REVERSE-PRACTICE-BUUCTF-31

原创

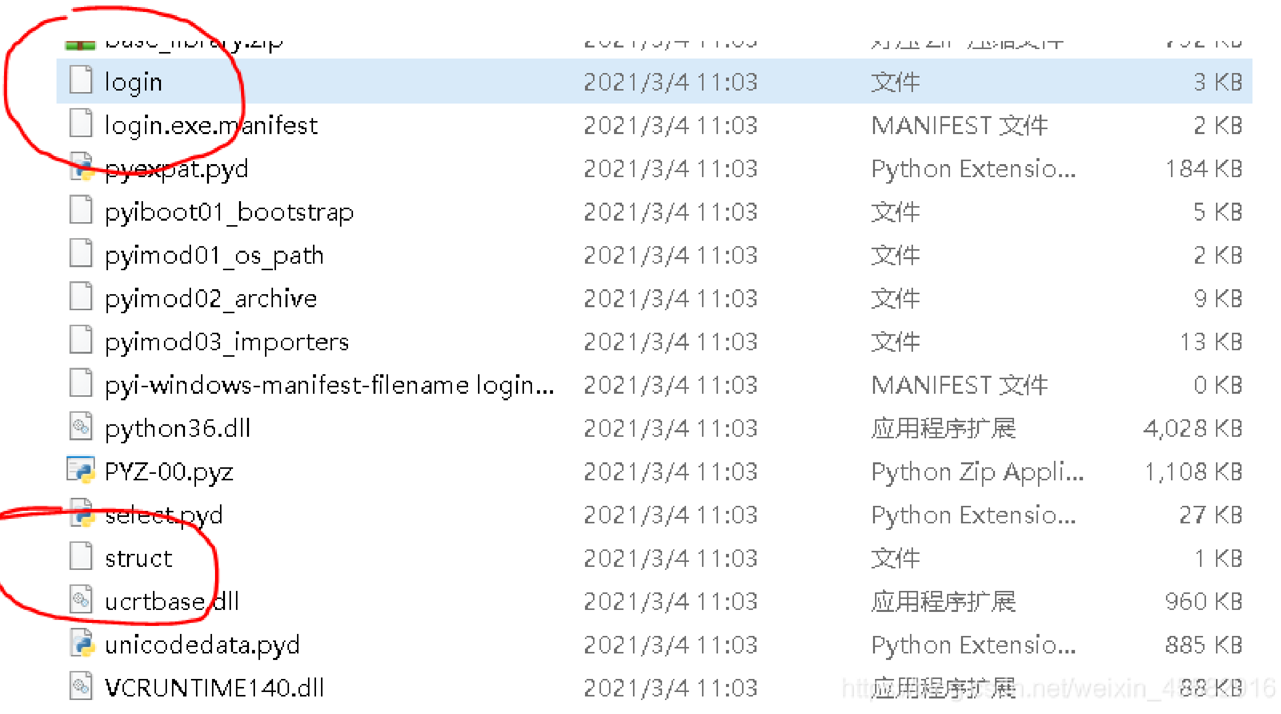
P1umH0 于 2021-03-06 11:16:20 发布 121 收藏
分类专栏： Reverse-BUUCTF

Reverse-BUUCTF 专栏收录该内容
32 篇文章 3 订阅
订阅专栏

REVERSE-PRACTICE-BUUCTF-31

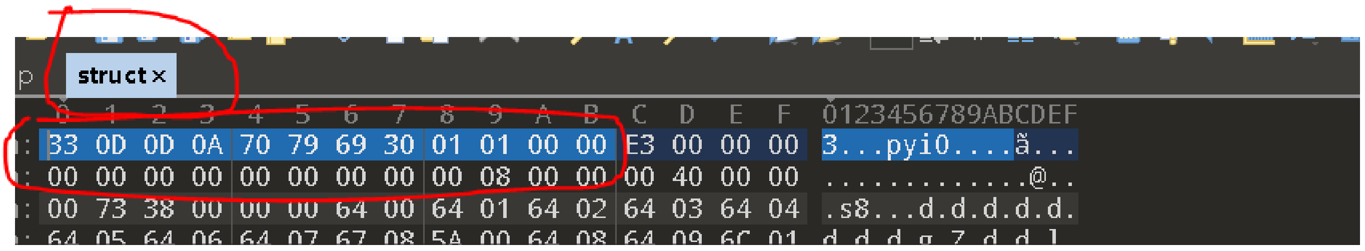
[羊城杯 2020]login

exe程序，运行后输入，无壳，ida分析
没找到主要逻辑，在字符串窗口看到一些“py”的字样，应该是python打包成了exe
用pyinstxtractor.py将exe解包，得到了这些文件

