

2020 SCTF 部分WriteUp

原创

Hk_Mayfly 于 2020-07-06 09:31:00 发布 967 收藏 1

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 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/qq_39542714/article/details/107172947

版权

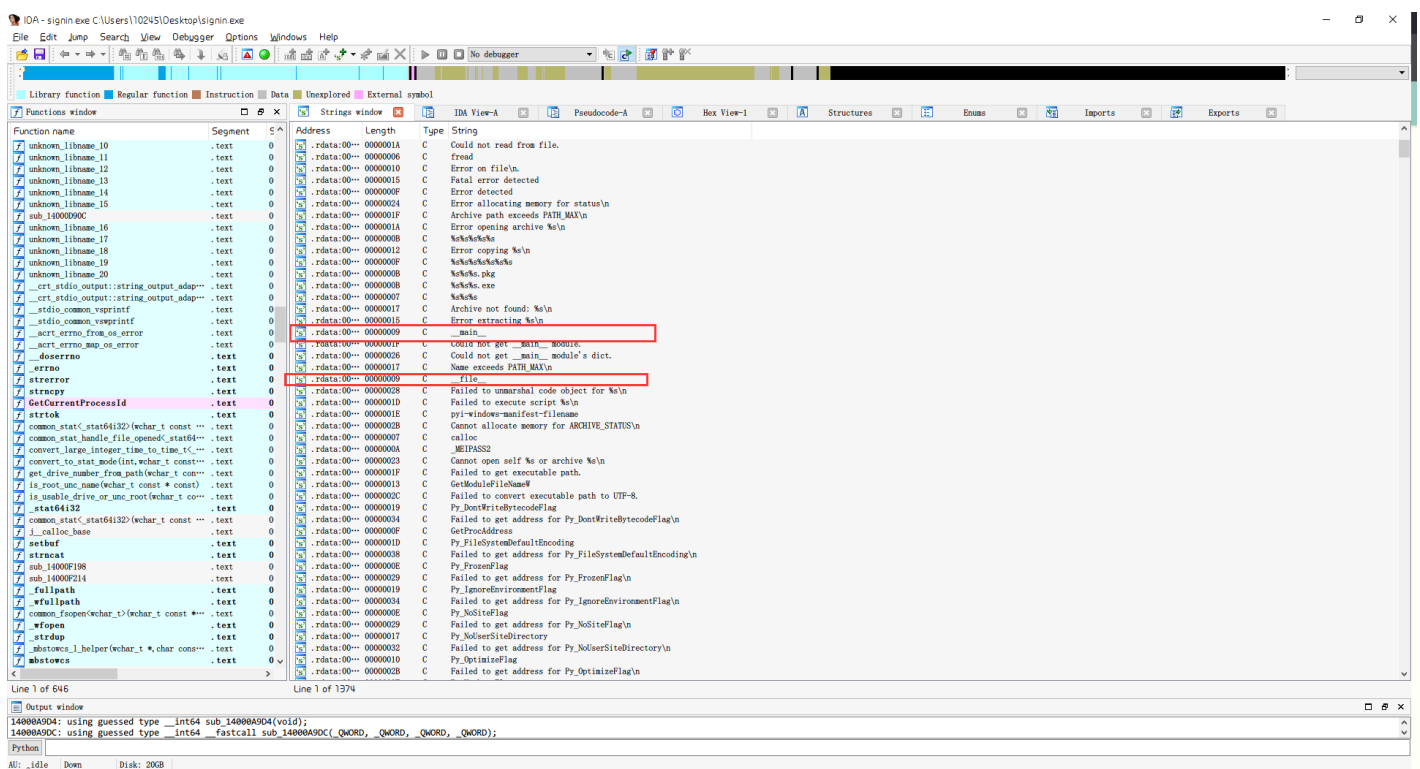
signin

准备

signin.exe: <https://www.lanzous.com/inlQdec11zi>



程序分析



可以判断出，这个程序实际上是由Python打包成的可执行文件，且在运行这个程序时，在同目录下产生了一个tmp.dll文件，猜测是程序调用某些函数的接口。

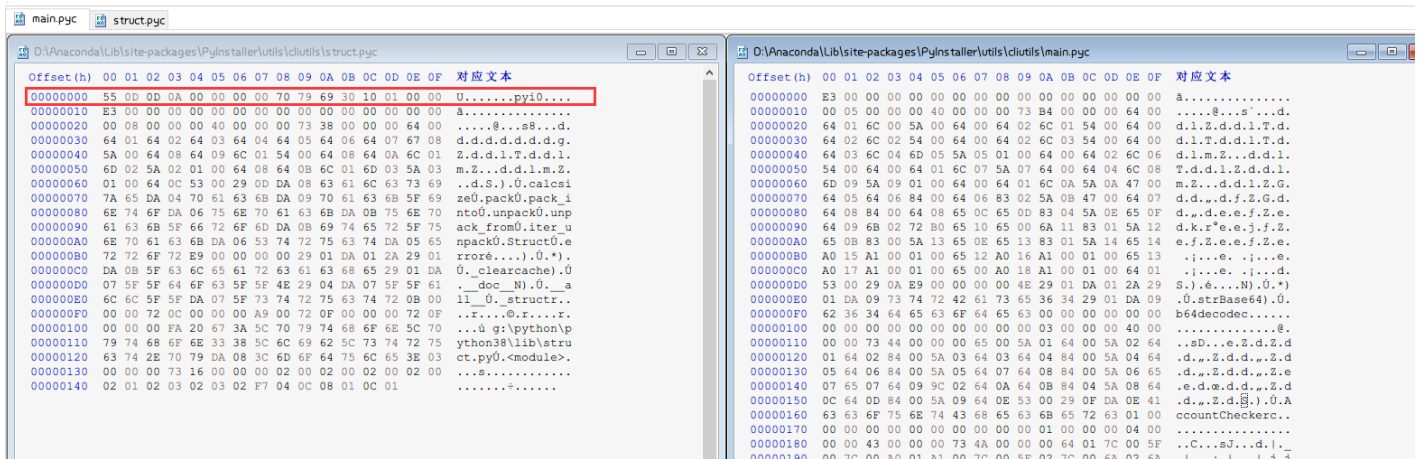
反编译

使用archive_viewer.py反编译为字节码文件

```
python archive_viewer.py signin.exe
```

```
(38002548, 935, 2406, 1, 'x', 'tk\\ttk\\sizegrip.tcl'),
(38003483, 1510, 4255, 1, 'x', 'tk\\ttk\\spinbox.tcl'),
(38004993, 2424, 8898, 1, 'x', 'tk\\ttk\\treeview.tcl'),
(38007417, 1692, 4546, 1, 'x', 'tk\\ttk\\ttk.tcl'),
(38009109, 3046, 8562, 1, 'x', 'tk\\ttk\\utils.tcl'),
(38012155, 1851, 9670, 1, 'x', 'tk\\ttk\\vistaTheme.tcl'),
(38014006, 815, 2867, 1, 'x', 'tk\\ttk\\winTheme.tcl'),
(38014821, 634, 2375, 1, 'x', 'tk\\ttk\\xpTheme.tcl'),
(38015455, 2538, 10252, 1, 'x', 'tk\\unsupported.tcl'),
(38017993, 6928, 26075, 1, 'x', 'tk\\xmfbox.tcl'),
(38024921, 1966212, 1966212, 0, 'z', 'PYZ-00.pyz')]
? x main
to filename? main.pyc
? x struct
to filename? struct.pyc
?
```

