攻防世界 Misc高手进阶区 6分题 pyHAHA

原创

思源湖的鱼 于 2021-01-13 22:57:54 发布 758 收藏 2
分类专栏： ctf 文章标签： ctf 攻防世界 misc pyc

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 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/weixin_44604541/article/details/112468128
版权

ctf 专栏收录该内容
200 篇文章 23 订阅
订阅专栏

前言

继续ctf的旅程
攻防世界Misc高手进阶区的6分题
本篇是pyHAHA的writeup

发现攻防世界的题目分数是动态的
就仅以做题时的分数为准了

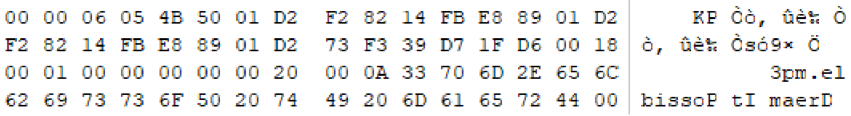
解题过程

拿到一个pyc

反编译
结果显示反编译不成功

猜测是pyc隐写
报错

扔进winhex



```
00 00 06 05 4B 50 01 D2  F2 82 14 FB E8 89 01 D2     KP Òò, ûè‰ Ò
F2 82 14 FB E8 89 01 D2  73 F3 39 D7 1F D6 00 18 ò, ûè‰ Òsó9× Ö
00 01 00 00 00 00 00 20  00 0A 33 70 6D 2E 65 6C            3pm.el
62 69 73 73 6F 50 20 74  49 20 6D 61 65 72 44 00 bissoP tI maerD
```

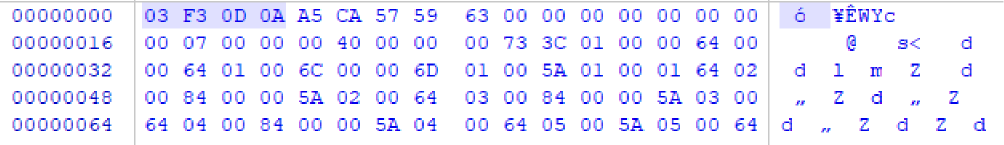
发现有压缩文件
且倒序了

```
f = open('PyHaHa2.pyc','wb')
with open('PyHaHa.pyc','rb') as g:
 f.write(g.read()[::-1])
f.close()
```

正序后

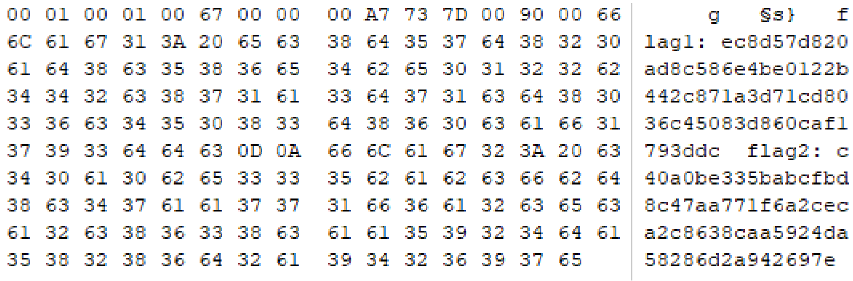
发现少文件头`03F30D0A`

补上



```
00000000  03 F3 0D 0A A5 CA 57 59  63 00 00 00 00 00 00 00   ó  ¥ÊWYc
00000016  00 07 00 00 00 40 00 00  00 73 3C 01 00 00 64 00      @   s<   d
00000032  00 64 01 00 6C 00 00 6D  01 00 5A 01 00 01 64 02   d  l  m  Z   d
00000048  00 84 00 00 5A 02 00 64  03 00 84 00 00 5A 03 00   „  Z  d  „  Z
00000064  64 04 00 84 00 00 5A 04  00 64 05 00 5A 05 00 64  d  „  Z  d  Z  d
```

