Securinets Sunshine encrypt 三场CTF逆向writeup

apeng / 2019-04-08 07:27:00 / 浏览数 2752 安全技术 CTF 顶(0) 踩(0)

近期三场ctf，题目都比较简单。

Securinets Prequals 2K19

AutomateMe

获取输入后比对长度3296，然后是3296个if一个一个的比对，一共有下面两种比对方式：

```
.text:0000000000000774 ; ---------------------------------------------------------
.text:0000000000000774
.text:0000000000000774 loc_774:                                ; CODE XREF: main+82↑j
.text:0000000000000774                 mov     rax, [rbp+var_20]
.text:0000000000000778                 add     rax, 8
.text:000000000000077C                 mov     rax, [rax]
.text:000000000000077F                 add     rax, 1
.text:0000000000000783                 movzx   eax, byte ptr [rax]
.text:0000000000000786                 cmp     al, 68h ; 'h'
.text:0000000000000788                 jz      short loc_7A0
.text:000000000000078A                 lea     rdi, aNope      ; "nope :( "
.text:0000000000000791                 mov     eax, 0
.text:0000000000000796                 call    _printf
.text:000000000000079B                 jmp     locret_283A0
.text:00000000000007A0 ; ---------------------------------------------------------
.text:00000000000007A0
.text:00000000000007A0 loc_7A0:                                ; CODE XREF: main+AE↑j
.text:00000000000007A0                 mov     rax, [rbp+var_20]
.text:00000000000007A4                 add     rax, 8
.text:00000000000007A8                 mov     rax, [rax]
.text:00000000000007AB                 movzx   eax, byte ptr [rax+2]
.text:00000000000007AF                 mov     [rbp+var_1], al
.text:00000000000007B2                 xor     [rbp+var_1], 0EBh
.text:00000000000007B6                 cmp     [rbp+var_1], 8Eh
.text:00000000000007BA                 jz      short loc_7D2
.text:00000000000007BC                 lea     rdi, aNope      ; "nope :( "
.text:00000000000007C3                 mov     eax, 0
.text:00000000000007C8                 call    _printf
.text:00000000000007CD                 jmp     locret_283A0
.text:00000000000007D2 ; ---------------------------------------------------------
```

由于代码都是相似的，比对的过程也是从头到尾，所以直接找到特定的代码提取出数据就行了

```
f = open('bin','rb')
f.read(0x74C)
g = open('flag.txt','ab')

with open('bin','rb') as f:
   f.read(0x74C)
   while(True):
       temp = f.read(1)
       if temp =='':
           break
       temp = ord(temp)
       if temp == 0x3C:
           t = f.read(1)
           k = f.read(1)
           if k=='\x74':
               g.write(t)
       elif temp ==0x80:
           if ord(f.read(1))==0x75:
               f.read(1)
               t = ord(f.read(1))
               f.read(1)
```

```
            f.read(1)
            f.read(1)
            k = ord(f.read(1))
            g.write(chr(k^t))
```

flag就在输出的中间一部分。

warmup

先将输入base64编码，然后在25个check里比对某一位的字符或某两位的相对位置。细心一点把25个check求一遍就行了。

```
table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
k = [0 for i in range(36)]
k[0] = table[28]
k[1] = table[54]
k[2] = k[10] = table[((28+54)>>2) + 1]
k[3] = 'j'
k[4] = chr(ord(k[0])+1)
k[12] = k[22] = k[24] = chr(ord(k[4])-1)
k[23] = k[11] = chr(48)
k[35] = chr(ord(k[11])+9)
k[6] = chr(ord(k[3]) - 32)
k[8] = chr(ord(k[0]) - 1)
k[27] = k[31] = chr(ord(k[4]) + 2)
k[25] = k[9] = chr(ord(k[27]) + 7)
k[13] = k[17] = k[21] = chr(ord(k[1]) + 1)
k[7] = 'p'
k[15] = chr(ord(k[7]) + 3)
k[14] = chr(ord(k[15]) + 1)
k[19] = 'z'
k[34] = chr(ord(k[0]) - 33)
k[5] = k[20] = k[29] = k[33] = 'X'
k[26] = chr(49)
k[16] = k[28] = chr(ord(k[9]) - 32)
k[1] = chr(50)
k[18] = k[30] = chr(ord(k[7]) - 30)
k[32] = k[4]
t = ''
for i in range(36):
   t+=k[i]

print(t.decode('base64'))
```

