

I春秋360_Reverse_登山_Writeup

原创

Flying Fatty 于 2017-04-23 12:13:56 发布 2530 收藏

分类专栏: CTF之旅 reverse

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/kevin66654/article/details/70493566>

版权



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

这个题的Hint: 求二维矩阵左上往下走, 可以向下, 也可以向右下的最大值

(打过ACM的话, 就知道是dp的数字三角形模型题)

IDA里静态分析, 发现程序逻辑比较简单, 先有一个init的初始化函数, 然后就是对我们的输入进行check

分析init:

```
memset(a, 0, (size_t)&unk_7D8320);
for ( i = 0; i <= 1013; ++i )
{
    for ( j = 0; j <= i; ++j )
    {
        v0 = my_rand();
        v1 = 1014 * i + j;
        a[2 * v1] = v0;
        dword_48B044[2 * v1] = 0;
    }
}
```

两个for循环得到了一个1014*1014的矩阵, 根据我们的hint, 这就是我们需要运算的矩阵, 所以我们需要构造出矩阵

这儿有个my_rand() 函数, 但是没有关系, 进去看一下, 发现随机种子固定, 所以每次生成的都是同一个矩阵

OD中下断点, 观察数的变化

0048B040 A4 05 48 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 |

地址	HEX 数据
0048CFF0	EC 1B 85 00 00 00 00 00 85 38 00 00 00 00 00 00
0048D000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

根据地址的变化, 我们可以算偏移, 得到运算完毕之后的结束地址:

```
>>> 0x48cff0-0x48b040
8112
>>> hex(8112*1014+0x48b040) in
'0xc63360'
```

所以我们等程序跑完, 然后把矩阵从数据窗口扣出来

00C63320	FE	DE	23	00	00	00	00	58	E2	1D	00	00	00	00	00
00C63330	F0	CD	64	00	00	00	00	E6	59	4C	00	00	00	00	00
00C63340	F9	24	5C	00	00	00	00	90	83	44	00	00	00	00	00
00C63350	00	BB	62	00	00	00	00	AB	DB	8A	00	00	00	00	00
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

右键，二进制，二进制复制，然后弄到txt中去

然后就是数据格式处理，需要整理成矩阵的形式，这个没啥特别的，等会有代码，先说逻辑

看到check：

进去就是v6的一个赋值，猜测是矩阵最终的最大值

然后有个getstep，有个map，我们在init里面也看到这个map了，说明输入的字符集合只可能 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/里面，而且顺序已经被打乱，所以我们要弄明白，进入init，有64次打乱，我们只需要知道最终的结果，可以不理会打乱的过程

找到这个地方：

```
00401467 8055 9B      lea edx, dword ptr ss:[ebp-0x65]
00401468 8B45 DC      mov eax, dword ptr ss:[ebp-0x24]
0040146D 01D0          add eax, edx
0040146F 890424        mov dword ptr ss:[esp], eax
00401472 B9 6033C600    mov ecx, 1.00C63360
00401477 E8 14400700    call 1.00475490
0040147C 83EC 04        sub esp, 4
0040147F 89C2          mov edx, eax
00401481 8B45 DC        mov eax, dword ptr ss:[ebp-0x24]
堆栈地址=0028FE33, (ASCII "'EBCDAFCHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'")
edx=00000041
```

发现与原来字符串不一样了，那么很明显是执行了某种变换操作，那我们让它执行64次，然后把得到的字符串抠出来，得到一个对应关系：

```
temp = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
string = 'EIFd6gwN42LR1vGrBYCnzHTStDqm+kxZpQVioj9078es3U1AKhXcfybPM5W/0aJu'
```

