【SCTF2020】signin WriteUp

原创

古月浪子 于 2020-07-06 11:03:27 发布 787 收藏 1

文章标签： CTF

本文链接：https://blog.csdn.net/tqydyqt/article/details/107152507

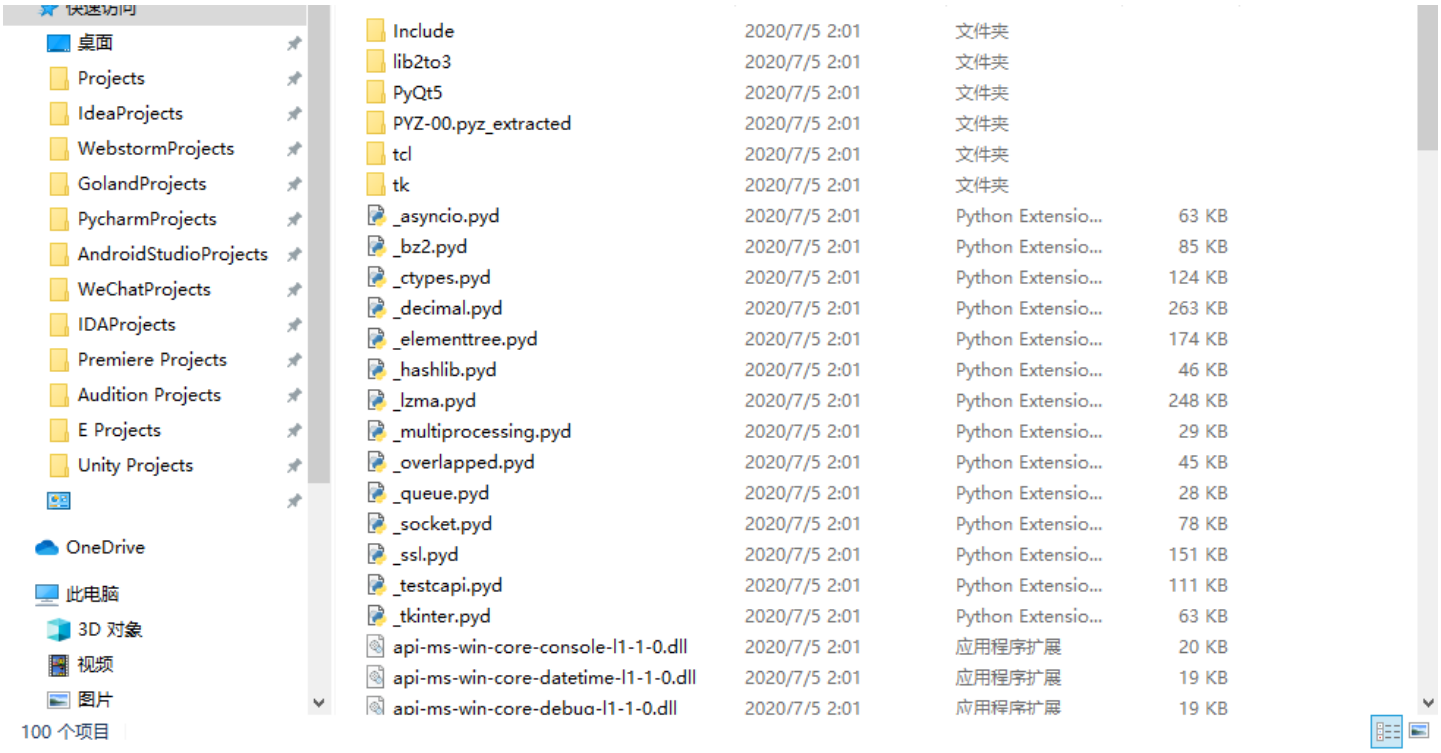
一道SCTF的逆向题，做了很久很久才做起，唉…

下载附件，打开发现是一个GUI程序

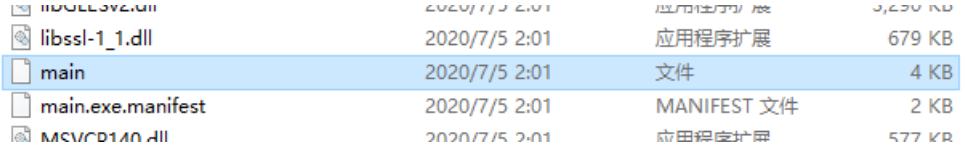
IDA打开，根据经验分析出这是一个pyqt程序，于是用解包脚本处理这个exe

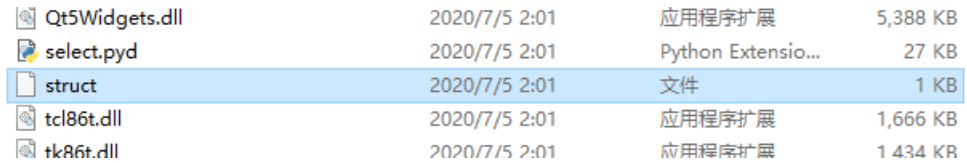
脚本Github：pyinstxtractor

执行脚本后得到了一个解包后的文件夹

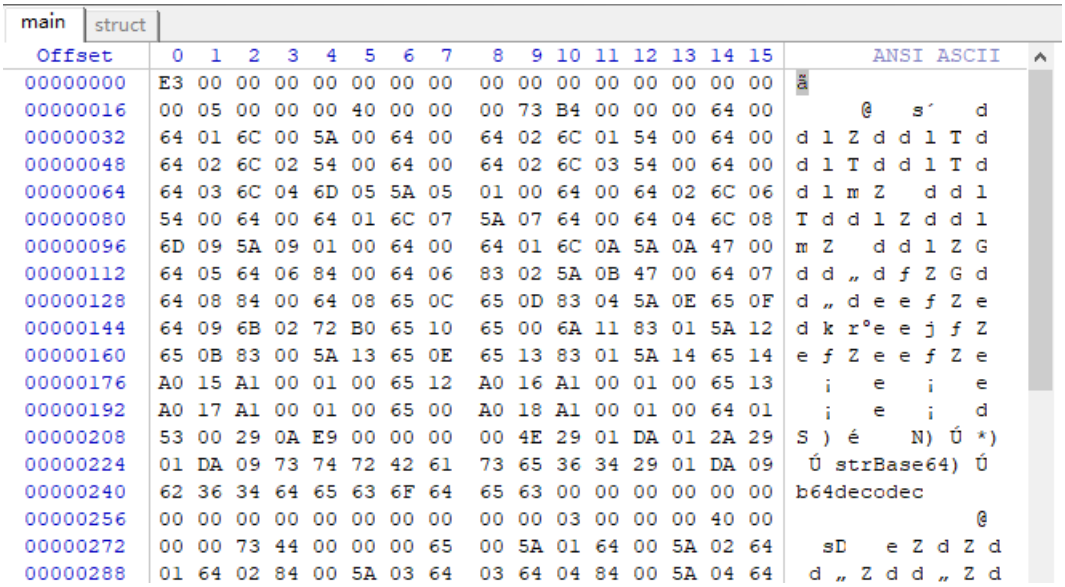


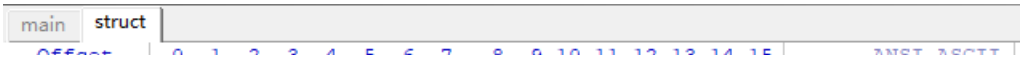
在这里我们重点关注main和struct文件





用WinHex打开，发现main其实是一个没有pyc文件头的pyc文件，而struct中有文件头





Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	55	0D	0D	0A	00	00	00	00	70	79	69	30	10	01	00	00	U pyi0
00000016	E3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ã
00000032	00	08	00	00	00	40	00	00	00	73	38	00	00	00	64	00	@ s8 d
00000048	64	01	64	02	64	03	64	04	64	05	64	06	64	07	67	08	d d d d d d d g
00000064	5A	00	64	08	64	09	6C	01	54	00	64	08	64	0A	6C	01	Z d d l T d d l
00000080	6D	02	5A	02	01	00	64	08	64	0B	6C	01	6D	03	5A	03	m Z d d l m Z
00000096	01	00	64	0C	53	00	29	0D	DA	08	63	61	6C	63	73	69	d S) Ú calcsi
00000112	7A	65	DA	04	70	61	63	6B	DA	09	70	61	63	6B	5F	69	zeÚ packÚ pack_i