25个check比较繁琐，需要细心一点。话说应该可以用angr，但是翻了好多文档试了好多脚本都求不出，有时间再仔细学学angr

Matrix of Hell

加密逻辑：

```
gets(s);
if ( strlen(s) != 14 || (sub_5593FC1E383A(), !v3) )
{
 printf("ACCESS DENIED");
 exit(0);
}
v16 = 0;
for ( k = 0; k < strlen(s); ++k )
{
 for ( l = 0; l <= 4; ++l )
 {
   for ( m = 0; m <= 4; ++m )
   {
     if ( matrix[m + 6LL * l] == s[k] )
     {
       s1[v16] = l + 65;
       v4 = v16 + 1;
       s1[v4] = m + 49;
       v16 = v4 + 1;
     }
```

```
   }
 }
}
for ( n = 0; n < strlen(s1); ++n )
 s2[n] = n % 4 ^ s1[n];
if ( strcmp(s3, s2) )
{
 printf("ACCESS DENIED", s2);
 exit(0);
}
```

先生成一个5*6的固定的矩阵，然后获取输入，输入长度需为14

从输入的第一位开始，在矩阵中找到对应的元素，行数 l+65 放到 字符串s1的2*i位置，列数
m+49放到字符串s2的2*i+1位置，最后将字符串的每一位异或位数(n%4)得到s2，再和常量字符串s3对比。

解密脚本：

```
t1 = [0x41, 0x42, 0x43, 0x44, 0x45, 0x0, 0x46, 0x47, 0x48, 0x49, 0x4B, 0x0, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x0, 0x51, 0x52, 0x5
s = 'B0C2A2C6A3A7C5@6B5F0A4G2B5A2'
s1 = []
for i in range(len(s)):
   s1.append(ord(s[i])^(i%4))
pswd = ''
for i in range(14):
   l = s1[2*i] - 65
   m = s1[2*i+1] - 49
   pswd += chr(t1[m + 6*l])
print(pswd)
```

Vectors

输入一个64位的十六进制数

接下来对数据的操作用到了vector，静态分析比较蛋疼，直接动调。先将输入的数据分成8个字节放入一个vector，在sub_F37可以看到它将另7个不同的8位16进制数放入另

```
[heap]:0000564A82C794B0 dword_564A82C794B0 dd 0Ch
[heap]:0000564A82C794B0 dd 0Dh
[heap]:0000564A82C794B0 dd 0ADh
[heap]:0000564A82C794B0 dd 0BEh
[heap]:0000564A82C794B0 dd 0DEh
[heap]:0000564A82C794B0 dd 0EDh
[heap]:0000564A82C794B0 dd 0EFh

[heap]:0000564A82C794E0 dword_564A82C794E0 dd 11h
[heap]:0000564A82C794E0 dd 22h
[heap]:0000564A82C794E0 dd 33h
[heap]:0000564A82C794E0 dd 44h
[heap]:0000564A82C794E0 dd 55h
[heap]:0000564A82C794E0 dd 66h
[heap]:0000564A82C794E0 dd 77h
[heap]:0000564A82C794E0 dd 88h
```

之后就直接比对结果了。直接输入0C0DADBEDEEDEF并不能的到flag，虽然显示通过了check但是输出的flag并不对：

```
Welcome resreveR!...

PASSCODE:0C0DADBEDEEDEF
GOOD JOB U GOT THIS, HERE IS UR FLAG:s6■■2ine'
```

因为一共有7!种排列方式。用itertools爆破求出flag即可

```
from pwn import *
from itertools import *
context.log_level = 'error'
a = [0xad,0xef,0xbe,0xde,0xc,0xed,0xd]
b = permutations(a,7)
for i in b:
   p = process('./bin')
   payload = ''
```

```
    for j in range(7):
        k = hex(i[j])[2:]
        if(len(k)==1):
            k = '0'+k
        payload+=k
    p.recv()
    p.sendline(payload)
    p.recvuntil('FLAG:')
    a = p.recv()
    if a.startswith('securinets'):
        print(a)
        exit()
    p.close()
```

RBOOM!

