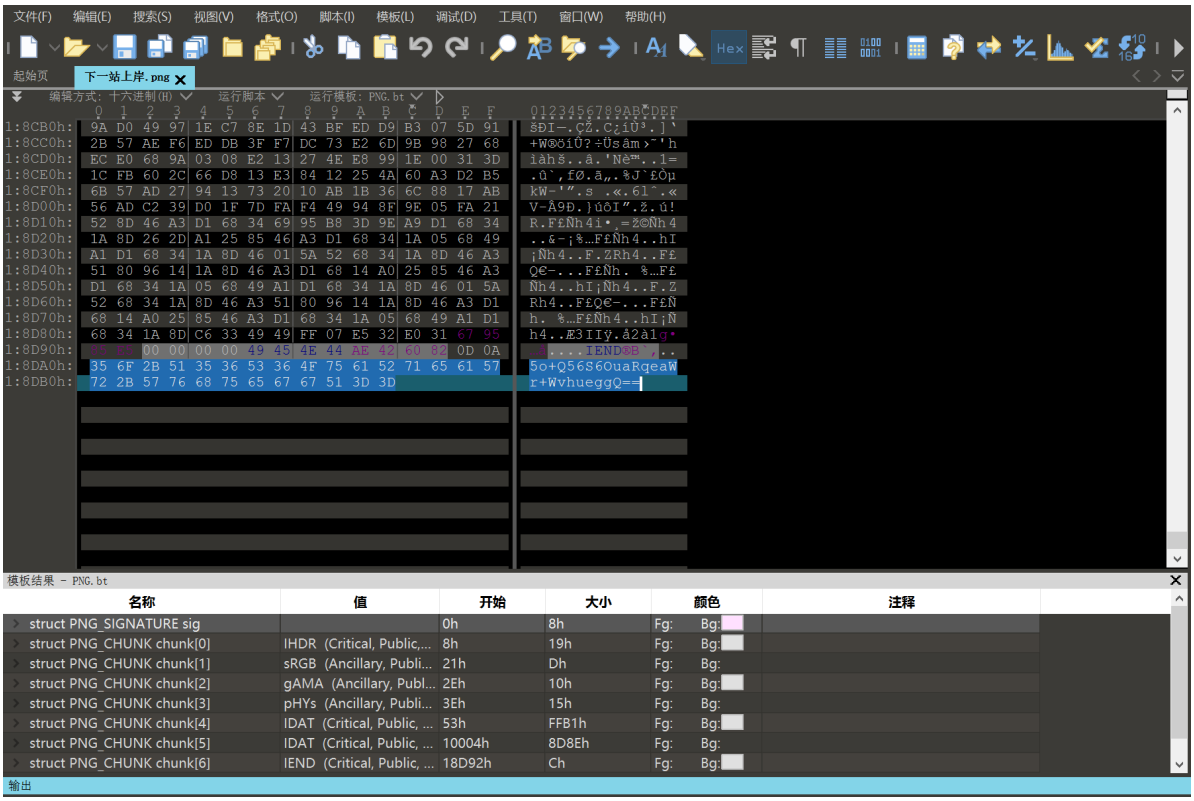
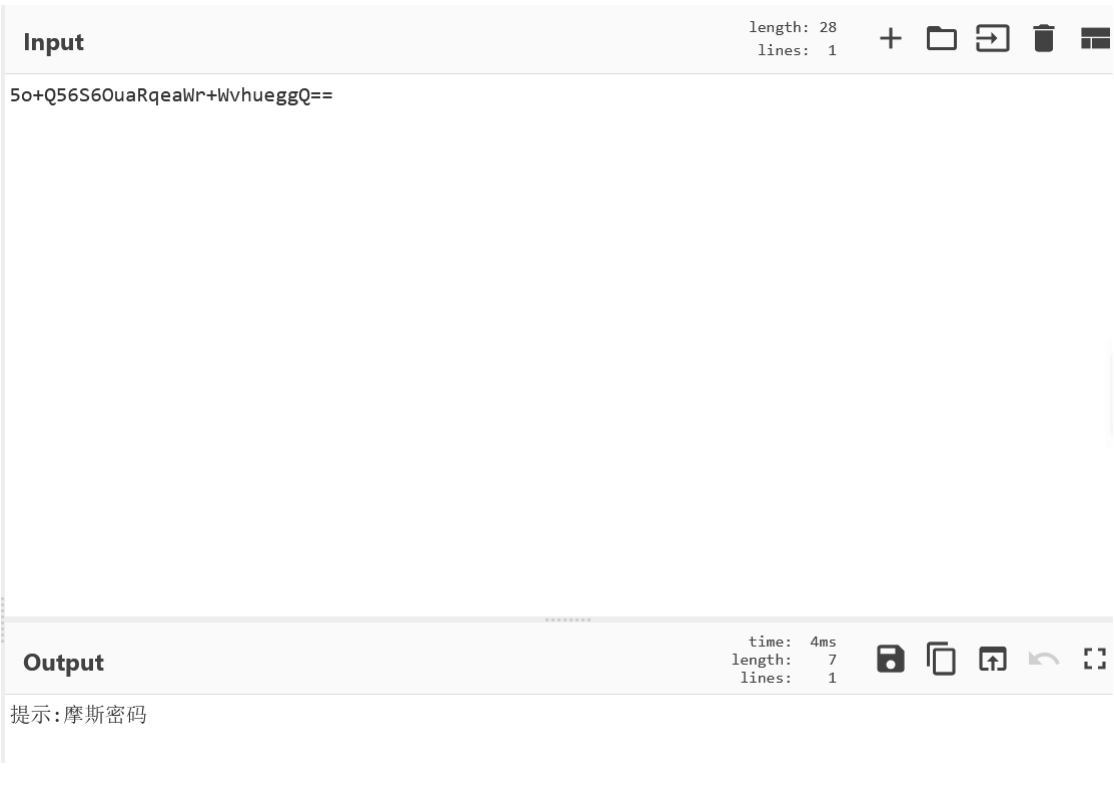
BeginCTF2024 Writeup