00000128	6E	74	6F	DA	06	75	6E	70	61	63	6B	DA	0B	75	6E	70	ntoÚ unpackÚ unp
00000144	61	63	6B	5F	66	72	6F	6D	DA	0B	69	74	65	72	5F	75	ack_fromÚ iter_u
00000160	6E	70	61	63	6B	DA	06	53	74	72	75	63	74	DA	05	65	npackÚ StructÚ e
00000176	72	72	6F	72	E9	00	00	00	00	29	01	DA	01	2A	29	01	rroré) Ú *)
00000192	DA	0B	5F	63	6C	65	61	72	63	61	63	68	65	29	01	DA	Ú _clearcache) Ú
00000208	07	5F	5F	64	6F	63	5F	5F	4E	29	04	DA	07	5F	5F	61	__doc__N) Ú __a
00000224	6C	6C	5F	5F	DA	07	5F	73	74	72	75	63	74	72	0B	00	ll__Ú _structr
00000240	00	00	72	0C	00	00	00	A9	00	72	0F	00	00	00	72	0F	r © r r
00000256	00	00	00	FA	20	67	3A	5C	70	79	74	68	6F	6E	5C	70	ú g:\python\p
00000272	79	74	68	6F	6E	33	38	5C	6C	69	62	5C	73	74	72	75	ython38\lib\stru
00000288	63	74	2E	70	79	DA	08	3C	6D	6F	64	75	6C	65	3E	03	ct.pyÚ <module>

于是我们把struct的前16个字节添加到main的最前面，并改名为main.pyc，得到如下文件

struct main.pyc

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	55	0D	0D	0A	00	00	00	00	70	79	69	30	10	01	00	00	U pyi0
00000016	E3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ã
00000032	00	05	00	00	00	40	00	00	00	73	B4	00	00	00	64	00	@ s´ d
