XCTF asong

原创

pipixia233333 于 2019-05-09 23:10:26 发布 613 收藏
分类专栏： 逆向之旅

本文链接：https://blog.csdn.net/qq_41071646/article/details/90043300

说实话 这个题 还是比较难受的

当我看到 有三个文件的时候 我有点慌 感觉又是什么玩意的时候

但是仔细分析了一下 感觉这个题 还有点好玩

一个是elf 文件 一个是 文本文件 一个是out out 里面是需要我们解密的数据

然后 我们先分析一下elf

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char *that_girl_index; // ST00_8
  const char *input; // ST08_8

  that_girl_index = malloc(188uLL);
  input = malloc(80uLL);
  init_s();
  scanf_s(input);                                   // 字符最多100个
  flag_ss(input);
  get_file_thatgirl_index("that_girl", that_girl_index);
  sub_400E54(input, that_girl_index);
  return 0LL;
}
```


这里没有什么好说的我们 get_file_thatgirl_index 这个函数其实就是存的频率

```
{
  int v2; // eax
  int *v4; // [rsp+0h] [rbp-20h]
  char buf; // [rsp+13h] [rbp-Dh]
  int fd; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v7; // [rsp+18h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  fd = open(a1, 0, a2, a1);
  while ( read(fd, &buf, 1uLL) == 1 )
  {
    v2 = get_index(buf);                            // 10-36 是'A'-'Z' or 'a'-'z'
    ++v4[v2];                                       // 这里是一个表  统计了 不同字符的频率
  }
  return close(fd);
}
```


0 -9 下标是0-9 小写大写的下标都一样

然后 我们看一下下面的函数

```
  int v4; // [rsp+1Ch] [rbp-44h]
  char v5[56]; // [rsp+20h] [rbp-40h]
  unsigned __int64 v6; // [rsp+58h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  v4 = strlen(a1);
  for ( i = 0; i < v4; ++i )
    v5[i] = a2[get_index(a1[i])];
  sub_400D33(v5);
  sub_400DB4(v5, v4);
  sub_400CC0(v5, "out", v4);
  return __readfsqword(0x28u) ^ v6;
}
```


这里就很重要了 如果get不到 v5的值 那么我们就没有办法 进行下一步了 重点 就是 a2 的值哪里来的

这里我跑了一个脚本 模仿了 get_file_thatgirl_index 这个函数

```
#!/usr/bin/python3
#coding=utf8
if __name__ =='__main__':
    f=open("that_girl")
    l=[]
    for line in f:
        for i in line:
            if ord(i)>=ord('A') and ord(i)<=ord('Z'):
                l.extend(chr(ord(i)+32))
            else:
                l.extend(i)
    summ={}
    while '\n' in l:
        l.remove('\n')
    for ch in l:
        if(ch not in summ):
            summ[ch]=0
        summ[ch]+=1
    print(summ)
    data={}
    data=sorted(summ.items(),key=lambda x:x[0])
    print(data)

    f.close()
```

然后 用 dbg 调试了一下 发现这两个相同 那么就很好解释了

```
pwndbg> x/30gx 0x603038
0x603038:       0x0000001e00000068      0x00000001d0000000f
0x603048:       0x000000130000000a9     0x0000004300000026
0x603058:       0x000000000000003c      0x0000002700000014
0x603068:       0x000000760000001c      0x0000001a000000a5
0x603078:       0x0000003d00000000      0x0000008500000033
0x603088:       0x000000070000002d      0x0000000000000022
0x603098:       0x000000000000003e      0x0000000000000000
0x6030a8:       0x0000000000000000      0x0000002800000000
0x6030b8:       0x0000000000000047      0x0000004200000000
0x6030c8:       0x00000000000000f5      0x0000000000000000
0x6030d8:       0x0000000000000061      0x0000000000445341
0x6030e8:       0x0000000000000000      0x0000000000000000
0x6030f8:       0x0000000000000000      0x0000000000000000
```


其实这里就是 得出的 输入字符 对应 那个文件的 频率　假如 你输入的 T 而 文件出现了133 次 那么就代表了 v5[0]是133

OK 那么我们往下看