修补文件



55 0D 0D 0A 00 00 00 00 70 79 69 30 10 01 00 00

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 对应文本
00000000 55 0D 0D 0A 00 00 00 00 70 79 69 30 10 01 00 00 U.....pyi0....
00000010 E3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....@...s'...d.
00000020 00 05 00 00 00 40 00 00 00 73 B4 00 00 00 64 00 .....@...s'...d.
00000030 64 01 6C 02 5A 00 64 00 64 02 6C 01 54 00 64 00 d.l.Z.d.d.l.T.d.
00000040 64 02 6C 02 54 00 64 00 64 02 6C 03 54 00 64 00 d.l.T.d.d.l.T.d.
00000050 64 03 6C 04 6D 05 5A 05 01 00 64 00 64 02 6C 06 d.l.m.Z...d.d.l.
00000060 54 00 64 00 64 01 6C 07 5A 07 64 00 64 04 6C 08 T.d.d.l.Z.d.d.l.
00000070 6D 09 5A 09 01 00 64 00 64 01 6C 0A 5A 0A 47 00 m.Z...d.d.l.Z.G.
00000080 64 05 64 06 84 00 64 06 83 02 5A 0B 47 00 64 07 d.d...d.f.Z.G.d.
00000090 64 08 84 00 64 08 65 0C 65 00 83 04 5A 0E 65 0F d...d.e.e.f.Z.e.
000000A0 64 09 6B 02 72 B0 65 10 65 00 6A 11 83 01 5A 12 d.k.r*e.e.j.f.Z.
000000B0 65 0B 83 00 5A 13 65 0E 65 13 83 01 5A 14 65 14 e.f.Z.e.e.f.Z.e.
000000C0 A0 17 A1 00 01 00 65 00 A0 18 A1 00 01 00 64 01 .....@...s'...d.
000000D0 53 00 29 0A E9 00 00 00 00 4E 29 01 DA 01 2A 29 S()e...N()*)
000000E0 01 DA 09 73 74 72 42 61 73 65 36 34 29 01 DA 09 .0.strBase64).0.
000000F0 62 36 34 64 65 63 6F 64 65 63 00 00 00 00 00 00 b64decode.....
00000100 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 .....@...s'...d.
00000110 00 00 73 44 00 00 65 00 5A 01 64 00 5A 02 64 .sD...e.Z.d.Z.d.
00000120 01 64 02 84 00 5A 03 64 03 64 04 84 00 5A 04 64 .d...Z.d.d...Z.d.
00000130 05 64 06 84 00 5A 05 64 07 64 08 84 00 5A 06 65 .d...Z.d.d...Z.e.
00000140 07 65 07 64 09 9C 02 64 0A 64 0B 84 04 5A 08 64 .e.d.x.d.d...Z.d.
00000150 0C 64 0D 84 00 5A 09 64 0E 53 00 29 0F DA 0E 41 .d...Z.d.d...Z.d.
00000160 63 63 6F 75 6E 74 43 68 65 63 6B 65 72 63 01 00 ccountChecker...
00000170 00 00 00 00 00 00 00 00 00 01 00 00 00 04 00 .....@...s'...d.
00000180 00 00 43 00 00 00 73 4A 00 00 64 01 7C 00 5F ..C...eJ...d...j.
00000190 00 7C 00 A0 01 A1 00 7C 00 5F 02 7C 00 6A 02 6A .l...l...l...l...l...
```

程序是在Python3.8环境下打包，因此我们需要在Python3.8下使用uncompyle6

```
uncompyle6 main.pyc > main.py
```

得到py文件

```
1 # uncompyle6 version 3.7.2
2 # Python bytecode 3.8 (3413)
3 # Decompiled from: Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)]
4 # Embedded file name: main.py
5 # Compiled at: 1995-09-28 00:18:56
6 # Size of source mod 2**32: 272 bytes
7 import sys
8 from PyQt5.QtCore import *
9 from PyQt5.QtWidgets import *
10 from signin import *
11 from mydata import strBase64
12 from ctypes import *
13 import _ctypes
14 from base64 import b64decode
15 import os
16
17 class AccountChecker:
18
19     def __init__(self):
20         self.dllname = './tmp.dll'
21         self.dll = self._AccountChecker__release_dll()
22         self.enc = self.dll.enc
23         self.enc.argtypes = (c_char_p, c_char_p, c_char_p, c_int)
24         self.enc.restype = c_int
25         self.accounts = {b'SCTFer': b64decode(b'PLHCu+fuJfZmMOMLGHCyWW0q5H5HDN2R5nHn1V30Q0EA')}
26         self.try_times = 0
27
28     def __release_dll(self):
29         with open(self.dllname, 'wb') as (f):
30             f.write(b64decode(strBase64.encode('ascii')))
31         return WinDLL(self.dllname)
32
33     def clean(self):
34         _ctypes.FreeLibrary(self.dll._handle)
35         if os.path.exists(self.dllname):
36             os.remove(self.dllname)
37
38     def _error(self, error_code):
39         errmsg = {0:'Unknown Error',
40                 1:'Memory Error'}
41         QMessageBox.information(None, 'Error', errmsg[error_code], QMessageBox.Abort, QMessageBox.Abort)
42         sys.exit(1)
43
44     def __safe(self, username: bytes, password: bytes):
45         pwd_safe = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
46         status = self.enc(username, password, pwd_safe, len(pwd_safe))
47         return (pwd_safe, status)
48
```