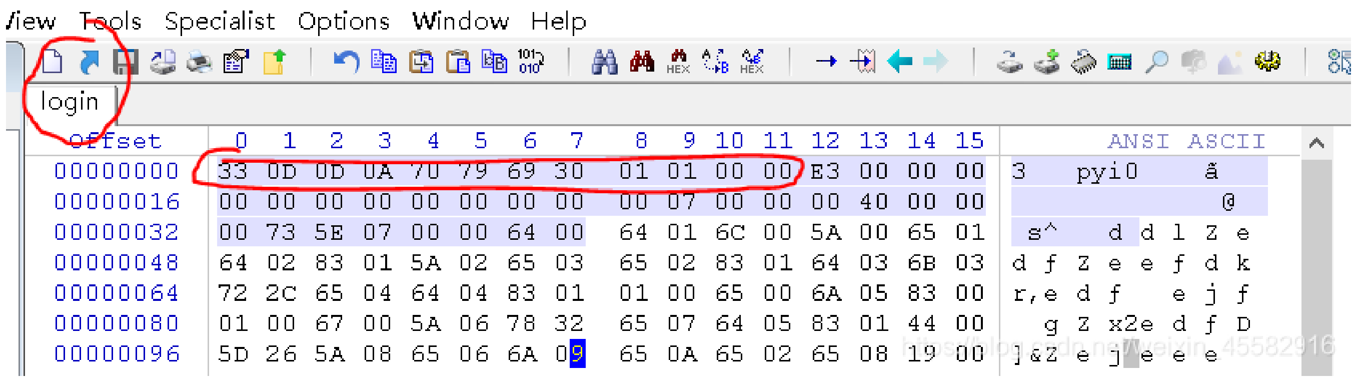


login	2021/3/4 11:03	文件	3 KB
login.exe.manifest	2021/3/4 11:03	MANIFEST 文件	2 KB
pyexpat.pyd	2021/3/4 11:03	Python Extensio...	184 KB
pyiboot01_bootstrap	2021/3/4 11:03	文件	5 KB
pyimod01_os_path	2021/3/4 11:03	文件	2 KB
pyimod02_archive	2021/3/4 11:03	文件	9 KB
pyimod03_importers	2021/3/4 11:03	文件	13 KB
pyi-windows-manifest-filename login...	2021/3/4 11:03	MANIFEST 文件	0 KB
python36.dll	2021/3/4 11:03	应用程序扩展	4,028 KB
PYZ-00.pyz	2021/3/4 11:03	Python Zip Appli...	1,108 KB
select.pyd	2021/3/4 11:03	Python Extensio...	27 KB
struct	2021/3/4 11:03	文件	1 KB
ucrtbase.dll	2021/3/4 11:03	应用程序扩展	960 KB
unicodedata.pyd	2021/3/4 11:03	Python Extensio...	885 KB
VCRUNTIME140.dll	2021/3/4 11:03	应用程序扩展	88 KB

login文件缺少magic number，用struct文件的magic number(头部的12个字节)补充，保存，改后缀名为.pyc



```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
:  33 0D 0D 0A 70 79 69 30 01 01 00 00 E3 00 00 00  3...pyi0....ã...
:  00 00 00 00 00 00 00 00 00 08 00 00 00 40 00 00  .............@..
:  00 73 38 00 00 00 64 00 64 01 64 02 64 03 64 04  .s8...d.d.d.d.d.
```



```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15     ANSI ASCII
00000000   33 0D 0D 0A 70 79 69 30  01 01 00 00 E3 00 00 00  3   pyi0    ã
00000016   00 00 00 00 00 00 00 00  00 07 00 00 00 40 00 00              @
00000032   00 73 5E 07 00 00 64 00  64 01 6C 00 5A 00 65 01   s^   d d l Z e
00000048   64 02 83 01 5A 02 65 03  65 02 83 01 64 03 6B 03  d f Z e e f d k
00000064   72 2C 65 04 64 04 83 01  01 00 65 00 6A 05 83 00  r,e d f   e j f
00000080   01 00 67 00 5A 06 78 32  65 07 64 05 83 01 44 00   g Z x2e d f D
00000096   5D 26 5A 08 65 06 6A 09  65 0A 65 02 65 08 19 00  ]&Z e j e e e
```

用uncompyle6反编译login.pyc，得到python源码

```python
#coding:utf-8
import sys
input1 = input('input something:')
if len(input1) != 14: #输入长度为14
    print('Wrong length!')
    sys.exit()
else:
    code = []
    for i in range(13):# i∈[0,12] code[i]=ord(input1[i]) ^ ord(input1[(i + 1)])
        code.append(ord(input1[i]) ^ ord(input1[(i + 1)]))

    code.append(ord(input1[13]))#code[13]=ord(input1[13])
    a1 = code[2]           #位置变换
    a2 = code[1]
    a3 = code[0]
    a4 = code[3]
    a5 = code[4]
    a6 = code[5]
    a7 = code[6]
    a8 = code[7]
    a9 = code[9]
    a10 = code[8]
    a11 = code[10]
    a12 = code[11]
    a13 = code[12]
    a14 = code[13]
    # 方程组验证
    if (a1 * 88 + a2 * 67 + a3 * 65 - a4 * 5 + a5 * 43 + a6 * 89 + a7 * 25 + a8 * 13 - a9 * 36 + a10 * 15 + a11 * 11 + a12 * 47 - a13 * 60 + a14 * 29 == 22748) & (a1 * 89 + a2 * 7 + a3 * 12 - a4 * 25 + a5 * 41 + a6 * 23 + a7 * 20 - a8 * 66 + a9 * 31 + a10 * 8 + a11 * 2 - a12 * 41 - a13 * 39 + a14 * 17 == 7258) & (a1 * 28 + a2 * 35 + a3 * 16 - a4 * 65 + a5 * 53 + a6 * 39 + a7 * 27 + a8 * 15 - a9 * 33 + a10 * 13 + a11 * 101 + a12 * 90 - a13 * 34 + a14 * 23 == 26190) & (a1 * 23 + a2 * 34 + a3 * 35 - a4 * 59 + a5 * 49 + a6 * 81 + a7 * 25 + (a8 << 7) - a9 * 32 + a10 * 75 + a11 * 81 + a12 * 47 - a13 * 60 + a14 * 29 == 37136) & (a1 * 38 + a2 * 97 + a3 * 35 - a4 * 52 + a5 * 42 + a6 * 79 + a7 * 90 + a8 * 23 - a9 * 36 + a10 * 57 + a11 * 81 + a12 * 42 - a13 * 62 - a14 * 11 == 27915) & (a1 * 22 + a2 * 27 + a3 * 35 - a4 * 45 + a5 * 47 + a6 * 49 + a7 * 29 + a8 * 18 - a9 * 26 + a10 * 35 + a11 * 41 + a12 * 40 - a13 * 61 + a14 * 28 == 17298) & (a1 * 12 + a2 * 45 + a3 * 35 - a4 * 9 - a5 * 42 + a6 * 86 + a7 * 23 + a8 * 85 - a9 * 47 + a10 * 34 + a11 * 76 + a12 * 43 - a13 * 44 + a14 * 65 == 19875) & (a1 * 79 + a2 * 62 + a3 * 35 - a4 * 85 + a5 * 33 + a6 * 79 + a7 * 86 + a8 * 14 - a9 * 30 + a10 * 25 + a11 * 11 + a12 * 57 - a13 * 50 - a14 * 9 == 22784) & (a1 * 8 + a2 * 6 + a3 * 64 - a4 * 85 + a5 * 73 + a6 * 29 + a7 * 2 + a8 * 23 - a9 * 36 + a10 * 5 + a11 * 2 + a12 * 47 - a13 * 64 + a14 * 27 == 9710) & (a1 * 67 - a2 * 68 + a3 * 68 - a4 * 51 - a5 * 43 + a6 * 81 + a7 * 22 - a8 * 12 - a9 * 38 + a10 * 75 + a11 * 41 + a12 * 27 - a13 * 52 + a14 * 31 == 13376) & (a1 * 85 + a2 * 63 + a3 * 5 - a4 * 51 + a5 * 44 + a6 * 36 + a7 * 28 + a8 * 15 - a9 * 6 + a10 * 45 + a11 * 31 + a12 * 7 - a13 * 67 + a14 * 78 == 24065) & (a1 * 47 + a2 * 64 + a3 * 66 - a4 * 5 + a5 * 43 + a6 * 112 + a7 * 25 + a8 * 13 - a9 * 35 + a10 * 95 + a11 * 21 + a12 * 43 - a13 * 61 + a14 * 20 == 27687) & (a1 * 89 + a2 * 67 + a3 * 85 - a4 * 25 + a5 * 49 + a6 * 89 + a7 * 23 + a8 * 56 - a9 * 92 + a10 * 14 + a11 * 89 + a12 * 47 - a13 * 61 - a14 * 29 == 29250) & (a1 * 95 + a2 * 34 + a3 * 62 - a4 * 9 - a5 * 43 + a6 * 83 + a7 * 25 + a8 * 12 - a9 * 36 + a10 * 16 + a11 * 51 + a12 * 47 - a13 * 60 - a14 * 24 == 15317):
        print('flag is GWHT{md5(your_input)}')
        print('Congratulations and have fun!')
    else:
        print('Sorry,plz try again...')
```