上来有个ptrace的反调试，直接jmp掉

读取输入后，写入文件"lla"。主要的加密逻辑在sub_93A里。从文件"lla"和文件"la"读取数据后，在sub_CCF加密。

```
v11 = 0;
 v13 = 0;
 for ( i = 0; i <= 255; ++i )
 {
   v18[i] = i;
   v17[i] = la[i % a4];
 }
 for ( j = 0; j <= 255; ++j )
 {
   v5 = (unsigned int)(((unsigned __int8)v18[j] + v11 + (unsigned __int8)v17[j]) >> 31) >> 24;
   v11 = (unsigned __int8)(v5 + v18[j] + v11 + v17[j]) - v5;
   v6 = v18[v11];
   v18[v11] = v18[j];
   v18[j] = v6;
 }
 v12 = 0;
 for ( k = 0; k < a2; ++k )
 {
   v13 = (unsigned __int8)((char *)&off_2F98
                         + 0xFFFFD069
                         + v13
                         + ((unsigned int)((signed int)((signed int)&off_2F98 + 0xFFFFD069 + v13) >> 31) >> 24))
       - ((unsigned int)((signed int)((signed int)&off_2F98 + 0xFFFFD069 + v13) >> 31) >> 24);
   v7 = (unsigned int)((v12 + (unsigned __int8)v18[v13]) >> 31) >> 24;
   v12 = (unsigned __int8)(v7 + v12 + v18[v13]) - v7;
   v8 = v18[v12];
   v18[v12] = v18[v13];
   v18[v13] = v8;
   a5[k] = v18[(unsigned __int8)(v18[v13] + v18[v12])] ^ input[k];
 }
```

看到有两个长度为256的循环，猜测一下是RC4。不太想分析太多，直接在input异或的地方00000F5F下断点，编辑断点添加condition：print(GetRegValue('ecx'))，这样

当然也可以直接用RC4解密，看个人喜好

```
a = [219, 87, 247, 80, 74, 188, 141, 29, 127, 165, 123, 43, 219, 11, 64, 236, 244, 233, 240, 132, 136, 239, 180, 2, 232, 137,
with open('ll','rb') as f:
    s = ''
    for i in range(33):
        s+=chr(a[i]^ord(f.read(1)))
    print(s)
```

monster

这题算法十分简单，但如果不熟悉大整数运算算法的话看起来会比较吃力

首先还是接受16进制的输入，主要加密逻辑在00000B52里面

```
void __cdecl encode(char *s, __int64 a3)//s:"tsebehtsignisrever" a3:input
{
```

```
 unsigned __int64 a1; // [esp+8h] [ebp-20h]
 size_t i; // [esp+18h] [ebp-10h]

 a1 = __PAIR__(HIDWORD(a3), (unsigned int)a3);
 for ( i = 0; strlen(s) > i; ++i )
 {
   temp[i] = a1 ^ s[i];
   a1 = mod(1337 * a1, 133713371337LL);
 }
}
```

当然原本是没有符号表的，这里是我改的。

这里的mod函数就是取模运算，不过是对64位大整数的取模运算，这是一个32位程序，所以要对64位数取模要用两个寄存器/Dword表示，会麻烦一些。一开始对着这个函
=