来分析程序逻辑：总共是1014个位，1个字符决定6个01，那么我们的输入长度是 $1014/6=169$ 个

要逆向得到原来的字符串，首先需要处理矩阵，然后使用dp得到最大值的地方，然后6个01为一组，得到一个字符

先说下dp的理论：（可自行百度：数字三角形dp）

怎么走才是最大：一个位置的最大值，只可能由两个位置过来：上面，或者左上，那么比较一下就好

6个01怎么得到一个字符呢？联想到2的6次方是64正好是0-63的范围，然后看到这个地方：

```
v4 = a1;
v5 = 32;
v2 = (_DWORD *)std::map<char,int,std::less<char>,std::allocator<std::pair<char const,int>>>::operator[](&v4);
return (*v2 & (v5 >> a2 % 6)) != 0;
```

这个a2是从0到5不断循环的，当a2等于0，就是判断*v2&32，那就是判断*v2有没有32这个位（二进制中 2^5 这个位）

所以就相当于是判断32，16，8，4，2，1这六个数在不在*v2里面

然后反向查询下

代码如下：

```
...
f = open('matrix.txt','r')

a = [[0 for i in range(1020)] for i in range(1020)]
index = 0

while True:
    line = f.readline()
    line = line.replace(' ','')
    line = line.replace('\n','')

    s = line[6:8] + line[4:6] + line[2:4] + line[0:2]
    a[index//1014][index%1014]=int(s,16)
    index += 1

    s = line[22:24] + line[20:22] + line[18:20] + line[16:18]
    a[index//1014][index%1014]=int(s,16)
    index += 1

    s = line[38:40] + line[36:38] + line[34:36] + line[32:34]
    a[index//1014][index%1014]=int(s,16)
    index += 1

    s = line[54:56] + line[52:54] + line[50:52] + line[48:50]
    a[index//1014][index%1014]=int(s,16)
    index += 1

#if index % 100000 == 0:
#    print index
if index >= 1014 * 1014:
    break
f.close()

dp = [[0 for i in range(1020)] for i in range(1020)]
choose = [[0 for i in range(1020)] for i in range(1020)]

dp[0][0] = a[0][0]
for i in xrange(1,1014):
    dp[i][0] = dp[i-1][0] + a[i][0]
    for j in xrange(1,i+1):
        dp[i][j] = max(dp[i-1][j-1],dp[i-1][j]) + a[i][j]
        if dp[i-1][j-1] > dp[i-1][j]:
            choose[i][j] = 1
        else:
            choose[i][j] = 0

Max = 0
Pos = 0
for i in range(0,1014):
    if dp[1013][i] > Max:
        Max = dp[1013][i]
        Pos = i
print Max
```

```

step = [0 for i in range(1020)]
i = 1013
while (i > 0):
    step[i] = choose[i][Pos]
    Pos -= choose[i][Pos]
    i -= 1
for i in range(0,1014):
    print step[i],',',
...
step = [0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 0 ,
print len(step)

temp = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
num = [0x2f,0x10,0x12,0x19,0,0x2,0xE,0x15,0x1,0x3E,0x30,0x0A,0x38,0x7,0x27,0x37,
       0x21,0x0B,0x17,0x16,0x2D,0x22,0x3A,0x32,0x11,0x1F,0x3D,0x36,0x33,0x3,0x2A,
       0x34,0x05,0x31,0x23,0x25,0x1D,0x2E,0x1B,0x13,0x24,0x20,0x1A,0x0F,0x2B,0x18,
       0x3F,0x0D,0x06,0x1E,0x35,0x14,0x3C,0x0C,0x09,0x2C,0x08,0x39,0x04,0x28,0x29,
       0x26,0x1C,0x3B]
flag = ''
for i in range(0,1014,6):
    number = step[i] * 32 + step[i+1] * 16 + step[i+2] * 8 + step[i+3] * 4 + step[i+4] * 2 + step[i+5]
    for pos in range(0,64):
        if number==num[pos]:
            flag += temp[pos]
print flag

string = 'EIFd6gwN42LR1vGrBYCnzHTStDqm+kxZpQvioj9078es3U1AKhXcfybPM5W/0aJu'
print hex(string.find('A'))

```



[创作打卡挑战赛 >](#)

[赢取流量/现金/CSDN周边激励大奖](#)