百度杯二月Reverse场Project的writeup

原创

Flying_Fatty 于 2017-02-28 09:42:35 发布 797 收藏
分类专栏： CTF之旅 reverse

本文链接：https://blog.csdn.net/kevin66654/article/details/58585538


CTF之旅 同时被 2 个专栏收录
84 篇文章 2 订阅
订阅专栏


reverse
24 篇文章 0 订阅
订阅专栏

在看完题解之后学习的一发，然后全部按照自己的writeup再次整理做题，巩固一发

打开IDA和OD，希望搜索到有用的字符串（Input the flag的提示信息），但是发现根本找不到，搜索strings会找到这个：

根据查到的资料，这24个字符构成了一种编码方式：base24

学习链接：base24

所以找到这个字符串的使用的地方：00404820函数处

可以看到如下几个特征：

```
v6 = strchr("BCDFGHJKMPQRTVWXY2346789", *(_BYTE *)(v4 + 2 * v5));
v7 = strchr("BCDFGHJKMPQRTVWXY2346789", *(_BYTE *)(v4 + 2 * v5 + 1));

*(&v15 + v5++) = (unsigned int)&Str[-(_BYTE)v7 + 23] | 16
                               * ((_BYTE)v6 - (unsigned int)"BCDFGHJKMPQRTVWXY2346789");
```

这儿是很明显的两位一个值，然后把v6和v7拼起来的（相当于base24解码），最终存在了v15里面（v5是个变量值，作为计数器使用的）

那么，我们需要构造的输入，是两位两位一组的，然后使用base24解码，可以得到某种需要处理的值

```
switch ( (unsigned int)v15 >> 4 )
{
  case 0xEu:
    LOBYTE(v3) = v15 & 0xF;
    ((void (__thiscall *)(int))loc_404B00)(v3);
    goto LABEL_14;
  case 0xCu:
    sub_404AA0(v15 & 0xF);
    goto LABEL_14;
  case 0xDu:
    v9 = dword_40A018 - v8;
    goto LABEL_11;
  case 0xFu:
    v9 = v8 + dword_40A018;
```

看到这里四个方向，然后v8的值是v15 & 0xf，主要是底下一个是加号，一个是减号，然后的跳转都是LABEL_11

上面的两个跳转都是LABEL_14，可以猜想到是不是4个方向，上下左右控制的（百度杯好多这种方向题）

那么，v8是控制步长的，v15>>4是控制方向的

看到0xC的404AA0函数，有这种的代码：

```
byte_40A6EC += a1;
dword_40A018 += 16 * a1 + a1;
```

可以猜想，40A6EC这个值是记录我在迷宫里总共走了多少步的，40A018这个值是记录我在迷宫里的位置的（这个17，就会是一行总共17个元素，相当于向下走了一行）

根据这样的方向，0xC对应向下，0xE对应向上，0xD对应向左，0xF对应向右

那么地图的数据从哪儿来的？

```
v10 = (unsigned __int8)byte_4076A0[v9];
v11 = (unsigned __int8)byte_4072C0[v9];
byte_40A6EC += v8;
v12 = (v10 ^ v11) - (unsigned __int8)byte_407410[v9];
dword_40A018 = v9;
v3 = v12 - (unsigned __int8)byte_407550[v9];
if ( v3 || !v8 )
  goto LABEL_13;
```

看到v3这个值，是由四个数据构成的，然后我们找到这些数据的所在地方，根据数据计算得到地图：

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1
1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1
1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1
1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1
1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1
1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1
1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1
1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1
1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1
1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1
1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1
1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

所以起点在（1，0），终点在（15，16）

然后再看到我们的结束条件是什么？

```
if ( byte_40A6EC != 54 || v9 != 271 )
  LOBYTE(v2) = dword_40A6F4(v3, 0);
```

我们在迷宫中总的移动步数是54步，最终的位置坐标是271，也就是（15，16）

那么我们只需要得到行走的路径（使用ACM的广搜或者深搜都是可以的）

然后得到往上走几步，往下走几步的一个移动路径，使得可以满足：从起点到终点的移动总步数为54步

因为这个会是整数值，比如3表示往右走3步，17表示往下走一步

然后根据题意得四个方向以及base24编码的加密原理，对我们的移动路径加密即可

路径为：

```
( 1 , 0 )--> ( 1 , 1 )--> ( 1 , 2 )--> ( 1 , 3 )--> ( 2 , 3 )--> ( 3 , 3 )--> ( 3 , 4 )--> ( 3 , 5 )--> ( 2
```