在末尾发现



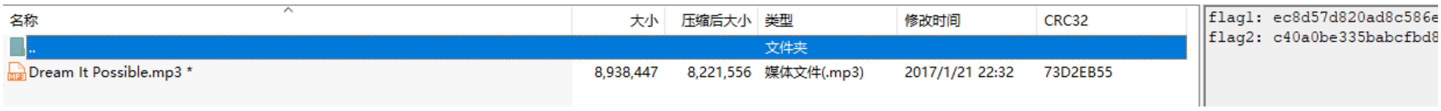
```
00 01 00 01 00 67 00 00  00 A7 73 7D 00 90 00 66       g   §s}   f
6C 61 67 31 3A 20 65 63  38 64 35 37 64 38 32 30  lag1: ec8d57d820
61 64 38 63 35 38 36 65  34 62 65 30 31 32 32 62  ad8c586e4be0122b
34 34 32 63 38 37 31 61  33 64 37 31 63 64 38 30  442c871a3d71cd80
33 36 63 34 35 30 38 33  64 38 36 30 63 61 66 31  36c45083d860caf1
37 39 33 64 64 63 0D 0A  66 6C 61 67 32 3A 20 63  793ddc  flag2: c
34 30 61 30 62 65 33 33  35 62 61 62 63 66 62 64  40a0be335babcfbd
38 63 34 37 61 61 37 37  31 66 36 61 32 63 65 63  8c47aa771f6a2cec
61 32 63 38 36 33 38 63  61 61 35 39 32 34 64 61  a2c8638caa5924da
35 38 32 38 36 64 32 61  39 34 32 36 39 37 65     58286d2a942697e
```

```
flag1: ec8d57d820ad8c586e4be0122b442c871a3d71cd8036c45083d860caf1793ddc
flag2: c40a0be335babcfbd8c47aa771f6a2ceca2c8638caa5924da58286d2a942697e
```

foremost分离

得到一个压缩包

里面有一个mp3



名称	大小	压缩后大小	类型	修改时间	CRC32
..			文件夹		
Dream It Possible.mp3 *	8,938,447	8,221,556	媒体文件(.mp3)	2017/1/21 22:32	73D2EB55

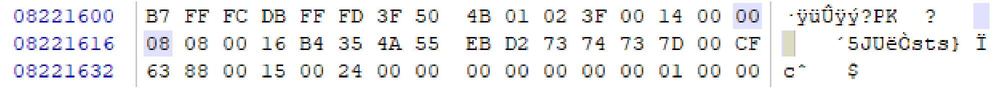
```
flag1: ec8d57d820ad8c586e
flag2: c40a0be335babcfbd8
```

解压要密码

看了下

是伪加密

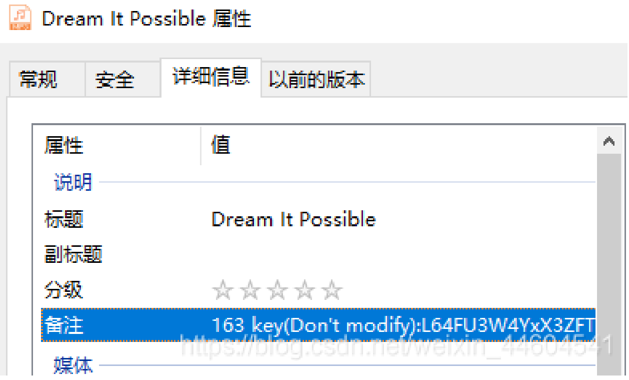
图示部分改为`0008`



```
08221600  B7 FF FC DB FF FD 3F 50  4B 01 02 3F 00 14 00 00  ·ÿüÛÿý?PK  ?
08221616  08 08 00 16 B4 35 4A 55  EB D2 73 74 73 7D 00 CF      ´5JUëÒsts} Ï
08221632  63 88 00 15 00 24 00 00  00 00 00 00 00 01 00 00  cˆ    $
```

解压得到mp3

在备注里有key



Dream It Possible 属性

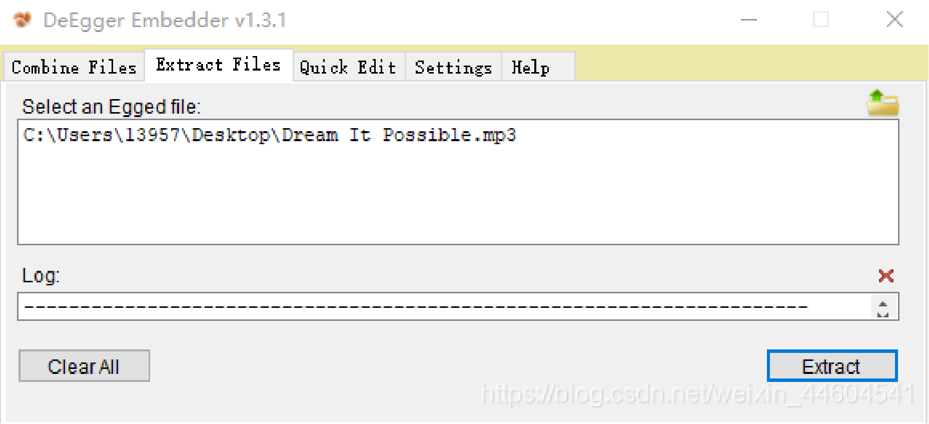
属性	值
说明	
标题	Dream It Possible
副标题	
分级	☆☆☆☆☆
备注	163 key(Don't modify):L64FU3W4YxX3ZFT
媒体	