```
{
  __int64 result; // rax
  char v2[5]; // [rsp+13h] [rbp-5h]

  v2[4] = 0;
  *v2 = *a1;
  while ( s[*&v2[1]] )
  {
    a1[*&v2[1]] = a1[s[*&v2[1]]];
    *&v2[1] = s[*&v2[1]];
  }                                     // v2[0]=a[0]
                                        // a[index]=a[[s[index]]]
                                        // index=s[index]
                                        // while(s[index]==0)
                                        // a[index]=a[0]
  result = v2[0];
  a1[*&v2[1]] = v2[0];
  return result;
}
```


IDA Vie… | Pseudocod… | Stack of sub_400… | Pseudocod… | Hex Vie… | Structu…

```
_BYTE *__fastcall sub_400DB4(_BYTE *a1, int a2)
{
  _BYTE *result; // rax
  char v3; // [rsp+17h] [rbp-5h]
  int i; // [rsp+18h] [rbp-4h]

  v3 = *a1 >> 5;
  for ( i = 0; a2 - 1 > i; ++i )
    a1[i] = 8 * a1[i] | (a1[i + 1] >> 5);       // for(int i=0;i<len-1;i++)
                                                // {
                                                //   a[i] << 3 | a[i+1] >> 5
                                                // }
  result = &a1[i];
  *result = 8 * *result | v3;
  return result;                                // a[len-1]*8|a[0]>>5
}
```


emmmm 两个加密 我们解密一下就好了

然后这里是我们C语言实现的代码

```
#include <stdio.h>
#include<iostream>
#include<iomanip>
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<vector>
#include<iostream>
#include<map>
#include<time.h>
#include<queue>
#include <Windows.h>
#include "windows.h"
using namespace std;
map<int,int>mapps;
map<int,char>ma;
int vable[25]={' ', '\'', '_', 'a', 'c', 'b', 'e', 'd', 'g', 'f', 'i', 'h', 'k', 'm', 'l', 'o', 'n', 'p', '
int key[25]={71, 40, 245, 104, 15, 30, 169, 29, 38, 19, 60, 67, 20, 28, 39, 165, 118, 26, 51, 61, 45, 133,
int mapp[38]={22, 0, 6, 2, 30, 24, 9, 1, 21, 7, 18, 10, 8, 12, 17, 23, 13, 4, 3, 14, 19, 11, 20, 16, 15, 5,
int l[38]={0xec,0x29,0xe3,0x41,0xe1,0xf7,0xaa,0x1d,0x29,0xed,0x29,0x99,0x39,0xf3,0xb7,0xa9,0xe7,0xac,0x2b,0
int main()
{
    for(int i=0;i<25;i++)
    {
        ma[key[i]]=vable[i];
    }
    for(int i=0;i<38;i++)
    {
        mapps[mapp[i]]=i;
    }
    int s[38];
    int temp=s[37]&0x7;
    for(int i=0;i<38;i++)
    {
        s[i]=(temp<<5)|(l[i]>>3);
        temp=l[i]&0x7;
    }
    int j=37;
    temp=s[j];
    while(mapps[j]!=37)
    {
        s[j]=s[mapps[j]];
        j=mapps[j];
    }
    s[j] = temp;
    for(int i=0;i<38;i++)
    {
        printf("%c",ma[s[i]]);
    }
    return 0;
}
```

然后 官方给的是 python代码 很简洁。

。https://www.xctf.org.cn/library/details/8723e039db0164e2f7345a12d2edd2a5e800adf7/

```
#!/usr/bin/python3
# -*- coding:utf-8 -*-
s = [22, 0, 6, 2, 30, 24, 9, 1, 21, 7, 18, 10, 8, 12, 17, 23, 13, 4, 3, 14, 19, 11, 20, 16, 15, 5, 25, 36,
mapp={' ': 71, "'": 40, '_': 245, 'a': 104, 'c': 15, 'b': 30, 'e': 169, 'd': 29, 'g': 38, 'f': 19, 'i': 60,
def decypt():
    enc = open("out","rb").read()
    d0 = []
    temp = ord(enc[len(enc)-1]) & 0x7
    for i in range(len(enc)):
        d0.append((temp << 5) | (ord(enc[i]) >> 3))
        temp = ord(enc[i]) & 0x7

    i = 37
    temp = d0[37]
    while s.index(i) != 37:
        d0[i] = d0[s.index(i)]
        i = s.index(i)
    d0[i] = temp
    flag = []
    for i in d0:
        flag.append(list(mapp.keys())[list(mapp.values()).index(i)])
    return "QCTF{%s}" % ''.join(flag)
print(decypt())
```