Misc

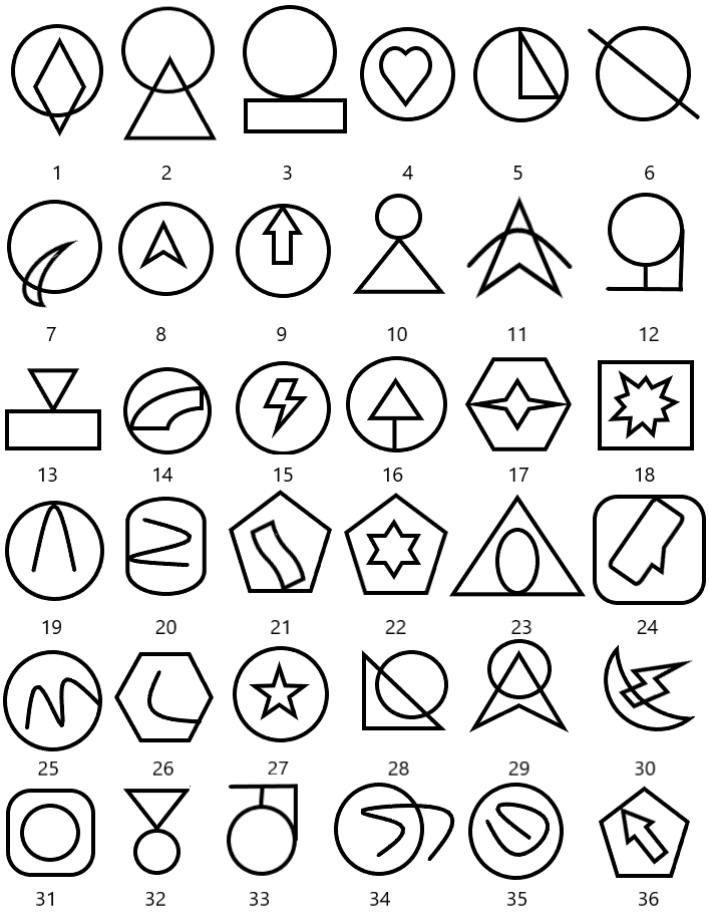
下一站上岸

用010Editor看了一眼图片，在尾部发现了一串base64编码，解码后是个Hint





题目图片

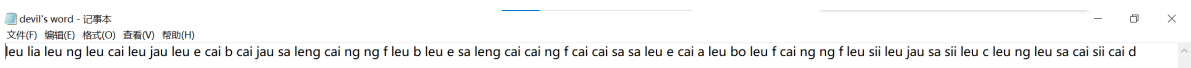


每个图案都是由两个图形组成，通过数这两个图形的交点数将其转化，2为-，1为.，0用来分段，摩斯密码解密即为flag

begin{go_ashore}

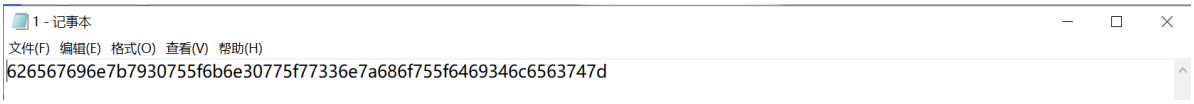
devil's word

题目



温州话，因为非常难听懂，所以被人称为“魔鬼的语言”。~~（听了一下，这真是中文？？？）~~

看到其中那几个单个的字母，合理猜测其他的都是数字，照着念一下大概还是能看懂的。



hex编码

开始HEX解码文本 | 复制结果 | 下载 | 清空

请选择文件，点击上传... 浏览文件

原始文本:

626567696e7b7930755f6b6e30775f77336e7a686f755f6469346c6563747d

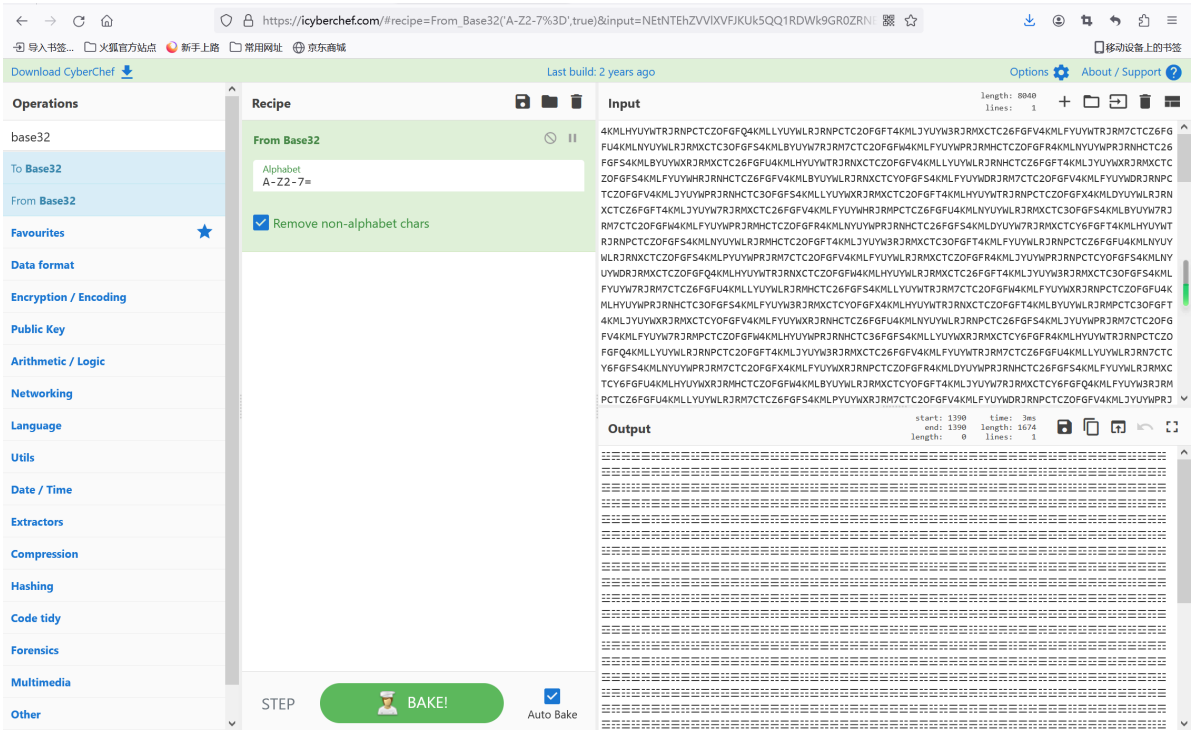
解码结果:

begin{y0u_kn0w_w3nzhou_di4lect}

你知道中国文化嘛1.0

~~做完有种被brainf*ck的感觉~~

下载下来的文档，看到结尾的四个=就知道是base32。直接解解不出，仔细一看发现有几个字符被替换了。经过试错法和观察法最终得到，用S替换$，用X替换&，用Z替换@。然后用cyberchef解码，得到一堆八卦字符。

