

南邮CTF逆向题第四道WxyVM解题思路

原创

iqiqiya 于 2017-12-24 09:14:27 发布 3693 收藏 1

分类专栏: -----南邮CTF 我的CTF进阶之路 文章标签: 南邮CTF writeup 逆向 WxyVM CTF

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/xiangshangbashaonian/article/details/78883486>

版权



-----南邮CTF 同时被 2 个专栏收录

6 篇文章 0 订阅

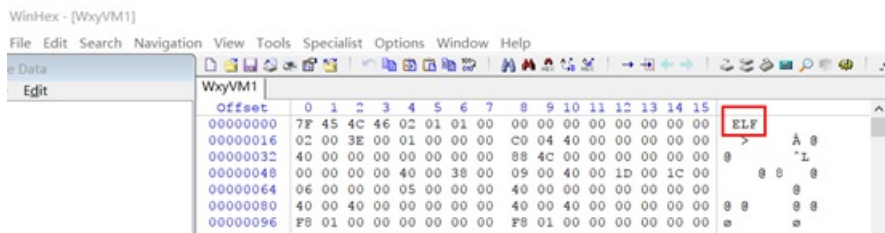
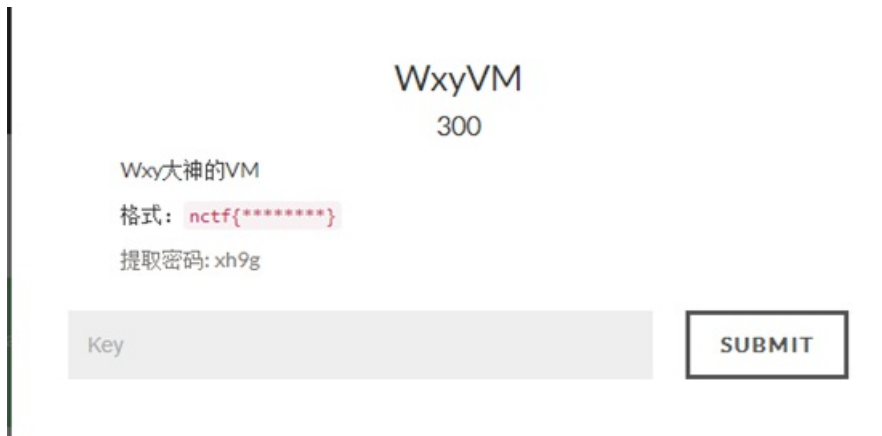
订阅专栏

我的CTF进阶之路

108 篇文章 18 订阅

订阅专栏

首先看下题目



ELF文件

直接载入64位IDA

进入main函数

F5查看C代码

The screenshot shows the IDA Pro interface with the main function selected in the Functions window. The main window displays the assembly code on the left and the pseudocode on the right. The pseudocode is as follows:

```

int64 __fastcall main(int64 a1, char **a2, char **a3)
{
    char v4; // [rsp+Bh] [rbp-5h]
    signed int i; // [rsp+Ch] [rbp-4h]

    puts("[WxyVM 0.0.1]");
    puts("input your flag:");
    scanf("%s", &byte_604B80);
    v4 = 1;
    sub_4005B6();
    if ( strlen(&byte_604B80) != 24 )
        v4 = 0;
    for ( i = 0; i <= 23; ++i )
    {
        if ( *(&byte_604B80 + i) != dword_601060[i] )
            v4 = 0;
    }
    if ( v4 )
        puts("correct");
    else
        puts("wrong");
    return 0LL;
}

```

分析main函数代码:

```

1  int64 __fastcall main(int64 a1, char **a2, char **a3) //main函数有三个参数
2  {
3      char v4; // [rsp+Bh] [rbp-5h] //定义一个char型变量v4
4      signed int i; // [rsp+Ch] [rbp-4h] //int i;
5
6      puts("[WxyVM 0.0.1]"); //输出字符串"[WxyVM 0.0.1]"
7      puts("input your flag:"); //输出字符串"input your flag:"
8      scanf("%s", &byte_604B80); //获取用户输入一个字符串 保存在地址604B80的位置
9      v4 = 1; //v4 = 1;
10     sub_4005B6(); //调用sub_4005B6这个函数
11     if ( strlen(&byte_604B80) != 24 ) //如果获取的字符串长度不等于24
12         v4 = 0; //令v4 = 0;
13     for ( i = 0; i <= 23; ++i ) //开始for循环 当i>23时跳出
14     {
15         if ( *(&byte_604B80 + i) != dword_601060[i] ) //比较404B80加上i的地址处保存的值与601060处的值
16             v4 = 0; //即地址604B80与601060 两个数组相比较 若相等 令v4 = 0;
17     }
18     if ( v4 ) //如果v4等于0 输出字符串"correct"
19         puts("correct");
20     else //否则输出字符串"wrong"
21         puts("wrong");
22     return 0LL;
23 }

```

`__int64 __fastcall main(__int64 a1, char**a2, char**a3)` //main函数有三个参数

{

`char v4;` // [rsp+Bh] [rbp-5h] //定义一个char型变量v4

`signed int i;` // [rsp+Ch] [rbp-4h] //int i;

