SCTF2020 signin writeup

原创

酸酸菜鱼 于 2020-07-09 13:05:19 发布 322 收藏 1
分类专栏： CTF

本文链接：https://blog.csdn.net/lhk124/article/details/107223858

1.丢进ida，查看字符串。发现带有python相关字眼，且是pyinstaller打包的。

2.用pyinstxtractor将exe编译回pyc

```
[+] Processing D:\data\study\CTF\SCTF\signin (1).exe
[+] Pyinstaller version: 2.1+
[+] Python version: 38
[+] Length of package: 40043445 bytes
[+] Found 1054 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth__tkinter.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_pyqt5.pyc
[+] Possible entry point: main.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python38 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: D:\data\study\CTF\SCTF\signin (1).exe

You can now use a python decompiler on the pyc files within the extracted directory
```


3.使用工具uncompyle6反编译pyc。由于是python3.8写的，所以存在几个情况

- 1.在python3.8环境下编译，可以直接打开。
- 2.缺乏python3.8环境，打不开。winhex添加pyc头部magic number，换成3.8的。经师傅提醒，可以查看同文件夹下的pyc文件，参考并修改。（可留多个版本的pyc文件，作为参考）
- 3.python版本不同，pyc头部magic number不同。

4.打开uncompyle6反编译成功后的main.py

```
# uncompyle6 version 3.7.2
# Python bytecode 3.8 (3413)
# Decompiled from: Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)]
# Embedded file name: main.py
# Compiled at: 1995-09-28 00:18:56
# Size of source mod 2**32: 184549648 bytes
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:
```

```python
class AccountChecker:
# 可知： 1.算法在tmp.dll的enc里  2.用户名为SCTFer
#         密码最终结果为  b'PLHCu+fujfZmMOMLGHCyWWOq5H5HDN2R5nHnlV30Q0EA'

    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\x
        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.checke
        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()
# okay decompiling C:\Users\admin\Desktop\main.pyc
```

运行时将动态链接tmp.dll复制出来，丢入ida分析。如下

```
len_pwdSafe = len(pwd_safe);                    // 32位
pwdSafe = pwd_safe;
pwd = password;
name = username;
v4 = &v9;
for ( i = 110i64; i; --i )
{
  *(_DWORD *)v4 = '■■■■';
  v4 += 4;
}
sub_180011078((__int64)&unk_180020003);
v10 = j_strlen(pwd);
len_username = j_strlen(name);
if ( v10 <= len_pwdSafe )
{
  while ( v10 < 32 )
    pwd[v10++] = 0;                             // 0填充
  v12 = 0;
  for ( j = 0; j < 4; ++j )
  {
    _mm_lfence();
    memset(&Dst, 0, sizeof(Dst));
    j_memcpy(&Dst, &pwd[v12], 8ui64);           // 8个为1组
    Dst = sub_180011311(Dst);                   // 判断值的范围，根据条件进行运算
    j_memcpy((void *)(v12 + pwdSafe), &Dst, 8ui64);
    v12 += 8;
  }
  for ( k = 0; k < 32; ++k )
  {
    v16 = k;
    *(_BYTE *)(pwdSafe + k) ^= name[k % len_username];// v19里的每一个数^v17[k%len(v17)]
  }
  *(_BYTE *)(pwdSafe + 32) = 0;
```

```
__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;                                              // dst
  v1 = &v4;
  for ( i = 66i64; i; --i )
  {
    *(_DWORD *)v1 = '■■■■';                             // CRC32算法
    v1 = (__int64 *)((char *)v1 + 4);
  }
  sub_180011078((__int64)&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;
}
```


值得注意的是：如果按**v6>=0**去编写代码，结果会是错的。需要是**==1**.如下

贴上官方代码，本人代码太丑。

```
from base64 import *
import struct


def u_qword(a):
    return struct.unpack('<Q', a)[0]


def p_qword(a):
    return struct.pack('<Q', a)


username = list(b'SCTFer')
enc_pwd = list(b64decode(b'PLHCu+fujfZmMOMLGHCyWWOq5H5HDN2R5nHnlV30Q0EA'))[:-1] # remove thelast x00
for i in range(32):
    enc_pwd[i] ^= username[i % len(username)]
#print(enc_pwd)
qwords = []
for i in range(4):
    qwords.append(u_qword(bytes(enc_pwd[i*8:i*8+8])))
    # bytes将数据转换成字节“数据”，再用struct.unpack把python的转换成c的，是字符形式的
    # bytes将数据转换成字符串类型，所以可以使用切片，如果没有bytes，不会形成四个，而是每个独立，再构成四个list
    # 参数<表示小端序，Q表示unsigned longlong
#print(qwords)
for i in range(4):   # CRC32类似算法
    qword = qwords[i]
    print(qword)
    for _ in range(64):
        if qword & 1 == 1:
            qword ^= 0xB0004B7679FA26B3
            qword >>= 1
            qword |= 0x8000000000000000
            continue
        else:
            qword >>= 1
        qwords[i] = qword
print(qwords)   # [13105084052108931695, 2298581059073635890, 16408946688824694790, 148766366331253429]
pwd = []
for i in range(4):
    pwd.append(p_qword(qwords[i]).decode())

flag = ''.join(pwd)
print(flag)
#print(pwd)
```

- python
- CRC32基本算法