https://blog.csdn.net/weixin_44604541

```
163 key(Don't modify):L64FU3W4YxX3ZFTmbZ+8/dx5jBNDPdsKv9gAgXYyj0/Z0Vl4ORaCLH5D0oN9v9nBBv6zxpBucgNeE2qqke4ugZs7dx
riT5lfUpulX5PYMzSg2pqL6APTHQjtIHw16ZCRTMBBkInrUGSTklA2MwPLGkuDTmWfzjHqDWEK5LLRP6oiOXe0JeI9mLHzL2nm6T3/ianaPEzeH7
lpE/ciWDyqFUXoTZWeyK0xTjYxBSj9RJFoaoIxmXk6P8MJq+EvcS7ratByRUvWLaUFhcXhJ1iRK58BKcVPS0hxkzA77WB1UnrUfpkS1U/F5uYmYW
GZ3Cz2fTvJHcsE228LUnnybaoIoDyY6BvZVUitmA+VhhTHoIKsREyCzC3VXq/HuwwuHdCdftzXJpXPoJSNzWAZH0oGnF+WE7m5Fqd36dqp2srL5b
gFjFnyZzIkxdaU+ZW+Rm7aIbrb7nK2Pp9iequOBV6rnKeHZyc3hzG4lVVapoXl/U2cvfVgIqVtnuJE4XM4NuUd
```