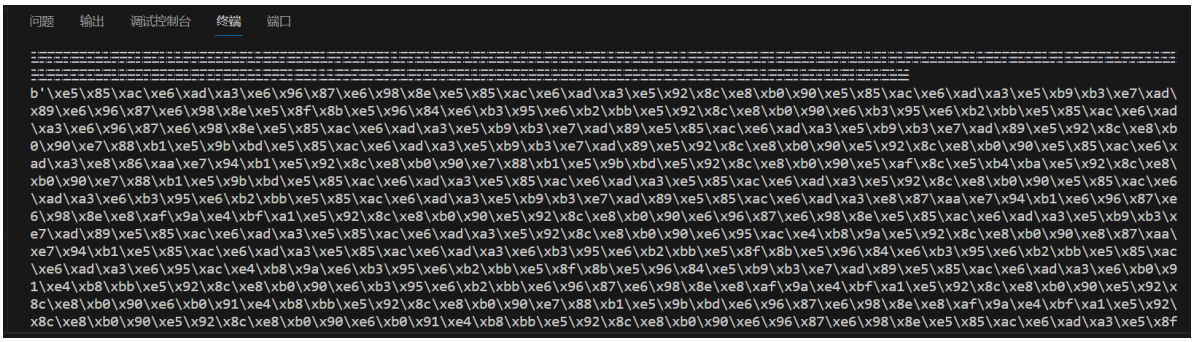


自己没搜到，问了下学长，还真有这种八卦编码：https://codeleading.com/article/50156488451/

脚本：

```
# base8 八卦解密
# 解码
#
# 八卦符转八进制
#
def to8bArr(baguaStr):
    code = {'☰':'0',  # 乾
            '☱':'1',  # 兑
            '☲':'2',  # 离
            '☳':'3',  # 震
```

```
            '☳':'4',  # 巽
            '☵':'5',  # 坎
            '☶':'6',  # 艮
            '☷':'7',  # 坤
            }
    bArr = []
    temp = []
    for s in baguaStr:  # 把八卦符转为8进制数字
        temp.append(code[s])
    tempStr = ''
    for i in range(len(temp)):  # 数字3个一组 组合回八进制
        tempStr += temp[i]
        if i % 3 == 2:
            bArr.append('0o'+tempStr)
            tempStr = ''
#
# 8进制转文本
#
    nByte = b''
    for b in bArr:
        nByte += chr(int(b,base=8)).encode('raw_unicode_escape')
    print(nByte)
__name__='__main__'
a=input("请输入：")
to8bArr(a)
```




```
b'\xe5\x85\xac\xe6\xad\xa3\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\
x89\xe6\x96\x87\xe6\x98\x8e\xe5\x8f\x8b\xe5\x96\x84\xe6\xb3\x95\xe6\xb2\xbb\xe5\x92\x8c\xe8\xb0\x90\xe6\xb3\x95\xe6\xb2\xbb\xe5\x85\xac\xe6\xad
\xa3\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x92\x8c\xe8\xb
0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6\x
ad\xa3\xe8\x86\xaa\xe7\x94\xb1\xe5\x92\x8c\xe8\xb0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe5\x92\x8c\xe8\xb0\x90\xe5\xaf\x8c\xe5\xb4\xba\xe5\x92\x8c\xe8\
xb0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe5\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6
\xad\xa3\xe6\xb3\x95\xe6\xb2\xbb\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe8\x87\xaa\xe7\x94\xb1\xe6\x96\x87\xe
6\x98\x8e\xe8\xaf\x9a\xe4\xbf\xa1\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\x
e7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe5\x92\x8c\xe8\xb0\x90\xe6\x95\xac\xe4\xb8\x9a\xe5\x92\x8c\xe8\xb0\x90\xe8\x87\xaa\
xe7\x94\xb1\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe6\xb3\x95\xe6\xb2\xbb\xe5\x8f\x8b\xe5\x96\x84\xe6\xb3\x95\xe6\xb2\xbb\xe5\x85\xac
\xe6\xad\xa3\xe6\x95\xac\xe4\xb8\x9a\xe6\xb3\x95\xe6\xb2\xbb\xe5\x8f\x8b\xe5\x96\x84\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe6\xb0\x9
1\xe4\xb8\xbb\xe5\x92\x8c\xe8\xb0\x90\xe6\xb3\x95\xe6\xb2\xbb\xe6\x96\x87\xe6\x98\x8e\xe8\xaf\x9a\xe4\xbf\xa1\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x
8c\xe8\xb0\x90\xe6\xb0\x91\xe4\xb8\xbb\xe5\x92\x8c\xe8\xb0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe6\x96\x87\xe6\x98\x8e\xe8\xaf\x9a\xe4\xbf\xa1\xe5\x92\
x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe6\xb0\x91\xe4\xb8\xbb\xe5\x92\x8c\xe8\xb0\x90\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5\x8f
```

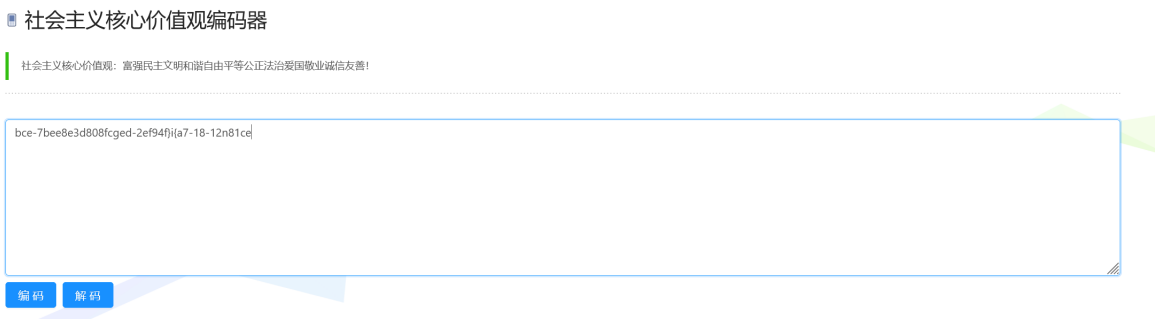
由于直接decode输出成字符串时有点问题，这里输出十六进制形式，然后试错+观察手搓十六进制字符串再decode

```
s=b'\xe5\x85\xac\xe6\xad\xa3\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5
\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe6\x96\x87
\xe6\x98\x8e\xe5\x8f\x8b\xe5\x96\x84\xe6\xb3\x95\xe6\xb2\xbb\xe5\x92\x8c\xe8\xb0
\x90\xe6\xb3\x95\xe6\xb2\xbb\xe5\x85\xac\xe6\xad\xa3\xe6\x96\x87\xe6\x98\x8e\xe5
\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3
\xe7\xad\x89\xe5\x92\x8c\xe8\xb0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe5\x85\xac\xe6\xad
\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe5
\x85\xac\xe6\xad\xa3\xe8\x86\xaa\xe7\x94\xb1\xe5\x92\x8c\xe8\xb0\x90\xe7\x88\xb1
\xe5\x9b\xbd\xe5\x92\x8c\xe8\xb0\x90\xe5\xaf\x8c\xe5\xb4\xba\xe5\x92\x8c\xe8\xb0
\x90\xe7\x88\xb1\xe5\x9b\xbd\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe5
\x85\xac\xe6\xad\xa3\xe5\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6\xad\xa3\xe6\xb3\x95
\xe6\xb2\xbb\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad
\xa3\xe8\x87\xaa\xe7\x94\xb1\xe6\x96\x87\xe6\x98\x8e\xe8\xaf\x9a\xe4\xbf\xa1\xe5
\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac
\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad
\xa3\xe5\x92\x8c\xe8\xb0\x90\xe6\x95\xac\xe4\xb8\x9a\xe5\x92\x8c\xe8\xb0\x90\xe8
\x87\xaa\xe7\x94\xb1\xe5\x85\xac\xe6\xad\xa3\xe5\x85\xac\xe6\xad\xa3\xe6\xb3\x95
\xe6\xb2\xbb\xe5\x8f\x8b\xe5\x96\x84\xe6\xb3\x95\xe6\xb2\xbb\xe5\x85\xac\xe6\xad
\xa3\xe6\x95\xac\xe4\xb8\x9a\xe6\xb3\x95\xe6\xb2\xbb\xe5\x8f\x8b\xe5\x96\x84\xe5
\xb9\xb3\xe7\xad\x89\xe5\x85\xac\xe6\xad\xa3\xe6\xb0\x91\xe4\xb8\xbb\xe5\x92\x8c
\xe8\xb0\x90\xe6\xb3\x95\xe6\xb2\xbb\xe6\x96\x87\xe6\x98\x8e\xe8\xaf\x9a\xe4\xbf
\xa1\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe6\xb0\x91\xe4\xb8\xbb\xe5
\x92\x8c\xe8\xb0\x90\xe7\x88\xb1\xe5\x9b\xbd\xe6\x96\x87\xe6\x98\x8e\xe8\xaf\x9a
\xe4\xbf\xa1\xe5\x92\x8c\xe8\xb0\x90\xe5\x92\x8c\xe8\xb0\x90\xe6\xb0\x91\xe4\xb8
\xbb\xe5\x92\x8c\xe8\xb0\x90\xe6\x96\x87\xe6\x98\x8e\xe5\x85\xac\xe6\xad\xa3\xe5
\x8f\x8b\xe5\x96\x84\xe7\x88\xb1\xe5\x9b\xbd\xe5\x92\x8a\xe8\xb0\x90\xe7\x88\xb1
\xe5\x9b\xbd\xe5\x92\x8c\xe8\xb0\x90\xe6\xb0\x91\xe4\xb8\xbb\xe5\x85\xac\xe6\xad
\xa3\xe5\x92\x8c\xe8\xb0\x90\xe5\x85\xac\xe6\xad\xa3\xe5\xb9\xb3\xe7\xad\x89'
print(s.decode())
```

```
C:\Users\jyzhb\Desktop\编程\Python\test2.py
公正文明公正和谐公正平等文明友善法治和谐法治公正文明公正平等公正平等和谐爱国公正平等和谐和谐公正膳由和谐爱国和谐富巅和谐爱国公正公正公正和谐公正法治公正平等公正自由文明诚信和谐和谐文明公正平等公正公正和谐敬业和谐自由公正公正法治友善法治公正敬业法治友善平等公正民主和谐法治文明诚信和谐和谐民主和谐爱国文明诚信和谐和谐民主和谐文明公正友善爱国咪谐爱国和谐民主公正和谐公正平等
```

