

百度杯二月Reverse场CrackMe11writeup

原创

Flying_Fatty 于 2017-02-25 09:21:46 发布 866 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/57073061>

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

这个题是被算法迷惑到的一个题目

首先在IDA中可以看到这两个重要条件:

```
if ( v3 > 20 || 2 * v3 != v4 )
return 5173;

v6 = *((_BYTE *)&v67 + v5);
if ( (v6 < 97 || v6 > 122) && (v6 < 65 || v6 > 90) )
return 5173;
```

看到这两个很明显:

A: 输入的长度要是36 (在OD中能够发现一个定值字符串, 长度v3 == 18, 那么v4 == 36)

B: 输入的字符范围在A-F, 0-9共16个字符中

然后就被各种异或算法迷惑, 在OD写注释写成了这样:

004015C4	- 02D2	add dl,dl	
004015C6	- 0FB6C2	movzx eax,dl	
004015C9	- 99	cdq	
004015CA	- F7FF	idiv edi	mod 13
004015CC	- FEC2	inc dl	plus 1
004015CE	- 32D1	xor dl,cl	xor v56
004015D0	- 325424 0F	xor dl,byte ptr ss:[esp+0xF]	xor v49
004015D4	- 80C2 02	add dl,0x2	
004015D7	- 325424 0E	xor dl,byte ptr ss:[esp+0xE]	xor v56
004015D8	- 325424 0C	xor dl,byte ptr ss:[esp+0xC]	xor v49
004015DF	- FEC2	inc dl	
004015E1	- 32FA	xor bh,dl	xor v16
004015E3	- 8A5424 0D	mov dl,byte ptr ss:[esp+0xD]	
004015E7	- 80C7 02	add bh,0x2	
004015EA	- 32FD	xor bh,ch	xor v57
004015EC	- 32F9	xor bh,cl	xor v56
004015EE	- FEC7	inc bh	
004015F0	- 4E	dec esi	esi == v20
004015F1	- 0F85 F9FEFF	jmp 004014F0	while(v20)

但是这样是很难逆向算法的, 原因很简单:

A: 就算你看懂了算法, 也得一个一个的重新计算

B: 因为不知道输入, 所以从输出套出输入还是需要暴力去做

那么我们可以试着猜想，这个题的本身思路就是暴力做（不然为什么要告诉你长度36，每个字符的范围呢）

分析处理过程，我们知道，18个字符处理成了18个结果

我们输入的36位字符，处理成了18个结果，这两个运算过程相同，而且独立，所以不会相互影响

也就是说：我们不需要知道运算的过程是什么，因为输入和最后比对的值无关，那么：我们可以把00 -- 0xFF的所有结果全部生成，然后加以比对

暴力出结果即可

下面是构造过程：

在OD中打开，中断到比较的地方，如图：

```
004018A0 > 8B840C AC000 mov eax,dword ptr ss:[esp+ecx+0xAC]
004018A7 - 3B840C D4000 cmp eax,dword ptr ss:[esp+ecx+0xD4]
004018AE ~ 75 22      jnz short 004018D2
004018B0 - 83C1 04    add ecx,0x4
004018B3 - 83F9 28    cmp ecx,0x28
004018B6 ^ 7C E8      j short 004018A0
004018B8 - B8 EF020000 mov eax,0x2EF
004018BD - 5F        pop edi
004018BE - 5E        pop esi
004018BF - 5B        pop ebx
004018C0 - 8B8C24 F0000 mov ecx,dword ptr ss:[esp+0xF0]

堆栈 ss:[0018F8E4]=3064813C
eax=FFFFFFEE
跳转来自 004018B6
```

在4018A0处下断点，然后构造我们的数据放进去，然后可以得到这样的一个截图：

0018F90C	2F86FDBC	0018F90C	6DEC0B82	0018F90C	7BB28928
0018F910	5B926908	0018F910	19F8B72E	0018F910	A7DE5534
0018F914	073E3514	0018F914	258443BA	0018F914	F36AA100
0018F918	534A0160	0018F918	B1D0CFA6	0018F918	3FD60D0C
0018F91C	00009F36	0018F91C	00001DDC	0018F91C	0000AB22
0018F90C	B918574E	0018F90C	477E7554	0018F90C	65C483FA
0018F910	45A463DA	0018F910	938A41A0	0018F910	F1100FE6
0018F914	51706FC6	0018F914	DF76AD2C	0018F914	5D1CB8F2
0018F918	3DFC9BD2	0018F918	4BC25938	0018F918	C968E71E
0018F91C	0000A948	0018F91C	0000F76E	0018F91C	00009574
0018F90C	33AAE140	0018F90C	91B0AF06	0018F90C	1FB6ED6C
0018F910	7F164D4C	0018F910	7D3CDB12	0018F910	8B029978
0018F914	EB62F958	0018F914	E98887BE	0018F914	37AEA504
0018F918	978E85E4	0018F918	B594D3CA	0018F918	C33A3150
0018F91C	0000A31A	0018F91C	000081E0	0018F91C	00004F26
0018F90C	9D5CFB32	0018F90C	2BA23998	0018F90C	29C8C7FE
0018F910	09A8275E	0018F910	D7CEC524	0018F910	F5D4130A
0018F914	D5B473EA	0018F914	E35AD1F0	0018F914	C1205FF6
0018F918	2180BF56	0018F918	EF46BD7C	0018F918	2DACCB42
0018F91C	00008D8C	0018F91C	00001B52	0018F91C	0000D9B8
0018F90C	77EEE544	0018F90C	15F4B32A	0018F90C	239A1130
0018F910	037A7190	0018F910	61C0FF96	0018F910	2F2F2F2F
0018F914	8F66DD9C	0018F914	CDCC6BE2	0018F914	2F2F2F2F
0018F918	3B7249E8	0018F918	79D8170E	0018F918	2F2F2F2F
0018F91C	0000679E	0018F91C	00000564	0018F91C	00002F2F