z3解方程组，

```python
from z3 import *
a1=Int('a1')
a2=Int('a2')
a3=Int('a3')
a4=Int('a4')
a5=Int('a5')
a6=Int('a6')
a7=Int('a7')
a8=Int('a8')
a9=Int('a9')
a10=Int('a10')
a11=Int('a11')
a12=Int('a12')
a13=Int('a13')
a14=Int('a14')
s=Solver()
s.add(a1 * 88 + a2 * 67 + a3 * 65 - a4 * 5 + a5 * 43 + a6 * 89 + a7 * 25 + a8 * 13 - a9 * 36 + a10 * 15 + a11 * 11 + a12 * 47 - a13 * 60 + a14 * 29 == 22748)
s.add(a1 * 89 + a2 * 7 + a3 * 12 - a4 * 25 + a5 * 41 + a6 * 23 + a7 * 20 - a8 * 66 + a9 * 31 + a10 * 8 + a11 * 2 - a12 * 41 - a13 * 39 + a14 * 17 == 7258)
s.add(a1 * 28 + a2 * 35 + a3 * 16 - a4 * 65 + a5 * 53 + a6 * 39 + a7 * 27 + a8 * 15 - a9 * 33 + a10 * 13 + a11 * 101 + a12 * 90 - a13 * 34 + a14 * 23 == 26190)
s.add(a1 * 23 + a2 * 34 + a3 * 35 - a4 * 59 + a5 * 49 + a6 * 81 + a7 * 25 + (a8 *128) - a9 * 32 + a10 * 75 + a11 * 81 + a12 * 47 - a13 * 60 + a14 * 29 == 37136)
s.add(a1 * 38 + a2 * 97 + a3 * 35 - a4 * 52 + a5 * 42 + a6 * 79 + a7 * 90 + a8 * 23 - a9 * 36 + a10 * 57 + a11 * 81 + a12 * 42 - a13 * 62 - a14 * 11 == 27915)
s.add(a1 * 22 + a2 * 27 + a3 * 35 - a4 * 45 + a5 * 47 + a6 * 49 + a7 * 29 + a8 * 18 - a9 * 26 + a10 * 35 + a11 * 41 + a12 * 40 - a13 * 61 + a14 * 28 == 17298)
s.add(a1 * 12 + a2 * 45 + a3 * 35 - a4 * 9 - a5 * 42 + a6 * 86 + a7 * 23 + a8 * 85 - a9 * 47 + a10 * 34 + a11 * 76 + a12 * 43 - a13 * 44 + a14 * 65 == 19875)
s.add(a1 * 79 + a2 * 62 + a3 * 35 - a4 * 85 + a5 * 33 + a6 * 79 + a7 * 86 + a8 * 14 - a9 * 30 + a10 * 25 + a11 * 11 + a12 * 57 - a13 * 50 - a14 * 9 == 22784)
s.add(a1 * 8 + a2 * 6 + a3 * 64 - a4 * 85 + a5 * 73 + a6 * 29 + a7 * 2 + a8 * 23 - a9 * 36 + a10 * 5 + a11 * 2 + a12 * 47 - a13 * 64 + a14 * 27 == 9710)
s.add(a1 * 67 - a2 * 68 + a3 * 68 - a4 * 51 - a5 * 43 + a6 * 81 + a7 * 22 - a8 * 12 - a9 * 38 + a10 * 75 + a11 * 41 + a12 * 27 - a13 * 52 + a14 * 31 == 13376)
s.add(a1 * 85 + a2 * 63 + a3 * 5 - a4 * 51 + a5 * 44 + a6 * 36 + a7 * 28 + a8 * 15 - a9 * 6 + a10 * 45 + a11 * 31 + a12 * 7 - a13 * 67 + a14 * 78 == 24065)
s.add(a1 * 47 + a2 * 64 + a3 * 66 - a4 * 5 + a5 * 43 + a6 * 112 + a7 * 25 + a8 * 13 - a9 * 35 + a10 * 95 + a11 * 21 + a12 * 43 - a13 * 61 + a14 * 20 == 27687)
s.add(a1 * 89 + a2 * 67 + a3 * 85 - a4 * 25 + a5 * 49 + a6 * 89 + a7 * 23 + a8 * 56 - a9 * 92 + a10 * 14 + a11 * 89 + a12 * 47 - a13 * 61 - a14 * 29 == 29250)
s.add(a1 * 95 + a2 * 34 + a3 * 62 - a4 * 9 - a5 * 43 + a6 * 83 + a7 * 25 + a8 * 12 - a9 * 36 + a10 * 16 + a11 * 51 + a12 * 47 - a13 * 60 - a14 * 24 == 15317)
if s.check():
    print(s.model())
# [a2 = 24,a13 = 88, a6 = 43,a9 = 52,a14 = 33,a5 = 104,a12 = 74,a7 = 28,a1 = 119, a10 = 108, a11 = 88, a8 = 91, a4 = 7, a3 = 10]
```