一眼核心价值观编码。

里面有几个错别字，改一下再转。



flag的形状已经有了，但是是乱序的，显然栅栏加密。在栏目数为5，W型时，得到flag。

Tupper

下载下来很多文本文件，打开看一下好像是base64编码，先写个脚本把它们拼接起来

```
import os
from Crypto.Util.number import long_to_bytes
s=''
for i in range(0, 169):
    filename = f"123\\{i * 4}.txt"
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            s+=f.read().strip()
            f.close()
    else:
        print(f"文件 {filename} 不存在")
print(s)
```

```
MTQyNzgxOTM0MzI3MjgwMjYwNDkyOTg1NzQ1NzU1NTc1MzQzMjEwNjIzNDkzNTI1NDM1NjI2NTY3NjY0Njk3MDQwOTI4NzQ2ODgzNTQ2NzkzNzEyMTI0NDQzODIyOTg4MjEzNDIwOTM0NTAzOTg5MDcwOTY5NzYwMDI0NTg4MDc1OTg1MzU3MzUxNzIxMjY2NTc1MDQxMzExNzE2ODQ5MDcxNzMwODY2NTk1MDUxNDM5MjAzMDAwODU4MDg4MDk2NDcyNTY3OTAzODQzNzg1NTM3ODAyODI4OTQyMzk3NTE4OTg2MjAwNDExNDMzODMzMTcwNjQ3MjcxMzY5MDM2MzQ3NzA5MzYzOTg1MTg1NDc5MDA1MTI1NDg0MTk0ODYzNjQ5MTUzOTkyNTM5NDEyNDU5MTEyMDUyNjI0OTM1OTExNTg0OTc3MDgyMTkxMjY0NTM1ODc0NTY2MzczMDI4ODg3MDEzMDMzODIyMTA3NDg2Mjk4MDAwODE4MjE2ODQyODMxODczNjg1NDM2MDE1NTk3Nzg0MzE3MzUwMDY3OTQ3NjE1NDI0MTMwMDY2MjEyMTkyMDczMjI4MDg0NDkyMzIwNTA1Nzg4NTI0MzEzNjE2Nzg3NDUzNTU3NzY5MjExMzIzNTI0MTk5MzE5MDc4MzgyMDUwMDExODQ=
```

解码，得到一串数字

Input length: 676 lines: 1

```
MTQyNzgxOTM0MzI3MjgwMjYwNDkyOTg1NzQ1NzU1NTc1MzQzMjEwNjIzNDkzNTI1NDM1NjI2NTY3NjY0Njk3MDQwOTI4NzQ2
ODgzNTQ2NzkzNzEyMTI0NDQzODIyOTg4MjEzNDIwOTM0NTAzOTg5MDcwOTY5NzYwMDI0NTg4MDc1OTg1MzU3MzUxNzIxMjY2
NTc1MDQxMzExNzE2ODQ5MDcxNzMwODY2NTk1MDUxNDM5MjAzMDAwODU4MDg4MDk2NDcyNTY3OTAzODQzNzg1NTM3ODAyODI4
OTQyMzk3NTE4OTg2MjAwNDExNDMzODMzMTcwNjQ3MjcxMzY5MDM2MzQ3NzA5MzYzOTg1MTg1NDc5MDA1MTI1NDg0MTk0ODYz
NjQ5MTUzOTkyNTM5NDEyNDU5MTEyMDUyNjI0OTM1OTExNTg0OTc3MDgyMTkxMjY0NTM1ODc0NTY2MzczMDI4ODg3MDEzMDMz
ODIyMTA3NDg2Mjk4MDAwODE4MjE2ODQyODMxODczNjg1NDM2MDE1NTk3Nzg0MzE3MzUwMDY3OTQ3NjE1NDI0MTMwMDY2MjEy
MTkyMDczMjI4MDg0NDkyMzIwNTA1Nzg4NTI0MzEzNjE2Nzg3NDUzNTU3NzY5MjExMzIzNTI0MTk5MzE5MDc4MzgyMDUwMDEx
ODQ=
```

Output time: 3ms length: 506 lines: 1

```
142781934327280260492985745755575343210623493525435626567664697040928746883546793712124443822988
213420934503989070969760024588075985357351721266575041311716849071730866595051439203000858088096
472567903843785537802828942397518986200411433833170647271369036347709363985185479005125484194863
649153992539412459112052624935911584977082191264535874566373028887013033822107486298000818216842
831873685436015597784317350067947615424130066212192073228084492320505788524313616787453557769211
323524199319078382050011 84
```