`puts("[WxyVM 0.0.1]");` //输出字符串 "[WxyVM 0.0.1]"

`puts("input your flag:");` //输出字符串 "input your flag:"

`scanf("%s", &byte_604B80);` //获取用户输入一个字符串 保存在地址604B80的位置

`v4 = 1;` //v4 = 1;

`sub_4005B6();` //调用sub_4005B6这个函数

`if (strlen(&byte_604B80) != 24)` //如果获取的字符串长度不等于24

`v4 = 0;` //令v4 = 0;

```

for( i =0; i <=23; ++i )//开始for循环当i>23时跳出

{

if (*(&byte_604B80 + i) != dword_601060[i])//比较404B80加上i的地址处保存的值与601060处的值

v4 =0;//即地址604B80与601060 两个数组相比较若相等令v4 = 0;

}

if( v4 )//如果v4等于0 输出字符串"correct"

puts("correct");

else//否则输出字符串"wrong"

puts("wrong");

return 0LL;

}

```

接着我们看看函数sub_4005B6:

分析一波

```

1  int64 sub_4005B6() //函数体
2  {
3  unsigned int v0; // ST04_4 //定义无符号整形v0
4  int64 result; // rax //定义64位整数result
5  signed int i; // [rsp+0h] [rbp-10h] //int i;
6  char v3; // [rsp+8h] [rbp-8h] //char v3;
7
8  for ( i = 0; i <= 14999; i += 3 ) //for循环部分
9  {
10 v0 = byte_6010C0[i];
11 v3 = byte_6010C0[i + 2];
12 result = v0;
13 switch ( v0 )
14 {
15 case 1u:
16 result = byte_6010C0[i + 1];
17 *(&byte_604B80 + result) += v3;
18 break;
19 case 2u:
20 result = byte_6010C0[i + 1];
21 *(&byte_604B80 + result) -= v3;
22 break;
23 case 3u:
24 result = byte_6010C0[i + 1];
25 *(&byte_604B80 + result) ^= v3;
26 break;
27 case 4u:
28 result = byte_6010C0[i + 1];
29 *(&byte_604B80 + result) *= v3;
30 break;
31 case 5u:
32 result = byte_6010C0[i + 1];
33 *(&byte_604B80 + result) ^= *(&byte_604B80 + byte_6010C0[i + 2]);
34 break;
35 default:
36 continue;
37 }
38 }
39 return result;
40 }

```

思路一:

该函数对我们主函数中输入的字符串进行修改，根据v0的值来确定进入哪个case，6010c0处的数组应该是每三个为一组，分别为v0,v1,v2，每组中的第一个数字决定跳到哪个操作进行运算，第二个数字决定对用户字符串的第几个字符进行操作，第三个数字决定我们要操作的操作数。

所以我们只要对上面运算进行逆运算即可得到flag

将6010c0处的数组保存下来，查看后发现v0的值只有1, 2, 3，那么我们就不管后三种情况，字符串长度只有24位，我们选择直接爆破，脚本如下所示：

```
b = ''

c = []

d = []

e = []

s

= ['0xc4', '0x34', '0x22', '0xb1', '0xd3', '0x11', '0x97', '0x7', '0xdb', '0x37', '0xc4', '0x6', '0x1d', '0xfc', '0x5b', '0xed']

with open('test.txt', 'r') as f:

while True:

a = f.readline() # 整行读取数据

if not a:

break

b += a[:24] + a[25:]

for i in range(0, len(b), 9):
```

```
c.append(int(b[i:i+2]))
```

```
d.append(int(b[i+3:i+5],16))
```

```
e.append(int(b[i+6:i+8],16))
```

```
def rplus(s,x,y):
```

```
s[x]=(s[x]+y)%2**8
```

```
def rsub(s,x,y):
```

```
s[x]=(s[x]-y)%2**8
```

```
def rxor(s,x,y):
```

```
s[x]=(s[x]^y)%2**8
```

```
def case(s,i,x,y):
```

```
if i ==1:
```

```
rplus(s,x,y)
```

```
elif i ==2:
```

```
rsub(s,x,y)
```

```
elif i ==3:
```

```
rxor(s,x, y)
```

```
count =0
```

```
flag =[0for i in range(24)]
```

```
result =''
```

```
for k in range(24):
```

```
for j in range(33,128):
```

```
flag[k]= j
```

```
for i in range(len(c)):
```

```
case(flag,c[i], d[i], e[i])
```

```
if hex(flag[k])==s[k]:
```

```
result += chr(j)
```

```
print chr(j)
```

```
break
```

```
print result
```

得到nctf{Embr4ce_Vm_j0in_R3}

思路二：

直接对全部进行逆运算

可以参考

http://blog.csdn.net/m0_37812124/article/details/76269151

<http://blog.csdn.net/whklhyyy/article/details/74793530?locationNum=1&fps=1>

ELF文件格式详解

<http://blog.csdn.net/tenfyguo/article/details/5631561>