写将位置换回以及逆异或运算脚本即可得到flag

```python
import hashlib
data=[119,24,10,7,104,43,28,91,52,108,88,74,88,33]
index=[2,1,0,3,4,5,6,7,9,8,10,11,12,13]
flag=[0]*14
for i in range(len(flag)):
    flag[index[i]]=data[i]
for i in range(len(flag)-2,-1,-1):
    flag[i]^=flag[i+1]
flag_str=''.join(chr(i) for i in flag)
print(flag_str)
# U_G07_th3_k3y!
h=hashlib.md5()
h.update(flag_str.encode(encoding='utf-8'))
print(h.hexdigest())
# 58964088b637e50d3a22b9510c1d1ef8
```

[羊城杯 2020]Bytecode

txt文件给了python的字节码，翻译成源码

```
#coding:utf-8
en=[3,37,72,9,6,132]
output=[101,96,23,68,112,42,107,62,96,53,176,179,98,53,67,29,
        41,120,60,106,51,101,178,189,101,48]
print('welcome to GWHT2020')
flag=raw_input('please input your flag:')
str=flag
def func0(): # 验证输入的长度
    a = len(str)
    if a < 38:
        print('lenth wrong!')
def func1(): # 验证输入的前5个字符
    if (((ord(str[0])*2020+ord(str[1]))*2020+ord(str[2]))*2020+ord(str[3]))*2020+ord(str[4])==1182843538814603:
        print('good!continue\xe2\x80\xa6\xe2\x80\xa6')
def func2(): # 验证输入花括号{}内的前26个字符
    x=[]
    k=5
    for i in range(13):
        b=ord(str[k])
        c=ord(str[k+1])
        a11=c^en[i%6]
        a22=b^en[i%6]
        x.append(a11)
        x.append(a22)
        k+=2
    if x==output:
        print('good!continue\xe2\x80\xa6\xe2\x80\xa6')
def func3(): # 验证输入花括号{}内的后6个字符
    l=len(str)
    a1=ord(str[l-7])
    a2=ord(str[l-6])
    a3 = ord(str[l - 5])
    a4 = ord(str[l - 4])
    a5 = ord(str[l - 3])
    a6 = ord(str[l - 2])
    if a1*3+a2*2+a3*5==1003 and a1*4+a2*7+a3*9==2013 and a1+a2*8+a3*2==1109 and a4*3+a5*2+a6*5==671 and a4*4+a5*
7+a6*9==1252 and a4+a5*8+a6*2==644:
        print('congraduation!you get the right flag!')
func0()
func1()
func2()
func3()
```

func1验证输入的前5个字符，写爆破脚本，得到"GWHT{"

```
for i in range(32,127):
    for j in range(32,127):
        for k in range(32, 127):
            for m in range(32, 127):
                for n in range(32, 127):
                    if (((i*2020+j)*2020+k)*2020+m)*2020+n==1182843538814603:
                        print(chr(i)+chr(j)+chr(k)+chr(m)+chr(n))
                        break
#GWHT{
```

func2验证输入花括号{}内的前26个字符，写逆脚本，得到"cfa2b87b3f746a8f0ac5c5963f"

```
en=[3,37,72,9,6,132]
output=[101,96,23,68,112,42,107,62,96,53,176,179,98,53,67,29,
        41,120,60,106,51,101,178,189,101,48]
k=0
flag=[]
for i in range(13):
    c1=output[k+1]^en[i%6]
    c2=output[k]^en[i%6]
    flag.append(c1)
    flag.append(c2)
    k+=2
print(''.join(chr(i) for i in flag))
# cfa2b87b3f746a8f0ac5c5963f
```

func3验证输入花括号{}内的后6个字符，用z3解方程，转成字符串，得到"aeff73"