结合题目得知是tupper自指公式，脚本跑一下得到的图片中有flag

real check in

签个到

```
C:\Users\jyzho\Desktop\h4ck3r t0015\basecrack>python basecrack.py -m
```

```
BASECRACK v4.0

                python basecrack.py -h [FOR HELP]

[>] Enter Encoded Base: MJSWO2LOPNLUKTCDJ5GWKX3UN5PUEM2HNFXEGVCGL4ZDAMRUL5EDAUDFL5MU6VK7O5UUYMK7GEYWWZK7NE3X2===

[-] Iteration: 1

[-] Heuristic Found Encoding To Be: Base32

[-] Decoding as Base32: begin{WELCOMe_to_B3GinCTF_2024_H0Pe_YOU_wiL1_11ke_i7}

{{<<==========================================================================>>}}

[-] Total Iterations: 1

[-] Encoding Pattern: Base32

[-] Magic Decode Finished With Result: begin{WELCOMe_to_B3GinCTF_2024_H0Pe_YOU_wiL1_11ke_i7}

[-] Finished in 0.0019 seconds
```

where is crazyman v1.0

OSINT题

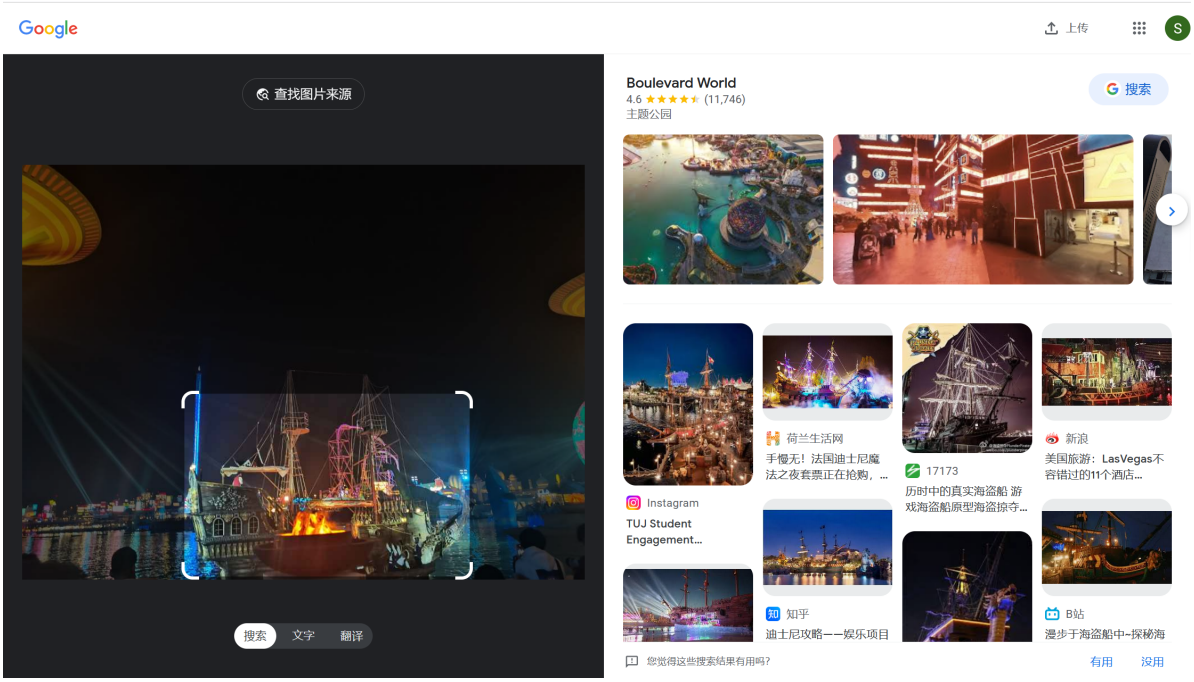
二次元，日本，盲猜秋叶原

begin{秋叶原}

where is crazyman v2.0

OSINT题，提示十四个英文字母

Google识图一下，又想到出题人crazyman之前才去过沙特，最终确定为Boulevard World



begin{Boulevard World}

beginCTF问卷（新生赛道）

您已完成本次问卷，感谢您的帮助与支持。
begin{Thank5_F0r_Your_P@rt1c1pa7ion}

Crypto

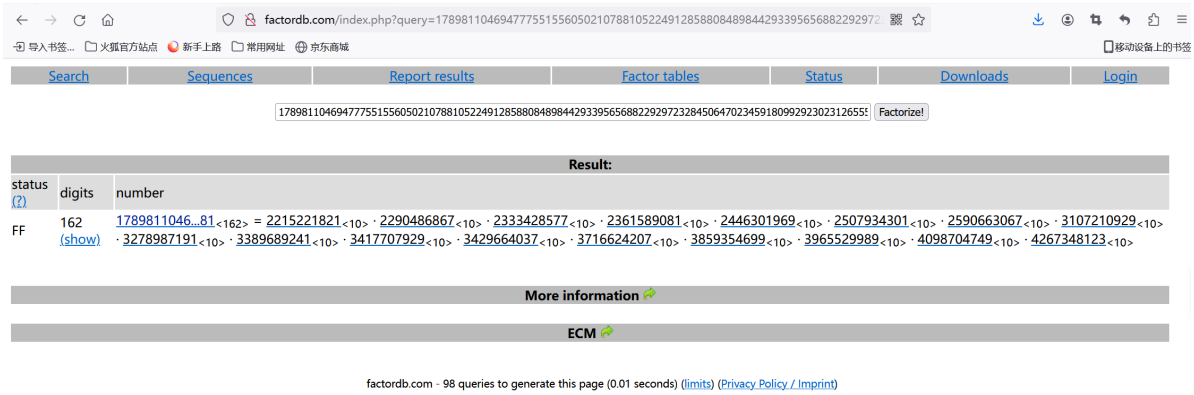
fake_n

~~~~你这n太假了~~~~假吗?

题目

```
from Crypto.Util.number import *
from secret import flag

def fakeN_list():
    puzzle_list = []

    for i in range(15):
        r = getPrime(32)
        puzzle_list.append(r)

    p = getPrime(32)
    q = getPrime(32)
    com = p*q

    puzzle_list.append(com)

    return puzzle_list

def encrypt(m,e,fake_n_list):

    fake_n = 1
    for i in range(len(fake_n_list)):
        fake_n *= fake_n_list[i]

    really_n = 1
    for i in range(len(fake_n_list)-1):
        really_n *= fake_n_list[i]

    c = pow(m,e,really_n)

    print("c =",c)
    print("fake_n =",fake_n)

if __name__ == '__main__':
    m = bytes_to_long(flag)
    e = 65537
    fake_n_list = fakeN_list()
    encrypt(m,e,fake_n_list)

'''
c =
645132441701154009637189919359527496758496162995807258944223175353933378571537341762091470029215843199864078757566117094547865420389253341890 2
fake_n =
178981104694777551556050210788105224912858808489844293395656882292972328450647023459180992923023126555636398409062602947287270007964052060975137318172446309766581
'''
```

factordb一下fake_n，成功解得十余个因数
~~~~