直接用z3约束求解器解吧

```
from z3 import *
from pwn import *
context.log_level = 'debug'
p = process('./rev',)
# p = remote('54.87.182.197','1337')
a= BitVec('a',64)
s = Solver()
const = [0xCA, 0x3D, 0x3B, 0x5B, 0x4C, 0x9D, 0xD2, 0xCB, 0xDD, 0x17, 0x8D, 0xDC, 0xB9, 0x49, 0x3B, 0xEA, 0x12, 0x25]
key = [  0x74, 0x73, 0x65, 0x62, 0x65, 0x68, 0x74, 0x73, 0x69, 0x67, 0x6E, 0x69, 0x73, 0x72, 0x65, 0x76, 0x65, 0x72]

for i in range(len(key)):
   const[i]^=key[i]
   s.add(a &0xff == const[i])
   a = (a*1337)%133713371337
if s.check() == sat:
   m = s.model()
   payload = hex(m[m[0]].as_long())
   print(payload)
# payload = '564e9367e6e30be'
p.sendline(payload)
print(p.recv())
```

比赛的服务器已经关闭了，本地肯定能打

Sunshine CTF 2019

Patches' Punches

打开程序后直接结束了，看一下汇编有一个永真的语句，把00000540patch成与1比较，直接得到flag

```
.text:0000052B                 push    ecx
.text:0000052C                 sub     esp, 10h
.text:0000052F                 call    __x86_get_pc_thunk_ax
.text:00000534                 add     eax, 1AA4h
.text:00000539                 mov     [ebp+var_10], 1
.text:00000540                 cmp     byte ptr [ebp+var_10], 1
.text:00000544                 jnz     short lose
.text:00000546                 mov     [ebp+var_C], 0
.text:0000054D                 jmp     short win
```

Smash

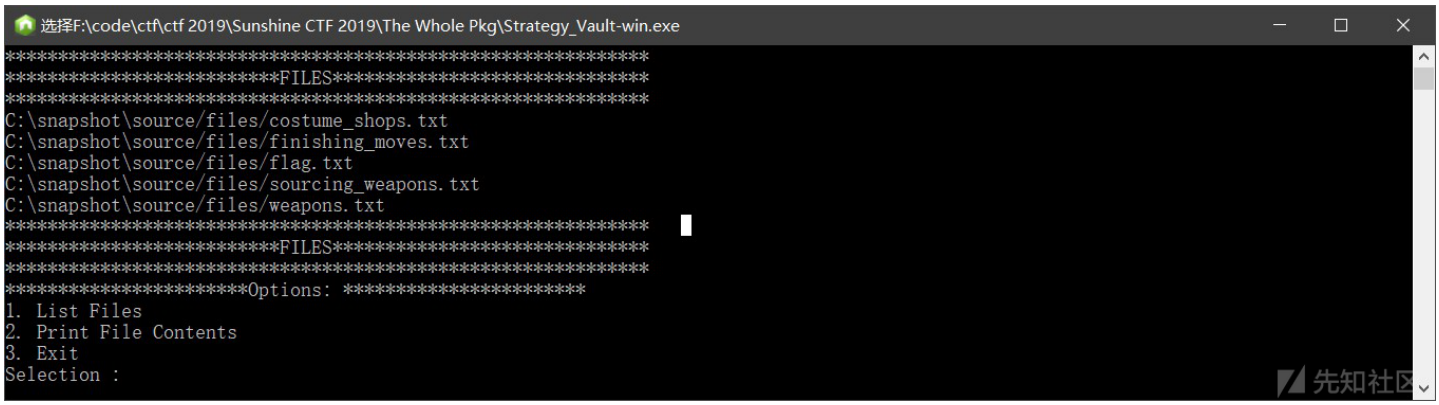
加密逻辑在checkAccessCode函数里，把输入右移一定大小，然后跟常量对比。

```
a = [0x0E60, 0x3A8, 0x1B80, 0x0F60, 0x120, 0x0EA0, 0x188, 0x358, 0x1A0, 0x9A0, 0x184, 0x4E0, 0x0C40, 0x0C20, 0x5A0, 0x1C8, 0x1
b = [5, 3, 6, 5, 2, 5, 3, 3, 3, 5, 2, 4, 6, 5, 5, 2, 2, 5, 2, 6, 5, 1, 3, 4, 5, 3, 4, 6, 6, 5]
s = ''
for i in range(len(a)):
   a[i] >>= b[i]
   s+=chr(a[i])
print(s)
```