```

49     def check(self, username, password):
50         self.try_times += 1
51         if username not in self.accounts:
52             return False
53         encrypted_pwd, status = self._AccountChecker__safe(username, password)
54         if status == 1:
55             self._AccountChecker__error(1)
56         if encrypted_pwd != self.accounts[username]:
57             return False
58         self.try_times -= 1
59         return True
60
61
62 class SignInWnd(QMainWindow, Ui_QWidget):
63
64     def __init__(self, checker, parent=None):
65         super().__init__(parent)
66         self.checker = checker
67         self.setupUi(self)
68         self.PB_signin.clicked.connect(self.on_confirm_button_clicked)
69
70     @pyqtSlot()
71     def on_confirm_button_clicked(self):
72         username = bytes((self.LE_usrname.text()), encoding='ascii')
73         password = bytes((self.LE_pwd.text()), encoding='ascii')
74         if username == b'' or password == b'':
75             self.check_input_msgbox()
76         else:
77             self.msgbox(self.checker.check(username, password))
78
79     def check_input_msgbox(self):
80         QMessageBox.information(None, 'Error', 'Check Your Input!', QMessageBox.Ok, QMessageBox.Ok)
81
82     def msgbox(self, status):
83         msg_ex = {0:'',
84                 1:'',
85                 2:"It's no big deal, try again!",
86                 3:'Useful information is in the binary, guess what?'}
87         msg = 'Succeeded! Flag is your password' if status else 'Failed to sign in\n' + msg_ex[(self.che
88         QMessageBox.information(None, 'SCTF2020', msg, QMessageBox.Ok, QMessageBox.Ok)
89
90
91 if __name__ == '__main__':
92     app = QApplication(sys.argv)
93     checker = AccountChecker()
94     sign_in_wnd = SignInWnd(checker)
95     sign_in_wnd.show()
96     app.exec()
97     checker.clean()
98     sys.exit()
99 # okay decompiling main.pyc

```



```

16 const char *Str; // [rsp+1D8h] [rbp+1B8h]
17 __int64 v19; // [rsp+1E0h] [rbp+1C0h]
18 int v20; // [rsp+1E8h] [rbp+1C8h]
19
20 v20 = a4; // pwd_safe长度
21 v19 = a3; // pwd_safe
22 Str = a2; // password
23 v17 = a1; // username
24 v4 = &v9;
25 for ( i = 110i64; i; --i )
26 {
27     *(_DWORD *)v4 = -858993460;
28     v4 += 4;
29 }
30 sub_180011078((__int64)&unk_180020003);
31 v10 = j_strlen(Str); // 密码长度
32 v11 = j_strlen(v17); // 用户名长度
33 if ( v10 <= v20 )
34 {
35     while ( v10 < 32 )
36         Str[v10++] = 0;
37     v12 = 0;
38     for ( j = 0; j < 4; ++j ) // 循环4次
39     {
40         _mm_lfence();
41         memset(&Dst, 0, sizeof(Dst));
42         j_memcpy(&Dst, &Str[v12], 8ui64); // 每次给Dst传入password的8字节数据
43         Dst = sub_180011311(Dst); // 加密函数
44         j_memcpy((void *)(v12 + v19), &Dst, 8ui64);
45         v12 += 8;
46     }
47     for ( k = 0; k < 32; ++k ) // 数据异或
48     {
49         v16 = k;
50         *(_BYTE *)(v19 + k) ^= v17[k % v11];
51     }
52     *(_BYTE *)(v19 + 32) = 0;
53     v6 = 0i64;
54 }
55 else
56 {
57     v6 = 1i64;
58 }
59 v7 = v6;
60 sub_18001131B(&v9, &unk_18001A4A0);
61 return v7;
62 }

```

观察代码，实际操作可以分为两部分，逆向分析

异或操作

第47~54行代码实际上就是将Dst与用户名循环异或，最后得到
b64decode(b'PLHCu+fujfZmMOMLGHCyWWOq5H5HDN2R5nHnIV30Q0EA')，因此我们只需要逆向异或就能
得到加密后的Dst