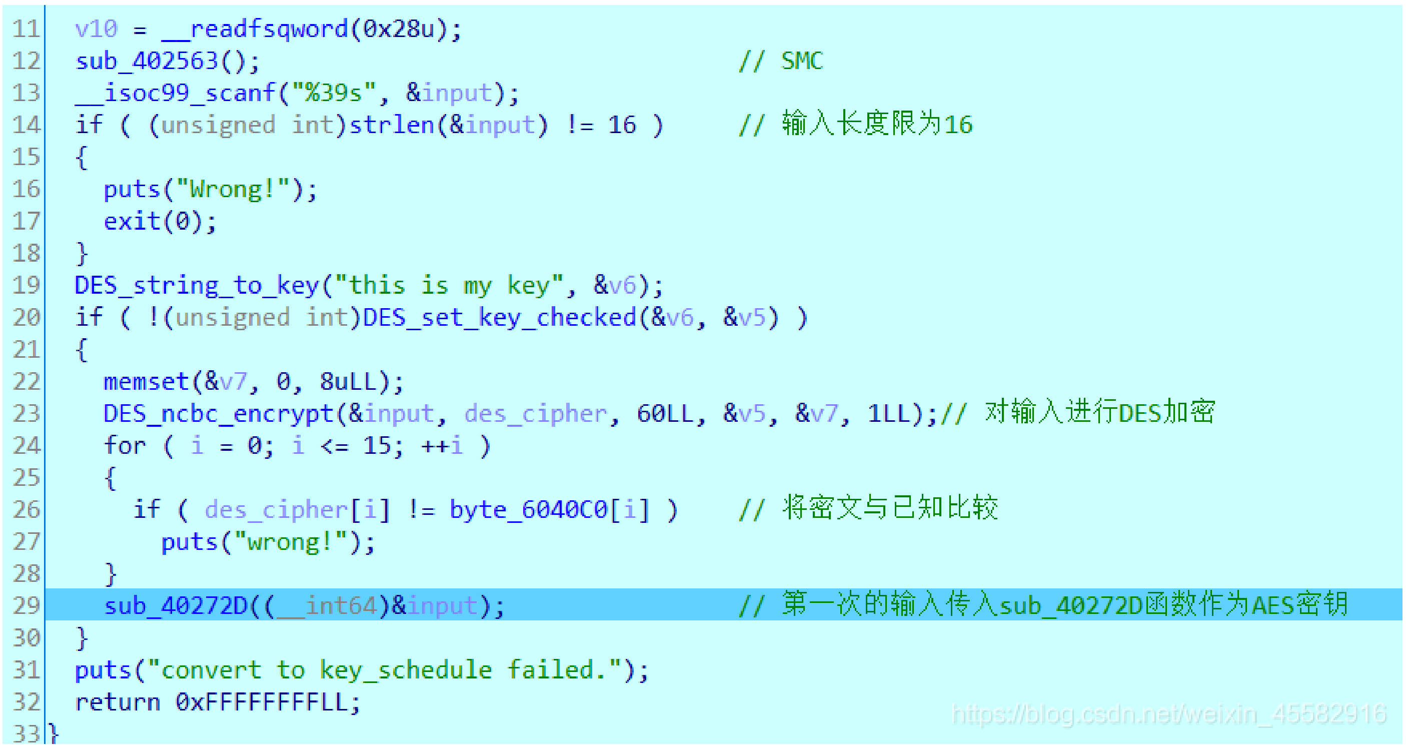
```
from z3 import *
a1=Int('a1')
a2=Int('a2')
a3=Int('a3')
a4=Int('a4')
a5=Int('a5')
a6=Int('a6')
s=Solver()
s.add(a1*3+a2*2+a3*5==1003)
s.add(a1*4+a2*7+a3*9==2013)
s.add(a1+a2*8+a3*2==1109)
s.add(a4*3+a5*2+a6*5==671)
s.add(a4*4+a5*7+a6*9==1252)
s.add(a4+a5*8+a6*2==644)
if s.check():
    print(s.model())
# [a5 = 55, a2 = 101, a6 = 51, a3 = 102, a4 = 102, a1 = 97]
data=[97,101,102,102,55,51]
print(''.join(chr(i) for i in data))
# aeff73
```

最后加上一个'}'，于是flag为`"GWHT{cfa2b87b3f746a8f0ac5c5963faeff73}"`

[羊城杯 2020]babyre

elf文件，无壳，ida分析

main函数，首先sub_402563函数进行一段SMC，获取输入，输入长度限为16，对输入进行DES加密，密钥动态调试可得，比较DES加密过的输入(密文)与已知的byte_6040C0，验证成功后，将未经DES加密过的输入传入sub_40272D函数作为AES加密的密钥

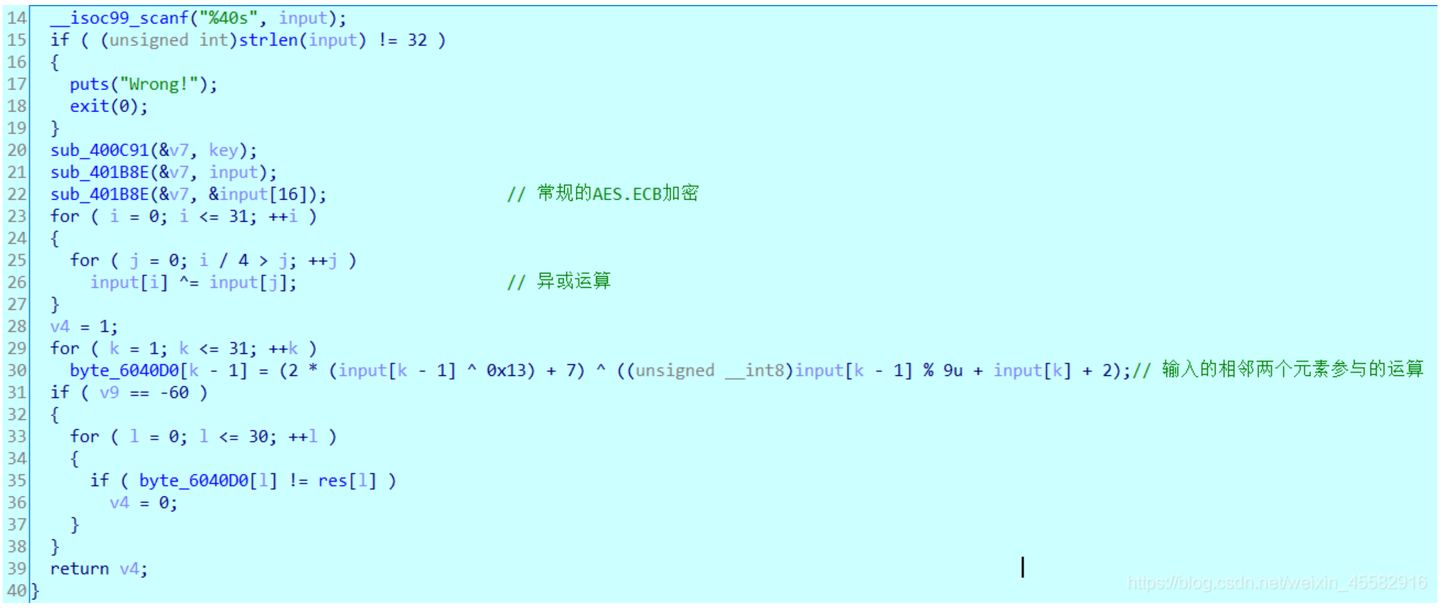


```
  v10 = __readfsqword(0x28u);
  sub_402563();                              // SMC
  __isoc99_scanf("%39s", &input);
  if ( (unsigned int)strlen(&input) != 16 )  // 输入长度限为16
  {
    puts("Wrong!");
    exit(0);
  }
  DES_string_to_key("this is my key", &v6);
  if ( !(unsigned int)DES_set_key_checked(&v6, &v5) )
  {
    memset(&v7, 0, 8uLL);
    DES_ncbc_encrypt(&input, des_cipher, 60LL, &v5, &v7, 1LL);// 对输入进行DES加密
    for ( i = 0; i <= 15; ++i )
    {
      if ( des_cipher[i] != byte_6040C0[i] )   // 将密文与已知比较
        puts("wrong!");
    }
    sub_40272D((__int64)&input);               // 第一次的输入传入sub_40272D函数作为AES密钥
  }
  puts("convert to key_schedule failed.");
  return 0xFFFFFFFFLL;
}
```