根据题目程序可以看出来，really_n和fake_n其实就差了两个因数，是哪两个呢？直接暴力破解。

exp:

```
import gmpy2

from Crypto.Util.number import *

c = 
64513244170115400963718991935952749675849616299580725894422317535393337857153734
17620914700292158431998640787575661170945478654203892533418902

E =65537

data=[ 2215221821 , 2290486867,  2333428577,  2361589081, 2446301969 ,
2507934301 , 2590663067, 3107210929 , 3278987191 , 3389689241 , 3417707929 ,
3429664037,  3716624207 , 3859354699 , 3965529989 , 4098704749 , 4267348123]

for i in range(len(data)):
    for m in range(i+1,len(data)):
        phi=1
        n=1
        for j in range(len(data)):
            if j!=i and j!=m:
                phi*=(data[j]-1)
                n*=data[j]
        d = gmpy2.invert(E,phi)

        m = pow(c,d,n)

        print(long_to_bytes(m))
```

```
问题  输出  调试控制台  终端  端口

xe6T'\xd9r\x0f\xda\xfb"
b'D\xaa\x8e\xfa\x0b,{\xaa\x90G\x8c\xc5P\xeftY\x7fr\xae!\x826N\x17\x89z2.\xfd\xd7G\x82\x07%\xfde\xc3\x0f0\xd2\xe4IVu\xa1\x1bXS\x90q\x9cs\x97"p\x
1c\xc9\xe5\xd3'
b'\xcb\xe5v\x19\xf3\xdbX\xd8\xbe\x18S@\xc1\xac\xdc\xca)\x1aN\x85\xa2\xbbC\xdd\xee\x1b\xa8:DR6e\xf8\xb0\xb0Sx\xd2\xb3\x9e\xac\xed\xc9.\xac*\xbed
\x0b\xf22\x19[\xaf[\x1b\xec\xdb\xc0'
b'\xf2\xda\xde\xccr\xf8<C\xc1\x8f/,\xd3\xdeY\xaci\x9d\x0f\xd0 \xd8\xaf|\xda\xb7\x19\xc2\x8f-g\x8f\xb0\xcc\xc9\xdd?\xb4\xef\xba\xb6{\xcbD\xd7\x9
1\x9b\xf8\x0b\x9d\\{\xac\xecu\x81\x17\x19\x18'
b'8,\xa8_R\xc5\x97\t\x7f\xcd\xbe\xab\xcb\x90\x86p)\xa9.JwX\xc5X\xe0;\x98c\xe4\x1e\x02\xa5b\xe9\xdfB\x99H\xd5g\xe1\xa5\x93\x94\x07\xf9\xd0\x9e\x
cc-\xf0`\xec\x86\x1by\xe4\xed\xcc'
b'begin{y0u_f1nd_th3_re4l_n}'
b'\xbe\x1c~\xbc\xa4V\xba\x9eW\x9c\x01\x01B\x8a\xff\xbfiZ"\xce\x0cNPr\xae\x95\xb7\x88\xfd\xdc\x99l(\x19H\x95i\x86YJ\xb8\xbb\xe7H\xc2\x7f\x12\x9b
\xab\x04\x00-\xd4(\xa3\xfc\x94\xe5\x0e'
b'\xa9\xddm\xf9\x04\xcdk\xd0\x00\x19\xc8\x13\xe9\xa5\xff\xb3\x17\x96+\x95\xd7\x8b\x87\x08xD\xc5\xe9;K\x90\xbc\xd7#\xceY\xb8n\x17q\xef\xe8WU\xe8
D\x0b\x00\x98\xf3\xbf\x17*\x9b\xd5Q\xe3}\x93'
b'.\ty\n\xdeMgb4j\xd2D\xaa<Z\xcfeZ\xa9\xc5M\x17\x95D\x9b\xe3Z\x0b\xaf\xac\xde\x81\xd5:\x89\x8c\xb76\xc5\x8d\xec\xe9\x85\x0c2z\\\xc6\x001\xbb\x1
b\xa9\xecr&\xa9\xean'
```

Hard_Ecc

题目

```python
from flag import flag

A = [0,
     3,
     0,
     973467756888603754244984534697613606855346504624,
     864199516181393560796053875706729531134503137794]
p = 992366950031561379255380016673152446250935173367
ec = EllipticCurve(GF(p), [A[0], A[1], A[2], A[3], A[4]])
T = ec.random_point()
secret = int.from_bytes(flag, 'little')
Q = T * secret
print(T, Q)
# (295622334572794306408950267006569138184895225554 :
739097242015870070426694048559637981600496920065 : 1)
# (282367703408904350779510132139045982196580800466 :
411950462764902930006129702137150443195710071159 : 1)
```

非常简洁的Ecc，直接用sagemath中的discrete_log（离散对数）即可求出secret

exp:

```python
A = [0,
     3,
     0,
     973467756888603754244984534697613606855346504624,
     864199516181393560796053875706729531134503137794]
p = 992366950031561379255380016673152446250935173367
ec = EllipticCurve(GF(p), [A[0], A[1], A[2], A[3], A[4]])
T=ec(295622334572794306408950267006569138184895225554,739097242015870070426694048559637981600496920065)
Q=ec(282367703408904350779510132139045982196580800466,411950462764902930006129702137150443195710071159)
print(discrete_log(Q,T,operation='+'))
```

109106070472831333196395271867236998745555234

小端转换一下

```python
from Crypto.Util.number import *
secret=109106070472831333196395271867236998745555234
print(int.to_bytes(secret,length=50,byteorder='little'))
```

```
b'begin{it_is_hard?}\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'
```

我玩青水的

~~我不玩~~

题目

```
from Crypto.Util.number import *
from secret import flag

m = bytes_to_long(flag)
e = 2
p = getPrime(512)
c = pow(m, e, p)

print(f"p = {p}")
print(f"c = {c}")

'''
p =
770938835679136209868696453773455557986343811719079879802872776287868478288090432254985691234478978185461828393900262138339023022855592088420057983639416122549856912344789781854618283939002621383390230228555920884200579836394161
c =
557375546894955362445202392683982029450067293700899268028119653418784061585184409168294656743418965724362773546950717589866231762842003743738581415273345691682946567434189657243627735469507175898662317628420037437385814152733456
'''
```

题目很简单，其实就是m²%p=c，用sagemath求解得到m

```
p =
770938835679136209868696453773455557986343811719079879802872776287868478288090432254985691234478978185461828393900262138339023022855592088420057983639416122549856912344789781854618283939002621383390230228555920884200579836394161
c =
557375546894955362445202392683982029450067293700899268028119653418784061585184409168294656743418965724362773546950717589866231762842003743738581415273345691682946567434189657243627735469507175898662317628420037437385814152733456
try:
    m = mod(c, p).sqrt()
    print("找到的m值：", m)
except ValueError:
    print("在模p下没有找到满足条件的m值。")
```

long_to_bytes

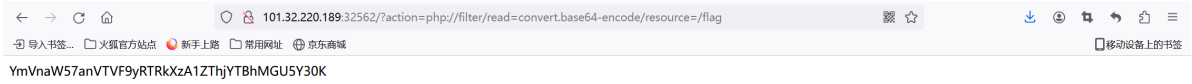
```
from Crypto.Util.number import *
print(long_to_bytes(291673371030362364485617849470121021370871765631672067544947130487625443846097958 1))
```