```

from base64 import *

username = "SCTFer"
pwd_safe = b64decode('PLHCu+fujfZmMOMLGHcyWwOq5H5HDN2R5nHn1V30Q0EA')
# print (len(pwd_safe))
num = ["%02x" % x for x in pwd_safe]
hex_num = [int(x, 16) for x in num]

print(num)
# print (len(num))
for i in range(32):
    hex_num[i] ^= ord(username[i % len(username)])
# print (hex_num)
hex_nums = bytes.fromhex(''.join([hex(x)[2:].rjust(2, '0') for x in hex_num]))

print (hex_nums)

```

得到

```
b'o\xfd\x96\xfd\x82\x9c\xde\x52v\x86yK3\xe6\x1f\x06\xd8\xb7=\x13J\xb8\xe3\xb52\xb3\xd38\x86\x10\x02\x00'
```

加密操作

每次传入了8字节数据进行加密（总共64字节），打开sub_180011311函数

```

1  int64 __fastcall sub_180013CE0(__int64 a1)
2  {
3  __int64 *v1; // rdi
4  signed __int64 i; // rcx
5  __int64 v4; // [rsp+0h] [rbp-20h]
6  int j; // [rsp+24h] [rbp+4h]
7  __int64 v6; // [rsp+120h] [rbp+100h]
8
9  v6 = a1;
10 v1 = &v4;
11 for ( i = 66i64; i; --i )
12 {
13 *( _DWORD *)v1 = -858993460;
14 v1 = (__int64 *)((char *)v1 + 4);
15 }
16 sub_180011078((__int64)&unk_180020003);
17 for ( j = 0; j < 64; ++j )
18 {
19 if ( v6 >= 0 )
20 v6 = sub_18001130C(2 * v6);
21 else
22 v6 = sub_18001130C(2 * v6 ^ 0xB0004B7679FA26B3u164);
23 }
24 return v6;
25 }

```

仔细观察代码，实际上这部分代码是使用CRC32的查表法，对数据进行加密。

加密原理实际上就是CRC32算法---输入一组长度32的字符串，每8个字节分为1组，共4组。对每一组取首位，判断正负。正值，左移一位；负值，左移一位，再异或0xB0004B7679FA26B3。重复判断操作64次，得到查表法所用的表。

因此我们只需要将整个加密过程逆向操作得到查表法的表，再进行CRC32计算，就能得到输入。

```

secret = []

# for i in range(4):
#     secret.append(int(hex_nums[i*8:(i + 1) * 8][::-1].hex(),16))

for i in range(4):
    secret.append(int.from_bytes(hex_nums[i*8:(i + 1) * 8], byteorder="little"))

print (secret)

key = 0xB0004B7679FA26B3

flag = ""

for s in secret:
    for i in range(64):
        sign = s & 1
        if sign == 1:
            s ^= key
        s //= 2
        if sign == 1:
            s |= 0x8000000000000000
    print(hex(s))
    j = 0
    while j < 8:
        flag += chr(s&0xFF)
        s >>= 8
        j += 1
print(flag)

```

因为计算机中采用小端排序，因此需要注意分组倒序。得到

```

[13105084052108931695, 2298581059073635890, 16408946688824694790, 148766366331253429]
0x3165577b46544353
0x5f6f545f336d3063
0x3230325f66746353
0x7d21215f65725f30
SCTF{Welc0m3_To_Sctf_2020_re_!!}

```

脚本


```

from base64 import *

username = "SCTFer"
pwd_safe = b64decode('PLHCu+fujfZmMOMLGHCyWwOq5H5HDN2R5nHn1V30Q0EA')
# print (len(pwd_safe))
num = ["%02x" % x for x in pwd_safe]
hex_num = [int(x, 16) for x in num]

print(num)
# print (len(num))
for i in range(32):
    hex_num[i] ^= ord(username[i % len(username)])
# print (hex_num)
hex_nums = bytes.fromhex(''.join([hex(x)[2:].rjust(2, '0') for x in hex_num]))

print (hex_nums)

secret = []

# for i in range(4):
#     secret.append(int(hex_nums[i*8:(i + 1) * 8][::-1].hex(),16))

for i in range(4):
    secret.append(int.from_bytes(hex_nums[i*8:(i + 1) * 8], byteorder="little"))

print (secret)

key = 0xB0004B7679FA26B3

flag = ""

