***  *#  *** *** ***     *********
    goto LABEL_20;
  v7 = "Congratulations!";
LABEL_21:
  puts(v7);
  return 0LL;
```


注释是一开始写的注释，还是分析的不够好，看事物太注重一步一步的分析了，导致整个分析没有连起来，分析的太过死板。

这里可以看到，第一句的if语句的意思就是，flag是s1，且长度为24，而且开头结尾是是nctf{ }，

后面进入循环，能够得到四个字符，O、o、0、. 当时没有想到这四个字符就是控制上下左右来走迷宫的，

其次这几个字符跟进去之后，不知道怎么分析，然后就是LABEL_15:，这个if语句这里看不懂，然后在知道那一串字符就是字符串的时候，不知道怎么把迷宫地图搞出来。

当时分析的时候，卡在了这几个问题，导致自己没有分析出这个题

4、查看大佬们的wp，复盘自己的分析

查看完大佬们的wp之后，我的问题所在就是，从一开始就分析的太死板，太去在乎语句表达的表面意思，没联合在一起分析，然后不知道怎么判断四个字符的方向，准确的说是不知道那四个字符跟上下左右挂钩，明知道是迷宫题还没想到这一点，只想着去分析，然后不知道怎么判断迷宫的x轴和y轴，在知道迷宫是那一串字符，不知道是怎么分的。

问题一，怎么判断O、o、0、. 它们的方向各是什么

```
    {
      v6 = sub_400650(v10);                 // 【rsp+4】【rbp-24】  x轴-1  左移
      goto LABEL_14;
    }
    if ( (unsigned __int8)v4 == 'o' )
    {
      v6 = sub_400660(v10);                 // 【rsp+4】【rbp-24】 x轴+1 右移
      goto LABEL_14;
    }
  }
  else
  {
    if ( (unsigned __int8)v4 == '.' )
    {
      v6 = sub_400670(&v9);                 // 【rsp+0】【rbp-28】 y轴-1  上移
      goto LABEL_14;
    }
    if ( (unsigned __int8)v4 == '0' )
    {
      v6 = sub_400680(&v9);                 // 【rsp+0】【rbp-28】  y轴+1  下移
EL_14:
      v5 = v6;
```