调试得到DES密钥为`b'\xAD\x52\xF2\x4C\xE3\x2C\x20\xD6'`，密文为`b'\x0A\xF4\xEE\xC8\x42\x8A\x9B\xDB\xA2\x26\x6F\xEE\xEE\xE0\xD8\xA2'`，分别用ECB模式和CBC模式解DES，两次解密结果的拼接即为第一次正确的输入

```python
from Crypto.Cipher import DES
key=b'\xAD\x52\xF2\x4C\xE3\x2C\x20\xD6'
des_ecb=DES.new(key,DES.MODE_ECB)
des_cbc=DES.new(key,DES.MODE_CBC,key)
cipher=b'\x0A\xF4\xEE\xC8\x42\x8A\x9B\xDB\xA2\x26\x6F\xEE\xEE\xE0\xD8\xA2'
m1=des_ecb.decrypt(cipher)
m2=des_cbc.decrypt(cipher)
print(m1)
print(m2)
#th1s1sth9�■q
#�:�?�_T�3n1c3k3y
#th1s1sth3n1c3k3y
```

进入sub_40272D函数，获取输入，第一次的输入作为AES的密钥，对输入进行常规的AES.ECB加密，密文异或运算，然后还有个相邻两个元素参与的运算给byte_6040D0赋值，最后byte_6040D0与已知的res比较

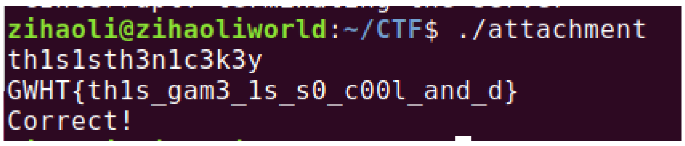


```
  __isoc99_scanf("%40s", input);
  if ( (unsigned int)strlen(input) != 32 )
  {
    puts("Wrong!");
    exit(0);
  }
  sub_400C91(&v7, key);
  sub_401B8E(&v7, input);
  sub_401B8E(&v7, &input[16]);                  // 常规的AES.ECB加密
  for ( i = 0; i <= 31; ++i )
  {
    for ( j = 0; i / 4 > j; ++j )
      input[i] ^= input[j];                     // 异或运算
  }
  v4 = 1;
  for ( k = 1; k <= 31; ++k )
    byte_6040D0[k - 1] = (2 * (input[k - 1] ^ 0x13) + 7) ^ ((unsigned __int8)input[k - 1] % 9u + input[k] + 2);// 输入的相邻两个元素参与的运算
  if ( v9 == -60 )
  {
    for ( l = 0; l <= 30; ++l )
    {
      if ( byte_6040D0[l] != res[l] )
        v4 = 0;
    }
  }
  return v4;
}
```