扔进audacity
无果

查了下
有个工具DeEgger Embedder

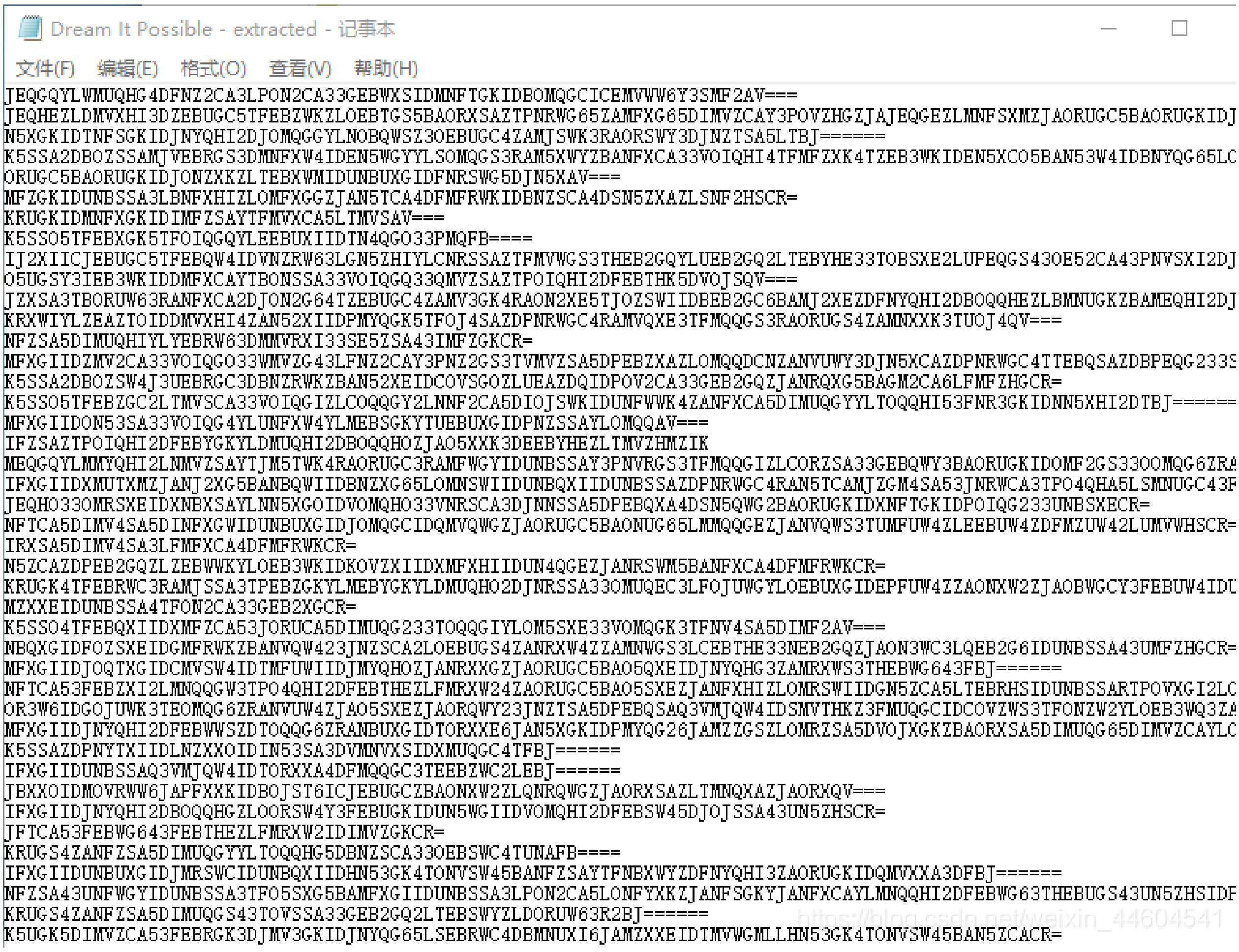


Select an Egged file:

C:\Users\13957\Desktop\Dream It Possible.mp3

Log:

```
--------------------------------------------------------------------------------
```