00000048	64	01	6C	00	5A	00	64	00	64	02	6C	01	54	00	64	00	d l Z d d l T d
00000064	64	02	6C	02	54	00	64	00	64	02	6C	03	54	00	64	00	d l T d d l T d
00000080	64	03	6C	04	6D	05	5A	05	01	00	64	00	64	02	6C	06	d l m Z d d l
00000096	54	00	64	00	64	01	6C	07	5A	07	64	00	64	04	6C	08	T d d l Z d d l
00000112	6D	09	5A	09	01	00	64	00	64	01	6C	0A	5A	0A	47	00	m Z d d l Z G
00000128	64	05	64	06	84	00	64	06	83	02	5A	0B	47	00	64	07	d d „ d ƒ Z G d
00000144	64	08	84	00	64	08	65	0C	65	0D	83	04	5A	0E	65	0F	d „ d e e ƒ Z e
00000160	64	09	6B	02	72	B0	65	10	65	00	6A	11	83	01	5A	12	d k r°e e j ƒ Z
00000176	65	0B	83	00	5A	13	65	0E	65	13	83	01	5A	14	65	14	e ƒ Z e e ƒ Z e
00000192	A0	15	A1	00	01	00	65	12	A0	16	A1	00	01	00	65	13	¡ e ¡ e
00000208	A0	17	A1	00	01	00	65	00	A0	18	A1	00	01	00	64	01	¡ e ¡ d
00000224	53	00	29	0A	E9	00	00	00	00	4E	29	01	DA	01	2A	29	S) é N) Ú *)
00000240	01	DA	09	73	74	72	42	61	73	65	36	34	29	01	DA	09	Ú strBase64) Ú
00000256	62	36	34	64	65	63	6F	64	65	63	00	00	00	00	00	00	b64decodec
00000272	00	00	00	00	00	00	00	00	00	00	03	00	00	00	40	00	@
00000288	00	00	73	44	00	00	00	65	00	5A	01	64	00	5A	02	64	sD e Z d Z d