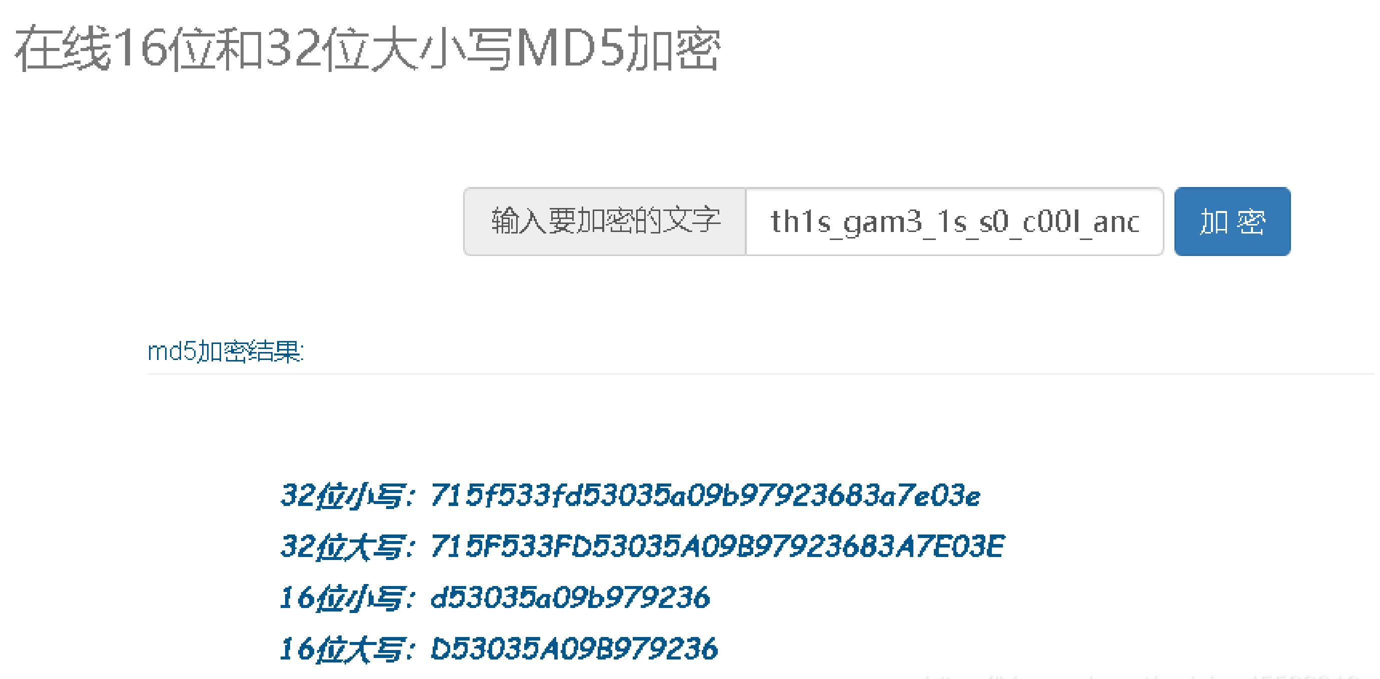
写逆运算脚本即可得到flag

```python
from Crypto.Cipher import AES
key="th1s1sth3n1c3k3y"
aes=AES.new(key,AES.MODE_ECB)
res=[0xBD, 0xAD, 0xB4, 0x84, 0x10, 0x63, 0xB3, 0xE1, 0xC6, 0x84,
  0x2D, 0x6F, 0xBA, 0x88, 0x74, 0xC4, 0x90, 0x32, 0xEA, 0x2E,
  0xC6, 0x28, 0x65, 0x70, 0xC9, 0x75, 0x78, 0xA0, 0x0B, 0x9F,
  0xA6]
for i in range(0,255):
    s=[]
    s.append(i)
    for j in range(1,len(res)+1):
        tmp=((res[j-1]^(2*(s[j-1]^0x13)+7))-2-s[j-1]%9)&0xff
        s.append(tmp)
    for j in range(31,-1,-1):
        for k in range(j//4):
            s[j]^=s[k]
    s_str=''.join(chr(i) for i in s)
    m=aes.decrypt(s_str)
    if 'GWHT' in m:
        print(m)
#GWHT{th1s_gam3_1s_s0_c00l_and_d}
```

两次输入，验证成功，再md5一下，提交成功



```
zihaoli@zihaoliworld:~/CTF$ ./attachment
th1s1sth3n1c3k3y
GWHT{th1s_gam3_1s_s0_c00l_and_d}
Correct!
```



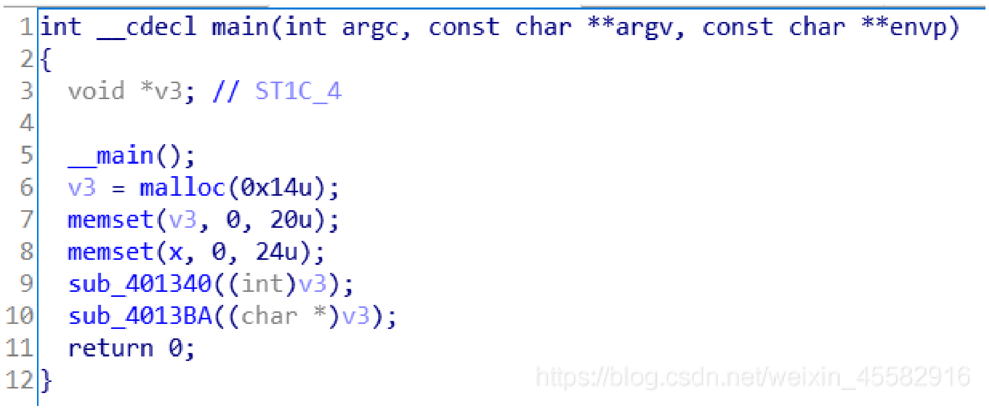
在线16位和32位大小写MD5加密

md5加密结果:

32位小写: 715f533fd53035a09b979236837e03e
32位大写: 715F533FD53035A09B979236837E03E
16位小写: d53035a09b979236
16位大写: D53035A09B979236