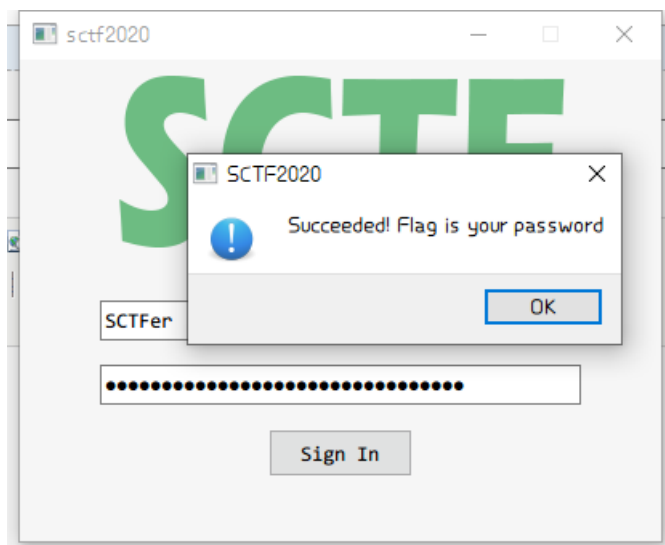
for s in secret:
    for i in range(64):
        sign = s & 1
        if sign == 1:
            s ^= key
        s //= 2
        if sign == 1:
            s |= 0x8000000000000000
    print(hex(s))
    j = 0
    while j < 8:
        flag += chr(s&0xFF)
        s >>= 8
        j += 1
print(flag)

```

get flag!

username:SCTFer

password:SCTF{We1c0m3_To_Sctf_2020_re_!!}



get_up

```
1 void sub_402700()
2 {
3     char *v0; // ST14_4
4     HMODULE lpAddress; // ST18_4
5     DWORD f101dProtect; // [esp+Ch] [ebp-14h]
6     char Dst; // [esp+10h] [ebp-10h]
7
8     printf("you should give me a word:");
9     memset(&Dst, 0, 0xAu);
10    scanf_s("%s", &Dst, 10);
11    if ( strlen(&Dst) > 6 || !sub_401DF0(&Dst) )
12    {
13        sub_401080(std::cout, "try again");
14        exit(0);
15    }
16    v0 = sub_402B00(".reloc");
17    lpAddress = GetModuleHandleW(0) + (*((__DWORD *)v0 + 3) >> 2);
18    VirtualProtect(lpAddress, *((__DWORD *)v0 + 4), 0x40u, &f101dProtect);
19    sub_402610((int)lpAddress, &Dst);
20 }
```

这是实际就是在说明输入字符串长度不大于6，且通过sub_401DF0返回1

```

1 char __cdecl sub_401DF0(char *Str)
2 {
3     char result; // a1
4     size_t Size; // [esp+Ch] [ebp-90h]
5     signed int i; // [esp+14h] [ebp-88h]
6     signed int j; // [esp+14h] [ebp-88h]
7     char v5; // [esp+18h] [ebp-84h]
8     char Str2; // [esp+30h] [ebp-6Ch]
9     char Dst; // [esp+54h] [ebp-48h]
10    char v8[39]; // [esp+55h] [ebp-47h]
11    unsigned __int8 v9[16]; // [esp+7Ch] [ebp-20h]
12    int v10; // [esp+98h] [ebp-4h]
13
14    Size = strlen(Str);
15    for ( i = 0; i < 1000000; ++i )
16        sub_402C90(Str, Size, (int)v9);
17    memset(&Dst, 0, 0x28u);
18    sub_4015C0("0123456789abcdef");
19    v10 = 0;
20    for ( j = 0; j < 16; ++j )
21    {
22        *(&Dst + 2 * j) = *(_BYTE *)sub_4019A0((signed int)v9[j] >> 4);
23        v8[2 * j] = *(_BYTE *)sub_4019A0(v9[j] % 16);
24    }
25    printf("\n");
26    strcpy(&Str2, "32c1d123c193aecc4280a5d7925a2504");// sycsyc
27    if ( !strcmp(&Dst, &Str2) )
28    {
29        v10 = -1;
30        std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_traits<char>,std::allocator<char>>(&v5);
31        result = 1;

```

返回1，实际就是满足加密后的Dst与32c1d123c193aecc4280a5d7925a2504相同，实际是MD5加密，得到输入为sycsyc

接下来，程序查找.reioc段，并将.reioc段数据与sycsyc循环异或，写出IDAPython脚本