然后命令行执行pip3 install uncompyle6、uncompyle6.exe main.pyc > main.py，得到python源码

```
import sys
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from signin import *
from mydata import strBase64
from ctypes import *
import _ctypes
from base64 import b64decode
import os

class AccountChecker:

    def __init__(self):
        self.dllname = './tmp.dll'
        self.dll = self._AccountChecker__release_dll()
        self.enc = self.dll.enc
        self.enc.argtypes = (c_char_p, c_char_p, c_char_p, c_int)
        self.enc.restype = c_int
        self.accounts = {b'SCTFer': b64decode(b'PLHCu+fujfZmMOMLGHCyWWOq5H5HDN2R5nHnlV30Q0EA')}
        self.try_times = 0
```

```
    def __release_dll(self):
        with open(self.dllname, 'wb') as (f):
            f.write(b64decode(strBase64.encode('ascii')))
        return WinDLL(self.dllname)

    def clean(self):
        _ctypes.FreeLibrary(self.dll._handle)
        if os.path.exists(self.dllname):
            os.remove(self.dllname)

    def _error(self, error_code):
        errormsg = {0:'Unknown Error',
         1:'Memory Error'}
        QMessageBox.information(None, 'Error', errormsg[error_code], QMessageBox.Abort, QMessageBox.Abort)
        sys.exit(1)

    def __safe(self, username: bytes, password: bytes):
        pwd_safe = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
        status = self.enc(username, password, pwd_safe, len(pwd_safe))
        return (pwd_safe, status)

    def check(self, username, password):
        self.try_times += 1
        if username not in self.accounts:
            return False
        encrypted_pwd, status = self._AccountChecker__safe(username, password)
        if status == 1:
            self._AccountChecker__error(1)
        if encrypted_pwd != self.accounts[username]:
            return False
        self.try_times -= 1
        return True


class SignInWnd(QMainWindow, Ui_QWidget):

    def __init__(self, checker, parent=None):
        super().__init__(parent)
        self.checker = checker
        self.setupUi(self)
        self.PB_signin.clicked.connect(self.on_confirm_button_clicked)

    @pyqtSlot()
    def on_confirm_button_clicked(self):
        username = bytes((self.LE_usrname.text()), encoding='ascii')
        password = bytes((self.LE_pwd.text()), encoding='ascii')
        if username == b'' or password == b'':
            self.check_input_msgbox()
        else:
            self.msgbox(self.checker.check(username, password))

    def check_input_msgbox(self):
        QMessageBox.information(None, 'Error', 'Check Your Input!', QMessageBox.Ok, QMessageBox.Ok)

    def msgbox(self, status):
        msg_ex = {0:'',
         1:'',
         2:"It's no big deal, try again!",
```

```
        3:'Useful information is in the binary, guess what?'}
        msg = 'Succeeded! Flag is your password' if status else 'Failed to sign in\n' + msg_ex[(self.checker.try
_times % 4)]
        QMessageBox.information(None, 'SCTF2020', msg, QMessageBox.Ok, QMessageBox.Ok)


if __name__ == '__main__':
    app = QApplication(sys.argv)
    checker = AccountChecker()
    sign_in_wnd = SignInWnd(checker)
    sign_in_wnd.show()
    app.exec()
    checker.clean()
    sys.exit()
```