Dream It Possible - extracted - 记事本
文件(F)
编辑(E)
格式(O)
查看(V)
帮助(H)
https://blog.csdn.net/weixin_44604541

```
JEQGQYLWMUQHG4DFNZ2CA3LPON2CA33GEBWXSIDMNFTGKIDBOMQGCICEMVWW6Y3SMF2AV===
JEQHEZLDMVXHI3DZEBUGC5TFEBZWKZLOEBTGS5BAORXSAZTPNRWG65ZAMFXG65DIMVZCAY3POVZHGZJAJEQGEZLMNFSXMZJAORUGC5BAORUGKIDJ
N5XGKIDTNFSGKIDJNYQHI2DJOMQGGYLNOBQWSZ3OEBUGC4ZAMJSWK3RAORSWY3DJNZTSA5LTBJ======
K5SSA2DBOZSSAMJVEBRGS3DMNFXW4IDEN5WGYYLSOMQGS3RAM5XWYZBANFXCA33VOIQHI4TFMFZXK4TZEB3WKIDEN5XCO5BAN53W4IDBNYQG65LC
ORUGC5BAORUGKIDJONZXKZLTEBXWMIDUNBUXGIDFNRSWG5DJN5XAV===
MFZGKIDUNBSSA3LBNFXHIZLOMFXGGZJAN5TCA4DFMFRWKIDBNZSCA4DSN5ZXAZLSNF2HSCR=
KRUGKIDMNFXGKIDIMFZSAYTFMVXCA5LTMVSAV===
K5SSO5TFEBXGK5TFOIQGQYLEEBUXIIDTN4QGO33PMQFB====
IJ2XIICJEBUGC5TFEBQW4IDVNZRW63LGN5ZHIYLCNRSSAZTFMVWGS3THEB2GQYLUEB2GQ2LTEBYHE33TOBSXE2LUPEQGS43OE52CA43PNVSXI2DJ
O5UGSY3IEB3WKIDDMFXCAYTBONSSA33VOIQGQ33QMVZSAZTPOIQHI2DFEBTHK5DVOJSQV===
JZXSA3TBORUW63RANFXCA2DJON2G64TZEBUGC4ZAMV3GK4RAON2XE5TJOZSWIIDBEB2GC6BAMJ2XEZDFNYQHI2DBOQQHEZLBMNUGKZBAMEQHI2DJ
KRXWIYLZEAZTOIDDMVXHI4ZAN52XIIDPMYQGK5TFOJ4SAZDPNRWGC4RAMVQXE3TFMQQGS3RAORUGS4ZAMNXXK3TUOJ4QV===
NFZSA5DIMUQHIYLYEBRW63DMMVRXI33SE5ZSA43IMFZGKCR=
MFXGIIDZMV2CA33VOIQGO33WMVZG43LFNZ2CAY3PNZ2GS3TVMVZSA5DPEBZXAZLOMQQDCNZANVUWY3DJN5XCAZDPNRWGC4TTEBQSAZDBPEQG233S
K5SSA2DBOZSW4J3UEBRGC3DBNZRWKZBAN52XEIDCOVSGOZLUEAZDQIDPOV2CA33GEB2GQZJANRQXG5BAGM2CA6LFMFZHGCR=
K5SSO5TFEBZGC2LTMVSCA33VOIQGIZLCOQQGY2LNNF2CA5DIOJSWKIDUNFWWK4ZANFXCA5DIMUQGYYLTOQQHI53FNR3GKIDNN5XHI2DTBJ======
MFXGIIDON53SA33VOIQG4YLUNFXW4YLMEBSGKYTUEBUXGIDPNZSSAYLOMQQAV===
IFZSAZTPOIQHI2DFEBYGKYLDMUQHI2DBOQQHOZJAO5XXK3DEEBYHEZLTMVZHMZIK
MEQGQYLMMYQHI2LNMVZSAYTJM5TWK4RAORUGC3RAMFWGYIDUNBSSAY3PNVRGS3TFMQQGIZLCORZSA33GEBQWY3BAORUGKIDOMF2GS33OOMQG6ZRA
IFXGIIDXMUTXMZJANJ2XG5BANBQWIIDBNZXG65LOMNSWIIDUNBQXIIDUNBSSAZDPNRWGC4RAN5TCAMJZGM4SA53JNRWCA3TPO4QHA5LSMNUGC43F
JEQHO33OMRSXEIDXNBXSAYLNN5XGOIDVOMQHO33VNRSCA3DJNNSSA5DPEBQXA4DSN5QWG2BAORUGKIDXNFTGKIDPOIQG233UNBSXECR=
NFTCA5DIMV4SA5DINFXGWIDUNBUXGIDJOMQGCIDQMVQWGZJAORUGC5BAONUG65LMMQQGEZJANVQWS3TUMFUW4ZLEEBUW4ZDFMZUW42LUMVWHSCR=
IRXSA5DIMV4SA3LFMFXCA4DFMFRWKCR=
N5ZCAZDPEB2GQZLZEBWWKYLOEB3WKIDKOVZXIIDXMFXHIIDUN4QGEZJANRSWM5BANFXCA4DFMFRWKCR=
KRUGK4TFEBRWC3RAMJSSA3TPEBZGKYLMEBYGKYLDMUQHO2DJNRSSA33OMUQEC3LFOJUWGYLOEBUXGIDEPFUW4ZZAONXW2ZJAOBWGCY3FEBUW4IDU
MZXXEIDUNBSSA4TFON2CA33GEB2XGCR=
K5SSO4TFEBQXIIDXMFZCA53JORUCA5DIMUQG233TOQQGIYLOM5SXE33VOMQGK3TFNV4SA5DIMF2AV===
NBQXGIDFOZSXEIDGMFRWKZBANVQW423JNZSCA2LOEBUGS4ZANRXW4ZZAMNWGS3LCEBTHE33NEB2GQZJAON3WC3LQEB2G6IDUNBSSA43UMFZHGCR=
MFXGIIDJOQTXGIDCMVSW4IDTMFUWIIDJMYQHOZJANRXXGZJAORUGC5BAO5QXEIDJNYQHG3ZAMRXWS3THEBWG643FBJ======
NFTCA53FEBZXI2LMNQQGW3TPO4QHI2DFEBTHEZLFMRXW24ZAORUGC5BAO5SXEZJANFXHIZLOMRSWIIDGN5ZCA5LTEBRHSIDUNBSSARTPOVXGI2LC
OR3W6IDGOJUWK3TEOMQG6ZRANVUW4ZJAO5SXEZJAORQWY23JNZTSA5DPEBQSAQ3VMJQW4IDSMVTHKZ3FMUQGCIDCOVZWS3TFONZW2YLOEB3WQ3ZA
MFXGIIDJNYQHI2DFEBWWSZDTOQQG6ZRANBUXGIDTORXXE6JAN5XGKIDPMYQG26JAMZZGSZLOMRZSA5DVOJXGKZBAORXSA5DIMUQG65DIMVZCAYLC
K5SSAZDPNYTXIIDLNZXXOIDIN53SA3DVMNVXSIDXMUQGC4TFBJ======
IFXGIIDUNBSSAQ3VMJQW4IDTORXXA4DFMQQGC3TEEBZWC2LEBJ======
JBXXOIDMOVRWW6JAPFXXKIDBOJST6ICJEBUGCZBAONXW2ZLQNRQWGZJAORXSAZLTMNQXAZJAORXQV===
IFXGIIDJNYQHI2DBOQQHGZLOORSW4Y3FEBUGKIDUN5WGIIDVOMQHI2DFEBSW45DJOJSSA43UN5ZHSCR=
JFTCA53FEBWG643FEBTHEZLFMRXW2IDIMVZGKCR=
KRUGS4ZANFZSA5DIMUQGYYLTOQQHG5DBNZSCA33OEBSWC4TUNAFB====
IFXGIIDUNBUXGIDJMRSWCIDUNBQXIIDHN53GK4TONVSW45BANFZSAYTFNBXWYZDFNYQHI3ZAORUGKIDQMVXXA3DFBJ======
NFZSA43UNFWGYIDUNBSSA3TFO5SXG5BAMFXGIIDUNBSSA3LPON2CA5LONFYXKZJANFSGKYJANFXCAYLMNQQHI2DFEBWG63THEBUGS43UN5ZHSIDF
KRUGS4ZANFZSA5DIMUQGS43TOVSSA33GEB2GQ2LTEBSWYZLDORUW63R2BJ======
K5UGK5DIMVZCA53FEBRGK3DJMV3GKIDJNYQG65LSEBRWC4DBMNUXI6JAMZXXEIDTMVWGMLLHN53GK4TONVSW45BAN5ZCACR=
```