```
1 bool __fastcall sub_400650(_DWORD *a1)
2 {
3   int v1; // eax
4
5   v1 = (*a1)--;                         // 这时O，跟进的函数，结果是是减1
6   return v1 > 0;
7 }
```

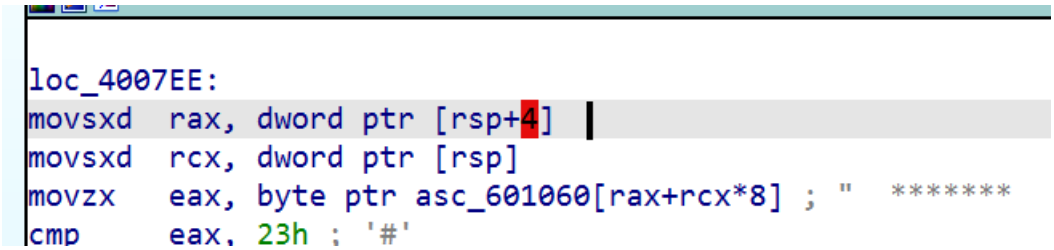

还有其他三个函数，分别点进去，则可以得到，四个字符所代表的方向位。

左减右加，上减下加

O左移 o右移 .上移 0下移

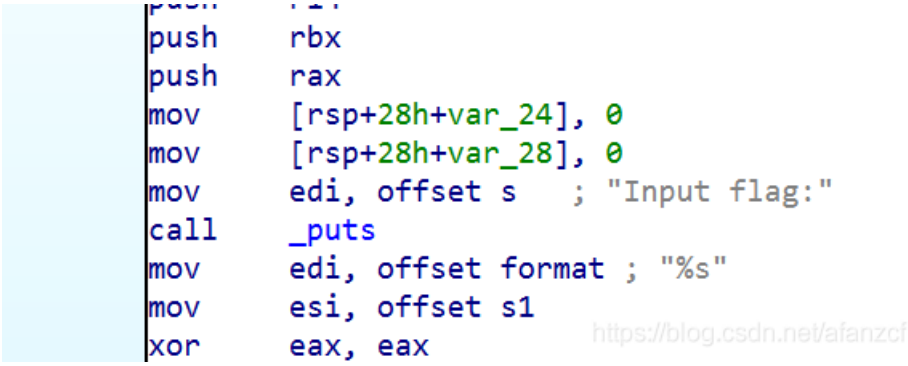
问题二，怎么判断出来，迷宫的x轴和y轴

问了工作室的好哥哥，终得以解惑。



加4的就是 x 轴，不加4 的就是 y 轴。

这个问题也可以在main函数的头部初始化的时候看到。



所以迷宫的起点就是（0，0）

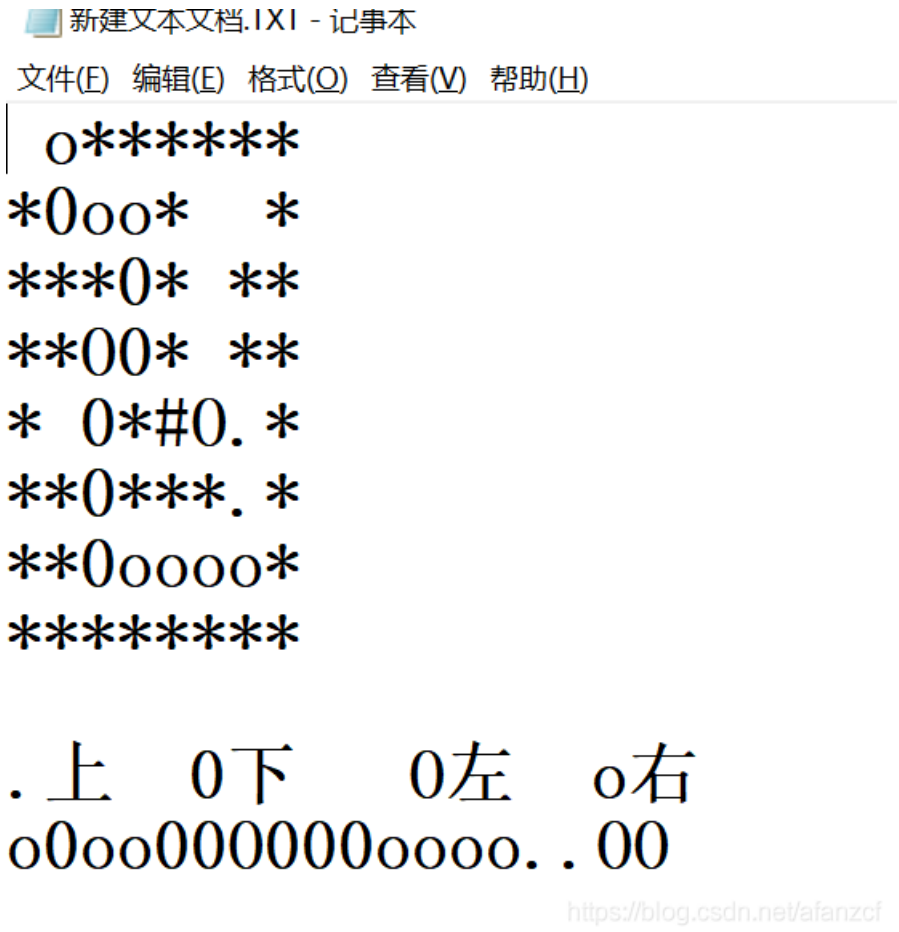
问题三，迷宫地图是怎么找出来的

sub_400690函数中是a2+a3*8,即a3表示行，a2表示列

a3 通过edx传递

a2 通过esi传递

行×8？再去看这个字符串，发现是64位，正好就是8×8



*是边界，空格是通路，＃号是终点。

```
67      }
68    }
69    if ( asc_601060[8 * v9 + v10[0]] != '#' )
70      goto LABEL_20;
71    v7 = "Congratulations!";
72  LABEL_21:
73    puts(v7);
74    return 0LL;
75  }
```

最终得到flag：

nctf{o0oo00O000oooo…OO}
此处flag不知怎么回事，编辑的时候是两个点，显示的时候，非显示三个点，删一个点，就变成一个点了，正确flag是中间两个点，此处显示问题。

5、总结

这个题，至此就做完了，通过BUUCTF 不一样的flag 让我一下就知道了那一串字符串是迷宫，但是还是无从下手，题目中，增加了x轴和y轴，已经四个字符，但是没有告诉上下左右，需要自己去判断，题目难度在于x轴，y轴是怎么判断的，好像是一种固定的形式，+4为x轴，不＋为y轴，目前只能如此判别了。通过这个题目，对于迷宫类的题目，又是多了几分感觉，下次再碰到迷宫题的时候，不会特别迷茫了。同时，这个题的分析，与其他人的分析，有所差异，别人分析，能得出题目信息，自己分析则是按着伪代码一步一步的分析，不能进行很好的联动，这是甚有不足的。

继续努力！