从源码中可以读出来，用户名为SCTFer，密码的计算方式是释放一个tmp.dll并调用其中的enc函数，通过传入用户名、密码来计算一个输出，并把输出和base64解码以后的数据对比
我们打开程序就能在相同目录下找到tmp.dll，用IDA打开分析一波~

```
signed __int64 __fastcall enc_0(char *a1, char *a2, char *a3, int a4)
{
  char *v4; // rdi
  signed __int64 i; // rcx
  signed __int64 v6; // rax
  signed __int64 v7; // rdi
  char v9; // [rsp+0h] [rbp-20h]
  int v10; // [rsp+24h] [rbp+4h]
  int v11; // [rsp+44h] [rbp+24h]
  int v12; // [rsp+64h] [rbp+44h]
  int j; // [rsp+84h] [rbp+64h]
  __int64 Dst; // [rsp+A8h] [rbp+88h]
  int k; // [rsp+C4h] [rbp+A4h]
  __int64 v16; // [rsp+198h] [rbp+178h]
  const char *username; // [rsp+1D0h] [rbp+1B0h]
  const char *password; // [rsp+1D8h] [rbp+1B8h]
  char *output; // [rsp+1E0h] [rbp+1C0h]
  int length; // [rsp+1E8h] [rbp+1C8h]

  length = a4;
  output = a3;
  password = a2;
  username = a1;
  v4 = &v9;
  for ( i = 110i64; i; --i )
  {
    *v4 = 0xCCCCCCCC;
    v4 += 4;
  }
  sub_180011078(&unk_180020003);
  v10 = j_strlen(password);
  v11 = j_strlen(username);
  if ( v10 <= length )
  {
    while ( v10 < 32 )
      password[v10++] = 0;
    v12 = 0;
    for ( j = 0; j < 4; ++j )
    {
      _mm_lfence();
      memset(&Dst, 0, sizeof(Dst));
      j_memcpy(&Dst, &password[v12], 8ui64);
      Dst = sub_180011311(Dst);
      j_memcpy(&output[v12], &Dst, 8ui64);
      v12 += 8;
    }
    for ( k = 0; k < 32; ++k )
    {
      v16 = k;
      output[k] ^= username[k % v11];
    }


    output[32] = 0;
    v6 = 0i64;
  }
  else
```

```
  {
    v6 = 1i64;
  }
  v7 = v6;
  sub_18001131B(&v9, &unk_18001A4A0);
  return v7;
}
```

直接就能在函数窗口中找到enc函数，内部是调用的enc_0函数，这里用n更改了几个变量名方便理解加密时的算法
大概逻辑是，把32位的密码分成4组，每组8字节，memcpy给一个__int64，然后对这个数进行某种操作，再把这个数memcpy到输出的对应位置，4组全部处理完后，把输出和用户名进行异或，最后得出真正的输出

```
__int64 __fastcall sub_180013CE0(__int64 a1)
{
  __int64 *v1; // rdi
  signed __int64 i; // rcx
  __int64 v4; // [rsp+0h] [rbp-20h]
  int j; // [rsp+24h] [rbp+4h]
  __int64 v6; // [rsp+120h] [rbp+100h]

  v6 = a1;
  v1 = &v4;
  for ( i = 66i64; i; --i )
  {
    *v1 = -858993460;
    v1 = (v1 + 4);
  }
  sub_180011078(&unk_180020003);
  for ( j = 0; j < 64; ++j )
  {
    if ( v6 >= 0 )
      v6 = sub_18001130C(2 * v6);
    else
      v6 = sub_18001130C(2 * v6 ^ 0xB0004B7679FA26B3ui64);
  }
  return v6;
}
```

具体的对数的操作逻辑是，循环64次，每次都把这个数乘以2，如果之前这个数是负数的话，还要再异或一个数
理清加密逻辑倒是没什么难度，但是对我来说写逆向算法卡了老久时间。。。
最后用c++写出的逆向脚本如下：

```
#include <iostream>

using namespace std;

int main()
{
 unsigned char flag[] = { 0x6f,0xf2,0x96,0xfd,0x82,0x9c,0xde,0xb5,0x32,0x76,0x86,0x79,0x4b,0x33,0xe6,0x1f,0x6,0x
d8,0xb7,0x3d,0x13,0x4a,0xb8,0xe3,0xb5,0x32,0xb3,0xd3,0x38,0x86,0x10,0x2,0 };
 __int64 Dst;
 for (size_t j = 0; j < 4; j++) {
  memcpy_s(&Dst, 8, flag + j * 8, 8);
  for (size_t i = 0; i < 64; i++) {
   if (Dst & 1)
    if (Dst < 0) {
     Dst ^= 0xB0004B7679FA26B3ui64;
     Dst /= 2;
     Dst += 0x8000000000000000;
    }
    else {
     Dst ^= 0xB0004B7679FA26B3ui64;
     Dst /= 2;
    }
   else
    if (Dst < 0) {
     Dst /= 2;
     Dst += 0x8000000000000000;
    }
    else
     Dst /= 2;
  }
  memcpy(flag + j * 8, &Dst, 8);
 }
 cout << flag;
}
```

成功计算出flag！

> SCTF{We1c0m3_To_Sctf_2020_re_!!}