得到一堆base32

回头看看那个处理好的pyc
反编译得到

```python
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
from os import urandom

def generate(m, k):
    result = 0
    for i in bin(m ^ k)[2:]:
        result = result << 1
        if int(i):
            result = result ^ m ^ k
        if result >> 256:
            result = result ^ P
            continue
    return result


def encrypt(seed):
    key = int(urandom(32).encode('hex'), 16)
    while True:
        yield key
        key = generate(key, seed) + 0x3653C01D55L


def convert(string):
    return int(string.encode('hex'), 16)

P = 0x10000000000000000000000000000000000000000000000000000000000000425L
flag1 = 'ThIs_Fl4g_Is_Ri9ht'
flag2 = 'Hey_Fl4g_Is_Not_HeRe'
key = int(urandom(32).encode('hex'), 16)
data = open('data.txt', 'r').read()
result = encrypt(key)
encrypt1 = bin(int(data, 2) ^ eval('0x' + hex(result.next())[2:-1] * 22))[2:]
encrypt2 = hex(convert(flag1) ^ result.next())[2:-1]
encrypt3 = hex(convert(flag2) ^ result.next())[2:-1]
print 'flag1:', encrypt2
print 'flag2:', encrypt3
f = open('encrypt.txt', 'w')
f.write(encrypt1)
f.close()
```

- encrypt实现的是一个256bit随机数生成器的功能
- generate实现的是在有限域GF(2^{256})下的平方运算：new_key=$(\text{old\_key+seed})^2$
- flag1和flag2的密文在前面的zip注释信息已给出
- 脚本对三段明文使用了同个Seed做了加密，其中后两段明文和密文还有第一段的密文（在那大段的base32里）已知

考虑OTP加密

- 先由后两段明文和密文算出 key2 和 key3，再在 GF(2^{256})下进行开方即可得到 seed，key3 = $(\text{key2+seed})^2$
- 再由第一段密文(即 base32 隐藏的数据)key1 和 seed 解得 key1，Key2= $(\text{key1+seed})^2$
- 最后对第一段密文(即 base32 隐藏的数据)和 22 次叠加的 key1 做异或得到原始二进制数据