```

beg_adr = 0x405000
dst = "sycsyc"
for i in range(0,0x200,16):
    for j in range(16):
        PatchByte(beg_adr+i+j,Byte(beg_adr+i+j)^ord(dst[j%6]))

```

然后跟去混淆方法类似，先转换为Data，再强制分析数据，转换为函数。

```

1 int sub_405000()
2 {
3     sub_403120(); // flag plz!
4     return sub_4027F0();
5 }

```

打开sub_4027F0函数

```

1 void sub_4027F0()
2 {
3     char *v0; // ST18_4
4     HMODULE lpAddress; // ST1C_4
5     signed int i; // [esp+10h] [ebp-40h]
6     DWORD f101dProtect; // [esp+14h] [ebp-3Ch]
7     char Dst[40]; // [esp+18h] [ebp-38h]
8     char Str[12]; // [esp+40h] [ebp-10h]
9
10    memset(Dst, 0, 0x28u);
11    memset(Str, 0, 0xAu);
12    scanf_s("%s", Dst, 40);
13    if ( strlen(Dst) != 30 ) // flag长度为30
14    {
15        sub_401080(std::cout, "try again");
16        exit(0);
17    }
18    for ( i = 0; i < 5; ++i )
19        Str[i] = Dst[i];
20    v0 = sub_402B00(".ebata");
21    lpAddress = GetModuleHandleW(0) + (*((__DWORD *)v0 + 3) >> 2);
22    VirtualProtect(lpAddress, *((__DWORD *)v0 + 4), 0x40u, &f101dProtect);
23    sub_4025B0((int)lpAddress, Str);
24    sub_404000();
25 }

```

这里实际和上面差不多，输入长度为30的flag，取前5个字符，与.ebata段的部分循环异或。（实际就是得到下面sub_404000函数的代码）。这里可以合理猜测flag的前五个字符为**SCTF{**，写出脚本

```

beg_adr = 0x404000
dst = "SCTF{"
for i in range(16,96):
    PatchByte(beg_adr+i,Byte(beg_adr+i)^ord(dst[i%5]))

```

跟上面一样将.ebata段转换为data，再分析代码，转换为函数，开头如果提示栈不平衡，重新单独分析一下就行。

```

13 char v12; // [esp+99Dh] [ebp-Fh]
14 char v13; // [esp+99Eh] [ebp-Eh]
15 char v14; // [esp+99Fh] [ebp-Dh]
16 char v15; // [esp+9A0h] [ebp-Ch]
17 char v16; // [esp+9A1h] [ebp-Bh]
18 char v17; // [esp+9A2h] [ebp-Ah]
19 char v18; // [esp+9A3h] [ebp-9h]
20 char v19; // [esp+9A4h] [ebp-8h]
21
22 v11 = 's';
23 v12 = 'y';
24 v13 = 'c';
25 v14 = 'l';
26 v15 = 'o';
27 v16 = 'v';
28 v17 = 'e';
29 v18 = 'r';
30 v19 = 0;
31 memset(Dst, 0, 0x28u);
32 for ( i = 0; i < strlen(Str); ++i )
33     Dst[i] = Str[i];
34 for ( j = 0; (signed int)j < 256; ++j )
35 {
36     v9[j] = j;
37     v1 = strlen(&v11);
38     v3[j] = *(&v11 + j % v1);
39 }
40 v5 = 0;
41 for ( k = 0; k < 256; ++k )
42 {
43     v5 = (v3[k] + v9[k] + v5) % 256;
44     v4 = v9[k];
45     v9[k] = v9[v5];
46     v9[v5] = v4;
47 }
48 return sub_401A70((int)v9, Dst);
49}

```

这里一个函数实际就是RC4生成S-BOX，另一个函数就是加密函数，key值为syclover

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef unsigned longULONG;

#pragma warning(disable:4996)

/*初始化函数*/
void rc4_init(unsigned char* s, unsigned char* key, unsigned long Len)
{
    int i = 0, j = 0;
    char k[256] = { 0 };
    unsigned char tmp = 0;
    for (i = 0; i < 256; i++)
    {
        s[i] = i;
        k[i] = key[i % Len];
    }
    for (i = 0; i < 256; i++)
    {
        j = (j + s[i] + k[i]) % 256;
        tmp = s[i];
        s[i] = s[j]; // 交换s[i]和s[j]
        s[j] = tmp;
    }
}

/*加解密*/
void rc4_crypt(unsigned char* s, unsigned char* Data, unsigned long Len)