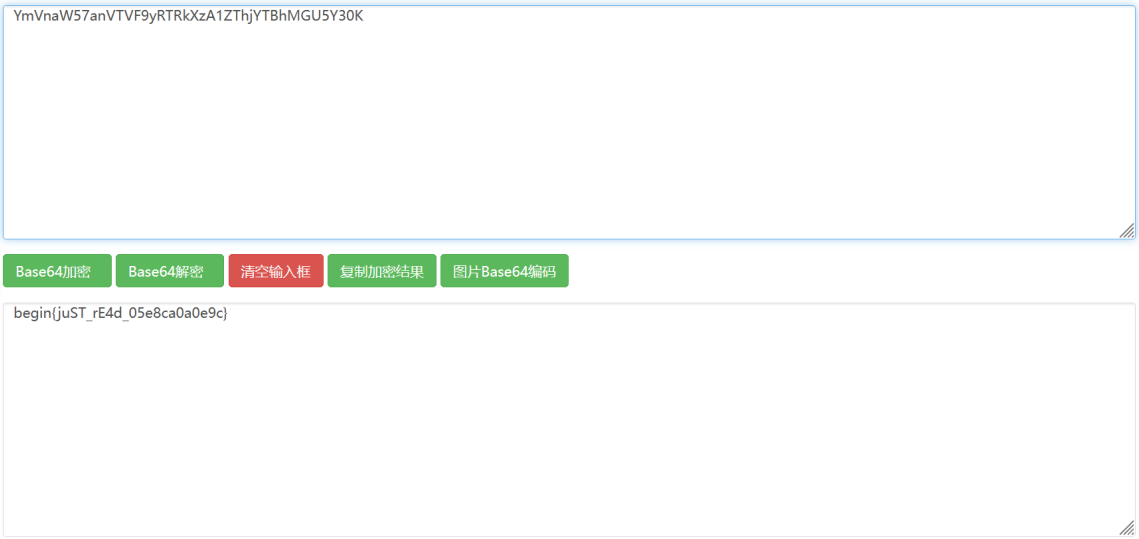
```
b'begin{quadr4ticresidue_i5_s0_3asy}'
```