那现在的问题就在于搞到base32的密文
再解密就行了

但是base32这里
直接解码得到的东西不对

猜测隐写

尝试发现base32解码再编码后尾部不同
是有隐写了

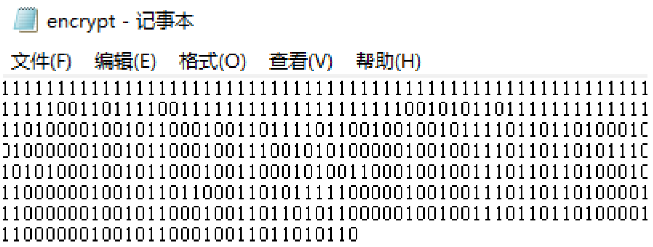
```python
import base64

def get_base32_diff_value(stego_line, normal_line):
    base32chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
    for i in range(len(normal_line)):
        if stego_line[i] != normal_line[i]:
            return abs(base32chars.index(chr(stego_line[i]))-base32chars.index(chr(normal_line[i])))
    return 0

# base32 隐写解密
def base32stego_decode(lines):
    res = ''
    for i in lines:
        stego_line = i.strip()
        normal_line = base64.b32encode(base64.b32decode(i.strip()))
        diff = get_base32_diff_value(stego_line, normal_line)
        if '=' not in str(stego_line):
            continue
        if diff:
            res += bin(diff)[2:]
        else:
            res += '0'
    return res

with open("Dream It Possible - extracted.txt", 'rb') as f:
    file_lines = f.readlines()
en=open("encrypt.txt","w")
en.write(base32stego_decode(file_lines))
en.close()
```

得到encrypt



```
1111111111111111111111111111111111111111111111111111111111
1111100110111100111111111111111111110010101101111111111111
1101000010010110001001101111011001001001011110110110100010
0100000010010110001001110010101000001001001110110110101110
1010100010010110001001100010100110001001001110110110100010
1100000010010110110001101011111000001001001110110110100001
1100000010010110001001101101011000001001001110110110100001
11000000100101100010011011010110
```

最后解密就行了

```
from os import urandom

def generate(m, k):
    result = 0
    for i in bin(m ^ k)[2:]:
        result = result << 1
        if int(i):
            result = result ^ m ^ k
        if result >> 256:
            result = result ^ P
            continue
    return result

def convert(string):
    return int(string.encode('hex'), 16)


P = 0x10000000000000000000000000000000000000000000000000000000000000425L
flag1 = 'ThIs_Fl4g_Is_Ri9ht'
flag2 = 'Hey_Fl4g_Is_Not_HeRe'
encrypt1 = open('encrypt.txt', 'r').read()
encrypt2 = 0xec8d57d820ad8c586e4be0122b442c871a3d71cd8036c45083d860caf1793ddc
encrypt3 = 0xc40a0be335babcfbd8c47aa771f6a2ceca2c8638caa5924da58286d2a942697e
key3 = encrypt3 ^ convert(flag2)
key2 = encrypt2 ^ convert(flag1)
print('Found key2:',key2)
print('Found key3:',key3)

tmp = key3 - 233333333333L
for i in range(0,255):
    tmp = generate(tmp,0)
seed = tmp ^ key2
print 'Found seed:',seed)
print 'use seed generate key3:',generate(key2,seed)+233333333333L

tmp = key2 - 233333333333L
for i in range(0,255):
    tmp = generate(tmp,0)
key1 = tmp ^ seed
print 'Found key1:',key1
print 'use key1 generate key2:',generate(key1,seed)+233333333333L

result = eval(hex(int(encrypt1,2))[:-1]) ^ eval('0x'+hex(key1)[2:-1]*22)
data = open('data.txt', 'w')
data.write(bin(result)[2:])
data.close()
```

得到一堆二进制

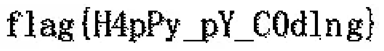
转ascii
乱码

查了查wp
说是图片

。。。

```
from PIL import Image

str=open("data.txt","r").read()
length=240
width=30
pic=Image.new("RGB",(length,width))
i=0
for x in range(length):
 for y in range(width):
  if str[i] == '0':
   pic.putpixel([x,y],(0,0,0))
  else:
   pic.putpixel([x,y],(255,255,255))
  i += 1
pic.show()
pic.save("Fl4g.png")
```



flag{H4pPy_pY_C0dlng}

得到flag

结语

蛮有意思的一题
最后还是查了wp

知识点

- pyc隐写
- pyc反编译
- zip伪加密
- DeEgger Embedder
- base32隐写
- OTP加密
- 二值图像