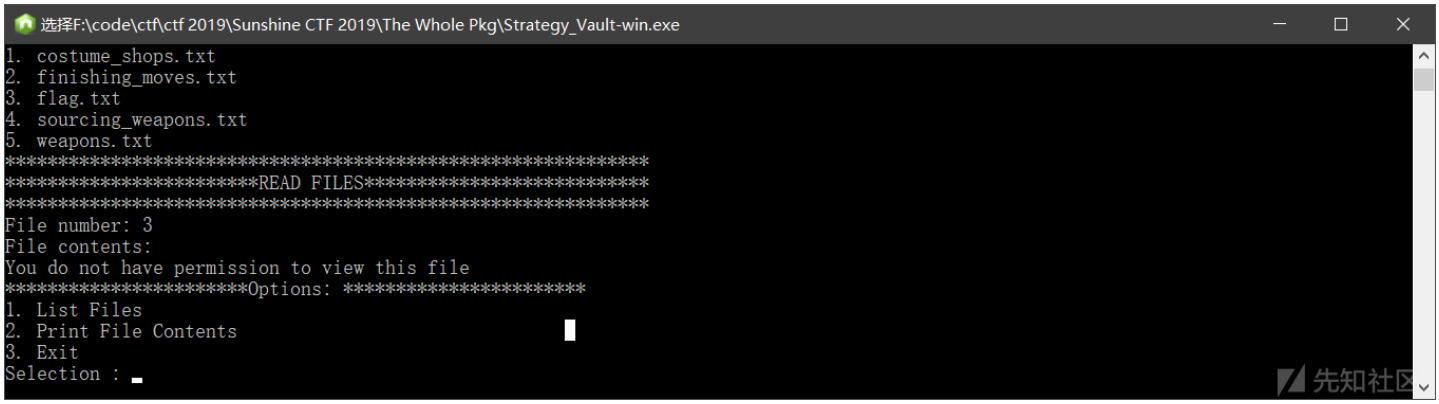
The Whole Pkg

这题就有点意思了，直接打开是个文件读取系统

输入1看文件目录，输入2打印文件



输入3提示没有权限

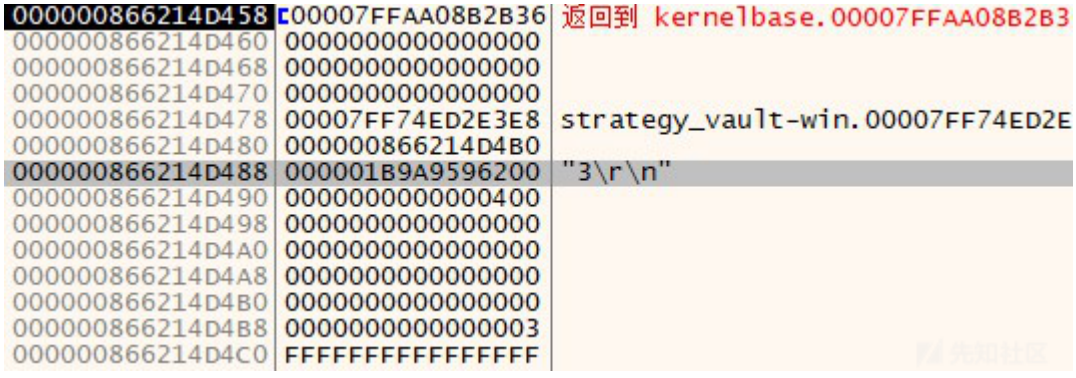


其他的文件都能正常读

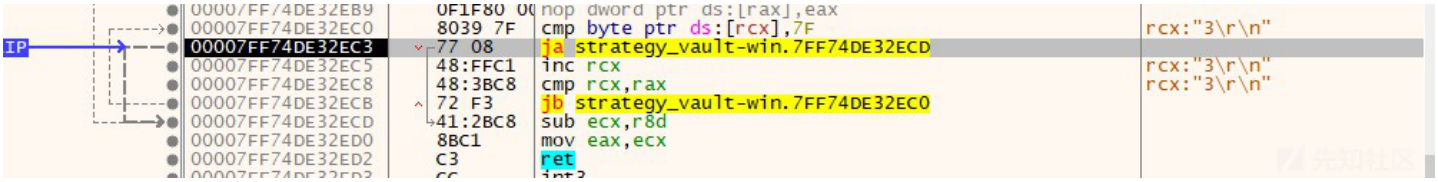