```

```

{
    int i = 0, j = 0, t = 0;
    unsigned long k = 0;
    unsigned char tmp;
    for (k = 0; k < Len; k++)
    {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        tmp = s[i];
        s[i] = s[j]; // 交换s[x]和s[y]
        s[j] = tmp;
        t = (s[i] + s[j]) % 256;
        Data[k] ^= s[t];
    }
}

int main()
{
    unsigned char s[256] = { 0 }, s2[256] = { 0 }; // S-box
    char key[256] = { "syclover" };
    char pData[512] = { 0 };
    int v12[] = { 128,85,126,45,209,9,37,171,60,86,149,196,54,19,237,114,36,147,178,200,69,236,22,107,103,2
    int i;
    //char m[32] = { 0 };

    for (int i = 0; i < 30; ++i) {
        pData[i] = (char)v12[i];
    }

    unsigned long len = strlen(pData);

    printf("pData=%s\n", pData);
    printf("key=%s,length=%d\n\n", key, strlen(key));
    rc4_init(s, (unsigned char*)key, strlen(key)); // 已经完成了初始化
    printf("完成对S[i]的初始化, 如下: \n\n");
    for (i = 0; i < 256; i++)
    {
        printf("%02X", s[i]);
        if (i && (i + 1) % 16 == 0) putchar('\n');
    }
    printf("\n\n");
    for (i = 0; i < 256; i++) // 用s2[i]暂时保留经过初始化的s[i], 很重要的!!!
    {
        s2[i] = s[i];
    }

    printf("已经初始化, 现在解密:\n\n");
    printf("len = %d\n", len);
    rc4_crypt(s, (unsigned char*)pData, len);
    printf("pData=%s\n\n", pData);

    system("PAUSE");
    return 0;
}

```

```
pData=€U~-?%?V?粵6[韓$撈葦?kg□ 棒  
key=sycl0ver, length=8
```

完成对s[i]的初始化, 如下:

```
315052C1CDA707393D04FDC42A72DF60  
E3BDE261E463EAF7FE09F9C3CE9E256A  
4CDC13F0833B24DBDD7FEF8C3EE1266F  
9206F796D0B4E7F85F78F2AA4432365A  
ECA67B08C02E9B84559CAE650A105388  
4946AB0C1F597EE5B3F6481AD1A4676B  
BA87734BA2CF1D18D4BB2C7093AC0F7A  
51B5218545021C9F815C4A951B16D2C7  
E0DA58A91ED7C26971D62D75BC66FA30  
AD998B77419722B8B2D347C679140BF4  
3F112BDEEEF3F5AFA58691BE8E8FE8E6  
B14E37D5FC4D5E42CA35C86EA389156D  
038A8DD876CBC5F16CB9176468E95629  
90ED740DEBB0A19480C97CB728055B23  
9DA02798437DBF4F1234A8D957339A01  
542F3C6238FB820E3A195DB64020CC00
```

已经初始化, 现在解密:

```
len = 30  
pData=SCTF{zzz~(|3[____]_rc4_5o_e4sy}
```

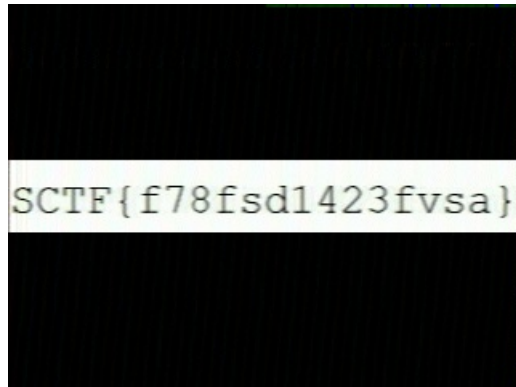
请按任意键继续. . .

SCTF{zzz~(|3[____]_rc4_5o_e4sy}

Can you hear

音频文件: <https://wwa.lanzous.com/iMv4zecuW3g>

音频是SSTV, 用软件Robot36接受音频中的数据



```
SCTF{f78fsd1423fvsa}
```

doudizhu

只要将手里牌出完, 就能获得flag

SCTF{Did-you-enj0y-1t?~::~::~::~::~!!~!!!~}

确定



5



1