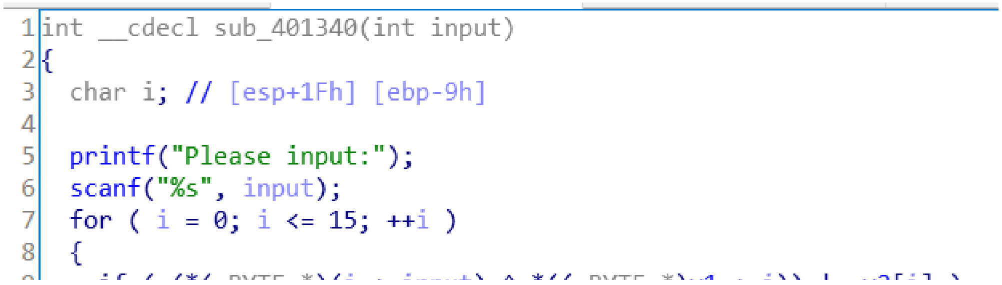
[ACTF新生赛2020]fungame

exe程序，运行后输入，无壳，ida分析
main函数，给v3和x填充0，x大小为36，只填充了24个0

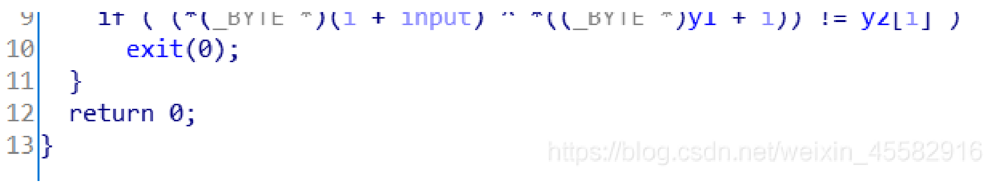


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void *v3; // ST1C_4

  __main();
  v3 = malloc(0x14u);
  memset(v3, 0, 20u);
  memset(x, 0, 24u);
  sub_401340((int)v3);
  sub_4013BA((char *)v3);
  return 0;
}
```

sub_401340函数，获取输入，对输入的前16个字符进行验证

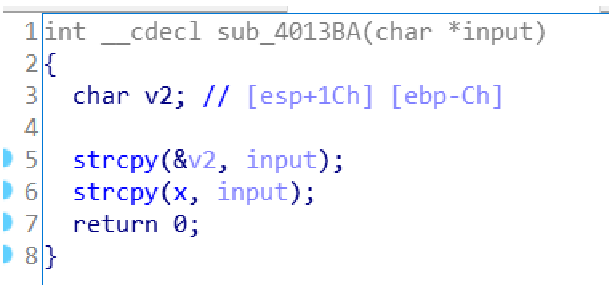


```
int __cdecl sub_401340(int input)
{
  char i; // [esp+1Fh] [ebp-9h]

  printf("Please input:");
  scanf("%s", input);
  for ( i = 0; i <= 15; ++i )
  {
```



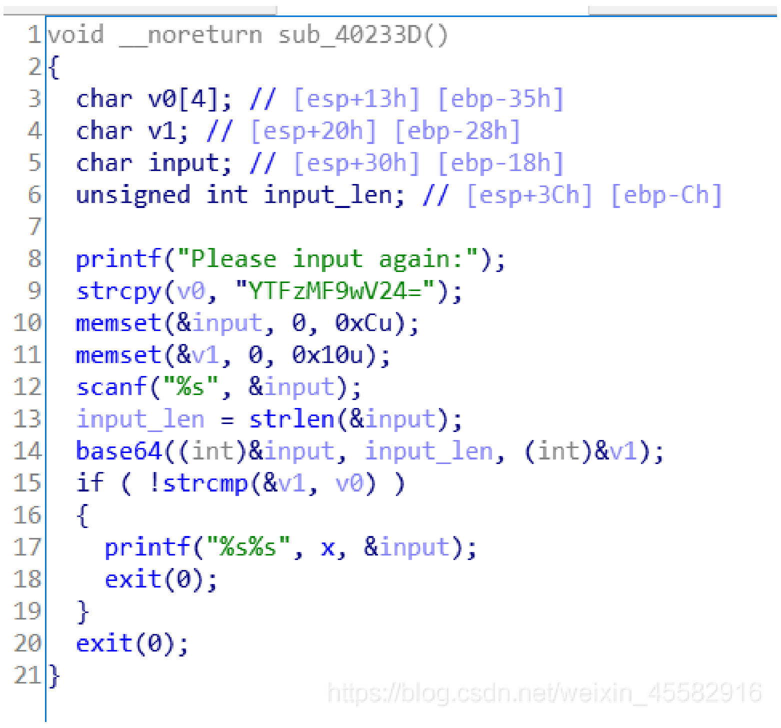
```
    if ( (*(_BYTE *)(i + input) ^ *((_BYTE *)y1 + i)) != y2[i] )
      exit(0);
  }
  return 0;
}
```

sub_4013BA函数，两次copy



```
int __cdecl sub_4013BA(char *input)
{
  char v2; // [esp+1Ch] [ebp-Ch]

  strcpy(&v2, input);
  strcpy(x, input);
  return 0;
}
```

查找x的交叉引用，除了main和sub_4013BA函数，第三处在sub_40233D函数
再次输入，对输入进行常规的base64编码，结果与已知的v0比较验证



```
void __noreturn sub_40233D()
{
  char v0[4]; // [esp+13h] [ebp-35h]
  char v1; // [esp+20h] [ebp-28h]
  char input; // [esp+30h] [ebp-18h]
  unsigned int input_len; // [esp+3Ch] [ebp-Ch]

  printf("Please input again:");
  strcpy(v0, "YTFzMF9wV24=");
  memset(&input, 0, 0xCu);
  memset(&v1, 0, 0x10u);
  scanf("%s", &input);
  input_len = strlen(&input);
  base64((int)&input, input_len, (int)&v1);
  if ( !strcmp(&v1, v0) )
  {
    printf("%s%s", x, &input);
    exit(0);
  }
  exit(0);
}
```

写脚本，但是提交失败

```
import base64
y1=[0x23, 0x61, 0x3E, 0x69, 0x54, 0x41, 0x18, 0x4D, 0x6E, 0x3B,
  0x65, 0x53, 0x30, 0x79, 0x45, 0x5B]
y2=[0x71, 0x04, 0x61, 0x58, 0x27, 0x1E, 0x4B, 0x22, 0x5E, 0x64,
  0x03, 0x26, 0x5E, 0x17, 0x3C, 0x7A]
flag=[]
for i in range(16):
    flag.append(y1[i]^y2[i])
flag_str=''.join(chr(i) for i in flag)
s="YTFzMF9wV24="
flag_str+=base64.b64decode(s)
print(flag_str)
#Re_1s_So0_funny!a1s0_pWn
```

最后参考Mz1师傅的wp：re | [ACTF新生赛2020]fungame