思考一下，可能输3之后会有个函数check，想办法定位到那个check，如果它是采用返回0/1的方式check，可以尝试修改返回值（eax）

用ida打开直接凉凉，里面一大堆库函数乱起八糟挺难分析的

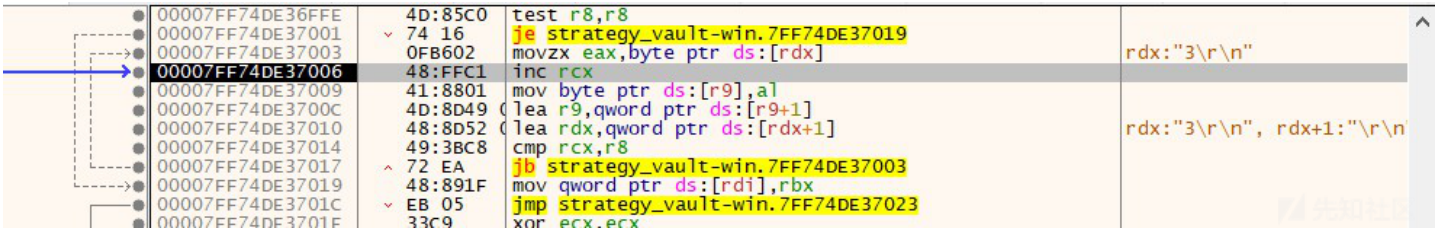
用x64dbg动调，直接运行，先输入一个2，接下来先暂停再输入3回车，这样会暂停到输完3那一步，可以看到我们的输入"3\r\n"在栈里面。



