

南邮CTF逆向题第六道WxyV2解题思路

原创

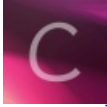
iqiqiya 于 2017-12-24 11:42:29 发布 757 收藏

分类专栏: -----南邮CTF 我的CTF进阶之路 文章标签: 南邮CTF writeup 逆向 WxyV2 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/78884187>

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

如题

WxyVM 2

500

Wxy大神的另一个VM

格式: `nctf{*****}`

提取密码: uef8

Key

SUBMIT

看提示应该与第四道题WxyVM一样是ELF文件 那么我们直接上IDAx64

<http://blog.csdn.net/xiangshangbashaonian/article/details/78883486>

首先找到main函数

还是习惯性查看下可疑字符串

Function name	Seqn	Address	Length	Type	String
._init_proc	.init	LOAD:000...	0000001C	C	/lib64/ld-linux-x86-64.so.2
sub_400460	.plt	LOAD:000...	0000000A	C	libc.so.6
._puts	.plt	LOAD:000...	00000005	C	puts
._strlen	.plt	LOAD:000...	00000007	C	strlen
.__libc_start_main	.plt	LOAD:000...	00000006	C	scanf
._scanf	.plt	LOAD:000...	00000012	C	__libc_start_main
._main_start_	.plt	LOAD:000...	0000000F	C	__main_start_
start	.text	LOAD:000...	0000000C	C	GLIBC_2.2.5
sub_4004F0	.text	.rodata:...	0000000E	C	[WxyVM 0.0.2]
sub_400570	.text	.rodata:...	00000011	C	input your flag
sub_400590	.text	.rodata:...	00000008	C	correct
main	.text	.rodata:...	00000006	C	wrong
init	.text	.eh_fram...	00000006	C	.*3\$\\
fini	.text				
._tera_proc	.fini				
puts	exte:				
strlen	exte:				
__libc_start_main	exte:				
scanf	exte:				

按下F5查看伪代码(中间有大段我省略截图了)

```

1  int64 __fastcall main( int64 a1, char **a2, char **a3)
2  {
3  char v4; // [sp+8h] [bp-5h]@1
4  signed int i; // [sp+Ch] [bp-4h]@3
5
6  puts("[WxyUM 0.0.2]");
7  puts("input your flag:");
8  scanf("%s", &byte_694100);
9  v4 = 1;
10 if ( strlen(&byte_694100) != 25 )
11     v4 = 0;
12 dword_694140 = -1677284711;
13 byte_694117 += 11;
14 byte_694103 += 31;
15 dword_69417C ^= dword_694178;
16 dword_694180 ^= dword_69417C;
17 dword_694184 ^= dword_694180;
18 dword_694188 ^= dword_694184;
19 dword_69418C ^= dword_694188;
20 dword_694190 ^= dword_69418C;
21 dword_694194 ^= dword_694190;
22 dword_694198 ^= dword_694194;
23
25021 dword_6941E0 ^= dword_6941DC;
25022 byte_694102 ^= 0xBu;
25023 byte_694100 ^= 0x5Eu;
25024 byte_694111 += 69;
25025 byte_694109 += 7;
25026 for ( i = 0; i <= 24; ++i )
25027 {
25028     if ( *(&byte_694100 + i) != dword_694060[i] )
25029         v4 = 0;
25030 }
25031 if ( v4 )
25032     puts("correct");
25033 else
25034     puts("wrong");
25035 return 0LL;
25036 }

```

这次就不一行一行分析了 大致意思是:

获取一个字符串存放到地址694100处

限制长度为25

那么如果要对这个字符串进行操作 只会从694100进行相应操作 那么其他的自然就是混淆我们 又可以清楚的看到 有效的操作都是以byte型进行 那么dword的我们直接过滤掉即可

1	过滤前:	1	过滤后:
2	dword_694140 = -1677284711;	2	17 += 11
3		3	
4	byte_694117 += 11;	4	03 += 31
5		5	
6	byte_694103 += 31;	6	05 += 78
7		7	
8	dword_69417C ^= dword_694178;	8	17 += 109
9		9	
10	dword_694180 ^= dword_69417C;	10	02 -= 25
11		11	
12	dword_694184 ^= dword_694180;	12	01 -= 48
13		13	
14	dword_694188 ^= dword_694184;	14	0B += 123
15		15	
16	dword_69418C ^= dword_694188;	16	0D ^= 0x26
17		17	
18	dword_694190 ^= dword_69418C;	18	17 ^= 0x59
19		19	
20	dword_694194 ^= dword_694190;	20	0C += 3
21		21	
22	dword_6941B4 ^= dword_6941B0;	22	0B -= 5
23		23	
24	dword_6941B8 ^= dword_6941B4;	24	0D += 75
25		25	
26	dword_6941BC ^= dword_6941B8;	26	04 ^= 0x48
27		27	
28	dword_6941C0 ^= dword_6941BC;	28	05 -= 88

a

```
= [0xffffffffc0, 0xffffffff85, 0xfffffffff9, 0x6c, 0xfffffbb2, 0x14, 0xfffffbb, 0xffffffe4, 0xd, 0x59, 0x1c, 0x23, 0xfffff88, 0x
```

f = open('key.txt') #过滤后的txt文本自增自减的直接格式改下

```
for b in f.readlines()[::-1]:
```

```
if b[3]=='+':
```

```
if 'x' in b:
```

```
a[int(b[:2],16)]-= int(b[8:],16 if 'x' in b else 10)
```

```
else:
```

```
a[int(b[:2],16)]-= int(b[6:],10)
```

```
if b[3]=='-':
```

```
if 'x' in b:
```

```
a[int(b[:2],16)]+= int(b[8:],16)
```

```
else:
```

```
a[int(b[:2],16)]+= int(b[6:],10)
```

```
if b[3]=='^':
```

```
if 'x' in b:
```

```
a[int(b[:2],16)]^= int(b[8:],16)
```

```
else:
```

```
a[int(b[:2],16)]^= int(b[6:],10)
```

```
flag = ""
```

```
for i in a:
```

```
flag += chr(i%256)
```

```
print flag
```

```
flag: nctf{th3_vM_w1th0ut_dAta}
```