路径整数值为：

```
goto = [3,34,2,-34,6,68,-6,34,-2,34,-2,68,4,34,11]
```

最终的flag为：

```
X6T7X7W7X2T4V2T7V7T7V7T4X4T7XT
```

然后附录上自己的py代码：

```
d_6a0 = [0x87,0xae,0x99,0x3c,0x59,0x41,0x2e,0x1d,0x84,0x23,0x30,0x23,0x26,0x10,0x83,0x5,0x29,0x9d,0xa0,0x16
d_2c0 = [0x10,0x27,0x1b,0x44,0x2e,0x30,0x99,0x7f,0x1f,0xb8,0xb1,0xab,0x78,0x90,0x3a,0x58,0x9e,0x1d,0x9b,0x5
d_410 = [0x4c,0x11,0x29,0x16,0x5d,0x51,0x4d,0x3e,0x6f,0x3e,0x59,0x65,0x57,0x2c,0x5d,0x2c,0x46,0x56,0x3b,0x4
d_550 = [0x4a,0x77,0x58,0x61,0x19,0x1f,0x69,0x23,0x2b,0x5c,0x27,0x22,0x6,0x53,0x5b,0x30,0x70,0x2a,0x0,0xa,0

t = []
for i in xrange(289):
    tmp = ((d_6a0[i] ^ d_2c0[i]) - d_410[i]) & 0xff
    if tmp == d_550[i]:
        t += [i]
#print t

s = ''
mp = ''
for i in xrange(289):
    if i in t:
        s += '0'
        mp += '0'
    else:
        s += '1'
        mp += '1'
    if i % 17 == 16:
        s += '\n'
    else:
        s += ' '
print s

mapsize = 30
dirsize = 4
a = [[0] * mapsize for i in range(mapsize)]
dx = [1,0,0,-1]
dy = [0,1,-1,0]

def printans(i,j):
    global point
    point += 1
    x = i - dx[a[i][j] - 1]
```

```
    y = j - dy[a[i][j] - 1]
    if x == 1 and y== 0:
        print '(',x,',',y,')-->',
        return
    else:
        printans(x,y)
    print '(',x,',',y,')-->',

def printans2(i,j):
    global goto
    x = i - dx[a[i][j] - 1]
    y = j - dy[a[i][j] - 1]
    if x == 1 and y== 0:
        goto += [a[i][j]]
        return
    else:
        printans2(x,y)
        #print a[i][j],
        goto += [a[i][j]]

def getans(i,j):
    global a
    #global flag
    #if flag == True:
        #return
    if i==15 and j==16:
        print 'Yes'
        printans(i,j)
        print '(15 , 16)'
        printans2(i,j)
        #flag = True
        return
    else:
        for k in range(0,dirsize):
            newi = i + dx[k]
            newj = j + dy[k]
            if (newi > 0 and newi < 17 and newj > 0 and newj < 17 and mp[newi*17+newj] == '0' and a[newi][n
                a[newi][newj] = k+1
                getans(newi,newj)

point = 1
goto = []
simple = []
getans(1,0)
print goto

dirc = [0,17,1,-1,-17]
tmp = 0
number = 0
for i in goto:
    if number == 0:
        tmp = i
        number += 1
    elif i == tmp:
        number += 1
    else:
        simple += [dirc[tmp] * number]
        tmp = i
        number = 1
```

```
simple += [dirc[tmp] * number]
print simple

string = 'BCDFGHJKMPQRTVWXY2346789'
def decode(num):
    global flag
    if num > 0:
        if num % 17 == 0:
            flag += string[0xc] + string[23 - num / 17]
        else:
            flag += string[0xf] + string[23 - num]
    else:
        if num % 17 == 0:
            flag += string[0xe] + string[23 + num / 17]
        else:
            flag += string[0xd] + string[23 + num]
flag = ''
for i in simple:
    decode(i)
print flag
```

题目链接：

百度杯二月Reverse题目

官方题解：

writeup