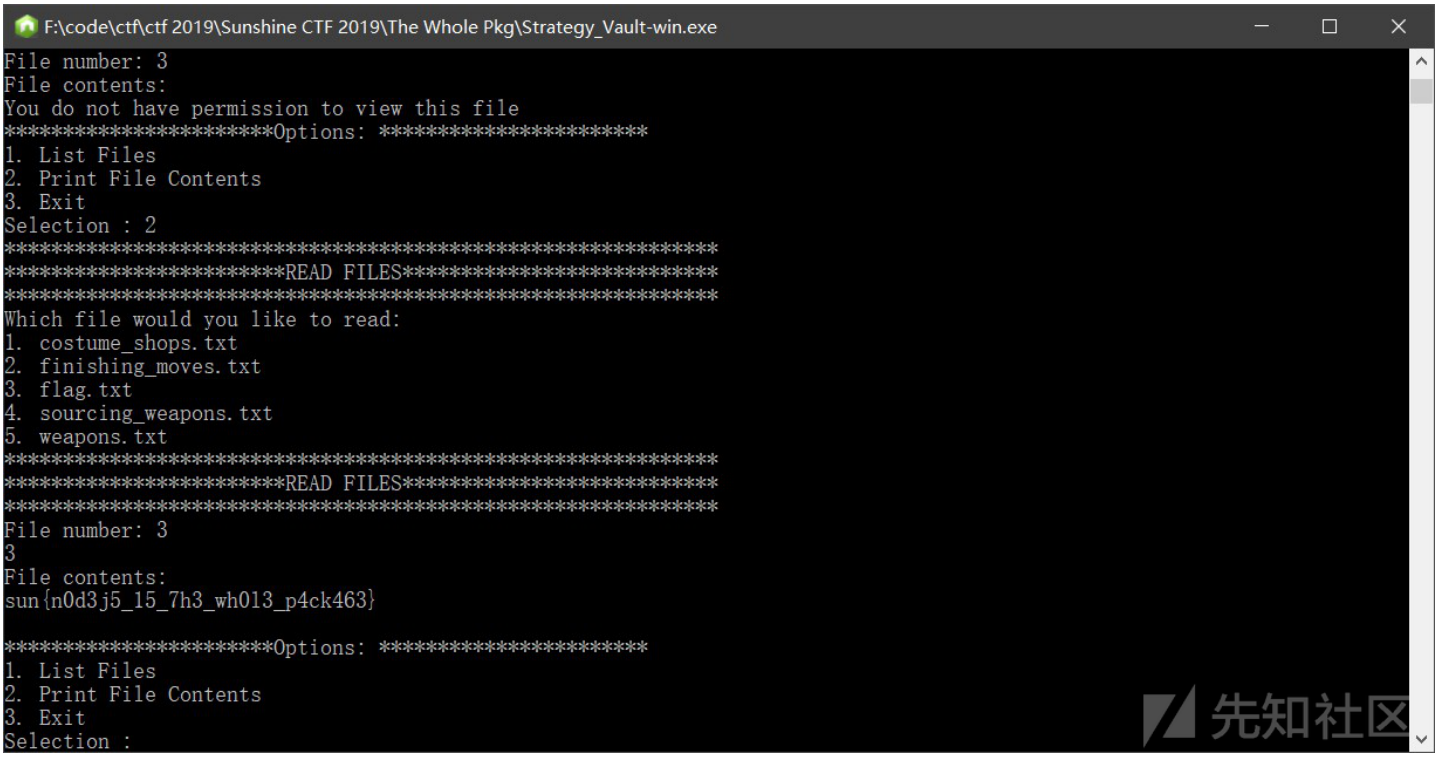
在字符'3'下硬件断点，继续运行，会停到一个函数内，看起来像是strlen，返回了函数长度3。



继续运行，断到另一个函数，看起来像是strcpy，在copy的新地址下硬件断点。



继续运行，来到一个类似check的地方，直接运行到返回，返回值是1，尝试改成0（返回后也能看到它和1，-1，0比较），把之前的断点删除继续与运行，这时候又来到一个

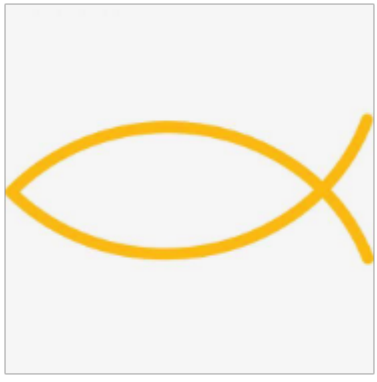


做法很玄学，希望大佬们能讲解下具体细节= =

点击收藏 | 0 关注 | 1

上一篇：初探php扩展之MAC下环境配置篇 下一篇：3种XXE不同攻击方式

1. 1 条回复



apeng 2019-04-08 13:50:42

还有一篇忘记复制上去了。。。

encrypt CTF

Creckme01

输入之后直接对比，有些是不可见字符，用pwntools发送或者直接把flag拔下来。。。

```
from pwn import *
context.log_level = 'debug'
p = process('./crackme01')
p.sendline('1D\x01A  !!e\x19\t')
print(p.recv())
```