这就是从00到0xFF的所有数据，那么我们写个程序，然后反向暴力打表就能够出结果了，直接贴我的py代码了

```
s = 'BinGzLFormiChunQiu'
```

```

number = ''
for i in s:
    number += str(hex(ord(i)))[2:]
#print number

s = '7569516E756843696D726F464C7A476E6942'
s = s.decode('hex')[::-1]
#print s,len(s)

for i in range(0,0xfc,18):
    data = ''
    for j in range(0,18):
        if i + j < 0x10:
            data += '0'
            data += str(hex(i+j))[2:]
        else:
            data += str(hex(i+j))[2:]
    data = data.upper()
#print data[0:8],data[8:16],data[16:24],data[24:32],data[32:36]
ans = data[6:8] + data[4:6] + data[2:4] + data[0:2]
ans += data[14:16] + data[12:14] + data[10:12] + data[8:10]
ans += data[22:24] + data[20:22] + data[18:20] + data[16:18]
ans += data[30:32] + data[28:30] + data[26:28] + data[24:26]
ans += data[34:36] + data[32:34]
#print ans

ans = 'FFFEFDFC' + '00' * 14
#print ans

ans = [0x2f,0x86,0xfd,0xbc,0x5b,0x92,0x69,0x08,0x07,0x3e,0x35,0x14,0x53,0x4a,0x01,0x60,0x9f,0x36,
0x6d,0xec,0xb,0x82,0x19,0xf8,0xb7,0x2e,0x25,0x84,0x43,0xba,0xb1,0xd0,0xcf,0xa6,0x1d,0xdc,
0x7b,0xb2,0x89,0x28,0xa7,0xde,0x55,0x34,0xf3,0x6a,0xa1,0x00,0x3f,0xd6,0xd,0xc,0xab,0x22,
0xb9,0x18,0x57,0x4e,0x45,0xa4,0x63,0xda,0x51,0x70,0x6f,0xc6,0x3d,0xfc,0x9b,0xd2,0xa9,0x48,
0x47,0x7e,0x75,0x54,0x93,0x8a,0x41,0xa0,0xdf,0x76,0xad,0x2c,0x4b,0xc2,0x59,0x38,0xf7,0x6e,
0x65,0xc4,0x83,0xfa,0xf1,0x10,0xf,0xe6,0x5d,0x1c,0xbb,0xf2,0xc9,0x68,0xe7,0x1e,0x95,0x74,
0x33,0xaa,0xe1,0x40,0x7f,0x16,0x4d,0x4c,0xeb,0x62,0xf9,0x58,0x97,0x8e,0x85,0xe4,0xa3,0x1a,
0x91,0xb0,0xaf,0x06,0x7d,0x3c,0xdb,0x12,0xe9,0x88,0x87,0xbe,0xb5,0x94,0xd3,0xca,0x81,0xe0,
0x1f,0xb6,0xed,0x6c,0x8b,0x02,0x99,0x78,0x37,0xae,0xa5,0x04,0xc3,0x3a,0x31,0x50,0x4f,0x26,
0x9d,0x5c,0xfb,0x32,0x09,0xa8,0x27,0x5e,0xd5,0xb4,0x73,0xea,0x21,0x80,0xbf,0x56,0x8d,0x8c,
0x2b,0xa2,0x39,0x98,0xd7,0xce,0xc5,0x24,0xe3,0x5a,0xd1,0xf0,0xef,0x46,0xbd,0x7c,0x1b,0x52,
0x29,0xc8,0xc7,0xfe,0xf5,0xd4,0x13,0x0a,0xc1,0x20,0x5f,0xf6,0x2d,0xac,0xcb,0x42,0xd9,0xb8,
0x77,0xee,0xe5,0x44,0x03,0x7a,0x71,0x90,0x8f,0x66,0xdd,0x9c,0x3b,0x72,0x49,0xe8,0x67,0x9e,
0x15,0xf4,0xb3,0x2a,0x61,0xc0,0xff,0x96,0xcd,0xcc,0x6b,0xe2,0x79,0xd8,0x17,0xe,0x05,0x64,
0x23,0x9a,0x11,0x30]

def enumerate(num):
    global flag
    for i in range(0,0x100):
        if ans[i] == num:
            print hex(i)
            flag += str(hex(i))[2:]

flag = ''
enumerate(0x3c)
enumerate(0x81)
enumerate(0x64)
enumerate(0x30)

enumerate(0xe8)

```

```
enumerate(0xee)
enumerate(0x0a)
enumerate(0x90)

enumerate(0x20)
enumerate(0x1b)
enumerate(0x46)
enumerate(0x52)

enumerate(0xc8)
enumerate(0x20)
enumerate(0xfe)
enumerate(0xd4)

enumerate(0x8c)
enumerate(0xfe)
print flag.upper()
```