Creckme02

输入username和password后，进行一波加密得到一个int，然后用这个int异或一个byte常量的每一位。知道flag第一个字符是'e'，求出要异或的数值即可

```
a =[0x08, 0x03, 0x0E, 0x1F, 0x14, 0x1D, 0x19, 0x2E, 0x39, 0x2B,
 0x16, 0x2C, 0x01, 0x0A, 0x02, 0x1F, 0x04, 0x19, 0x05, 0x00,
 0x1E, 0x40, 0x03, 0x02, 0x19, 0x40, 0x08, 0x0C, 0x1E, 0x14,
 0x10]
b = ord('e')^a[0]
s = ''
for i in range(31):
   s+=chr(a[i]^b)
print(s)
```

crackme03

分五次对五个不同的输入check，前四次都正常，都是与常量比较。到第五次的时候出了点小问题。第五个check是这样的：

```
void __cdecl sub_565CD410(const char *a1)
{
 char dest[9]; // [esp+12h] [ebp-16h]
 unsigned int v2; // [esp+1Ch] [ebp-Ch]

 v2 = __readgsdword(0x14u);
 strncpy(dest, a1, 10u);
 puts("Validating Input 4");
 if ( dest[0] + dest[8] != 0xD5 )
   sub_565CD258();
 if ( dest[1] + dest[7] != 0xCE )
   sub_565CD258();
 if ( dest[2] + dest[6] != 0xE7 )
   sub_565CD258();
 if ( dest[3] + dest[5] != 0xC9 )
   sub_565CD258();
 if ( dest[4] == 0x69 )
   puts("you earned it");
 if ( __readgsdword(0x14u) != v2 )
   sub_565CD670();
}
```

然而我另前4位位0后四位是对应的数据后，迟迟不能通过。最后动调一下才发现问题所在

```
.text:565CD459                movzx   eax, [ebp+dest]
.text:565CD45D                movsx   edx, al
.text:565CD460                movzx   eax, [ebp+dest+8]
.text:565CD464                movsx   eax, al
.text:565CD467                add     eax, edx
.text:565CD469                cmp     eax, 0D5h
```

这里用了movsz有符号扩展，导致0xD5被扩展成了0xFFFFFFD5，再与0xD5比较时就出了问题，其实就是有符号数和无符号数之间的区别。其实如果光看C代码并细心一

```
unsigned __int8 dest[9]; // [esp+12h] [ebp-16h]
 unsigned int v2; // [esp+1Ch] [ebp-Ch]

 v2 = __readgsdword(0x14u);
 strncpy((char *)dest, (const char *)a1, 10u);
 puts("Validating Input 4");
 if ( (char)dest[0] + (char)dest[8] != 0xD5 )
   sub_565CD258();
```

脚本：

```
from pwn import *
context.log_level = 'error'
payload = ["CRACKME02","\xef\xbe\xad\xde","ZXytUb9fl78evgJy3KJN","1","\x75\x7e\x77\x79\x69\x50\x70\x50\x60"]
p = process('./crackme03')
p = remote("104.154.106.182","7777")
for i in range(5):
   print(p.recv())
   p.sendline(payload[i])
print(p.recvall())
```

dontlook at this Challenge

开始有个ptrace，和signal(5,func)，把他们patch掉。

看题目描述以为挺难的，然而在最后比对字符串的地方设个断点运行后，直接看到了flag。。。

直接分析加密算法也不难，前面一大堆操作只是为了生成一个字符串"dontlook"，密文是"hbpkjdhMWT{qhyhz00u__1wg_zr_dsqbhh}"，所谓的加密只是单纯的把明文

解密脚本：

```
cipher = "hbpkjdhMWT{qhyhz00u__1wg_zr_dsqbhh}"
key = "dontlook"
plain = ""
t = 0
for i in range(len(cipher)):
   if ord(cipher[i])>=97 and ord(cipher[i])<=ord('z'):
       plain+=chr(((ord(cipher[i]) - 97) - (ord(key[t%8]) - 97) + 26)%26 + 97)
   elif ord(cipher[i])>=65 and ord(cipher[i])<=ord('Z'):
       plain+=chr(((ord(cipher[i]) - 65) - (ord(key[t%8]) - 97) + 26)%26 + 65)
   else:
       plain += cipher[i]
       continue
   t+=1
print(plain)
```

0 回复Ta