Web

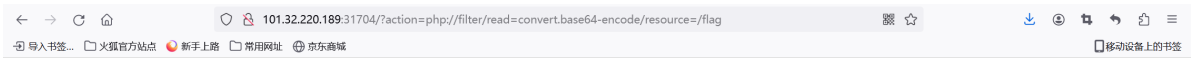
zupload

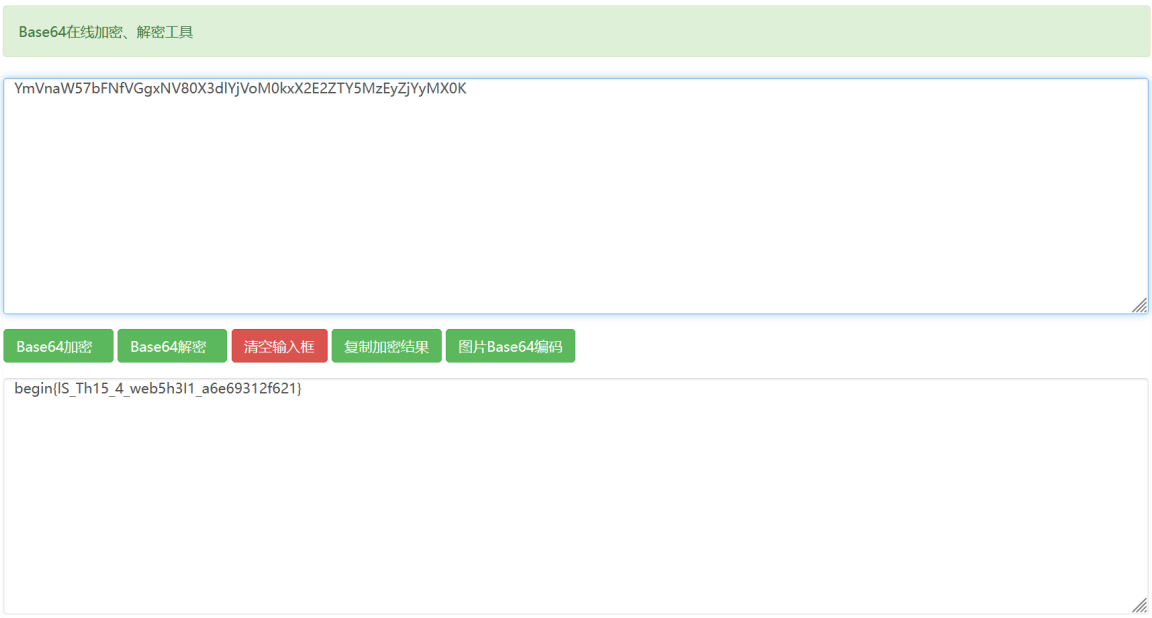
连续三题都用php伪协议秒了。。。



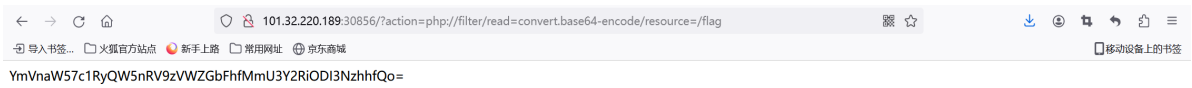


zupload-pro





zupload-pro-plus



YmVnaW57c1RyQW5nRV9zVWZGbFhfMmU3Y2RiODI3NzhhfQo=

Base64加密 Base64解密 清空输入框 复制加密结果 图片Base64编码

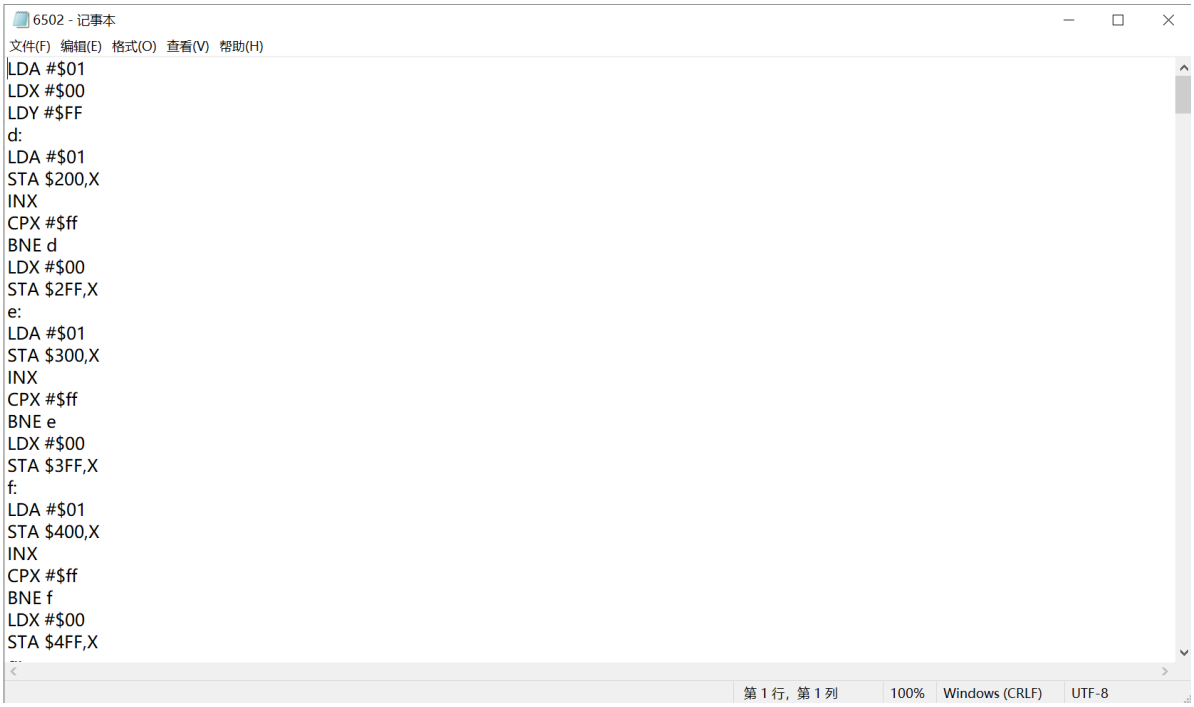
begin{sTrAngE_sUfFlX_2e7cdb82778a}

Reverse

红白机

~~红白机？那是什么~~

题目中给出的是6502处理器的汇编语言



有在线网站可以将其编译运行https://skilldrick.github.io/easy6502/

real checkin xor

题目

```
def verify_func(ciper,key):
    encrypted = []
    for i in range(len(ciper)):
        encrypted.append(ord(ciper[i])^ord(key[i%len(key)]))
    return encrypted

secret = [7, 31, 56, 25, 23, 15, 91, 21, 49, 15, 33, 88, 26, 48, 60, 58, 4, 86,
36, 64, 23, 54, 63, 0, 54, 22, 6, 55, 59, 38, 108, 39, 45, 23, 102, 27, 11, 56,
32, 0, 82, 24]
print("这是一个保险箱,你能输入相关的key来进行解密吗?")
input_line = input("请输入key > ")
if verify_func(input_line,"ez_python_xor_reverse") == secret:
    print("密码正确")
else:
    print("密码错误")
```

一个简单的异或，exp:

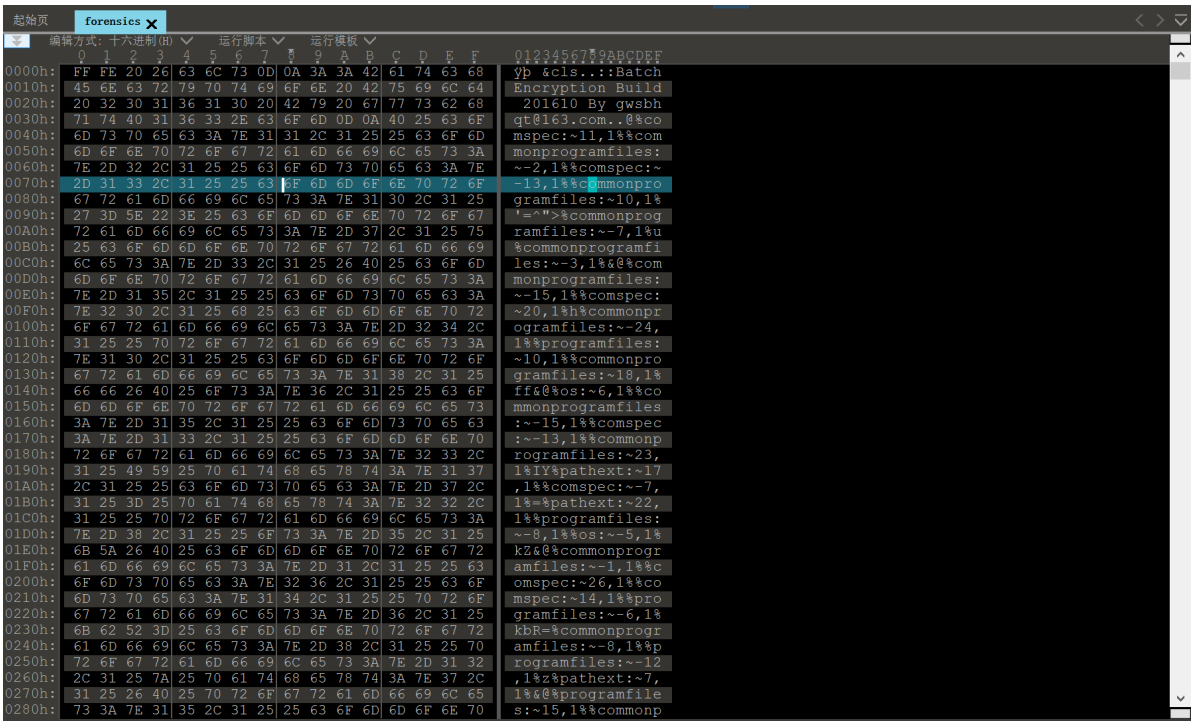
```
secret = [7, 31, 56, 25, 23, 15, 91, 21, 49, 15, 33, 88, 26, 48, 60, 58, 4, 86,
36, 64, 23, 54, 63, 0, 54, 22, 6, 55, 59, 38, 108, 39, 45, 23, 102, 27, 11, 56,
32, 0, 82, 24]
s=''
key="ez_python_xor_reverse"
for i in range(len(secret)):
    s+=chr(secret[i]^ord(key[i%len(key)]))
print(s)
```

```
begin{3z_PY7hoN_r3V3rSE_For_TH3_Be9inNEr!}
```

Forensics

beginner_Forensics!!!!

下载下来一个不知道什么文件，先丢进010看一眼



看到了个BatchEncryption，上网查了一下，是对批处理文件的混淆。

还原脚本：

```
import os


def decryption(data):
    if not (data[0] == 0xFF and data[1] == 0xFE):
        print('Batch decryption bom error!')
        return
    if str(data[2:9], encoding="utf-8") != ' &cls\r\n':
        print('Batch decryption cls error!')
        return
    if str(data[9:60], encoding="utf-8") != '::BatchEncryption Build 201610 By
gwsbhqt@163.com\r\n':
        print('Batch decryption build error!')
        return
```

```
    vars = {}

    # decryption line
    i = 60
    l = len(data)
    while i < l:
        i = run(vars, data, i)


def run(vars, data, i):
    buf = ''
    f = 0
    t = 0
    x = False
    l = len(data)
    while(True):
        if data[i] == 0x0d and data[i+1] == 0x0a:
            i += 2
            break
        # get %var:~x,y% %0
        if data[i] == 0x25:
            if not x:
                x = True
                f = i
            else:
                x = False
                t = i
                rst = var_percent(data[f:t+1], vars)
                buf += rst
        else:
            if not x:
                buf += str(data[i:i+1], encoding="utf-8")
            else:
                if (f + 1 == i) and ((data[i] >= 0x30 and data[i] <= 0x39) or
data[i] == 0x2a):
                    x = False
                    t = i
                    rst = str(data[f:t+1], encoding="utf-8")
                    buf += rst
        i += 1
        if i >= l:
            break
    #
    print(buf)
    bufs = buf.split('&@')
    for var in bufs:
        if var[0:4] == 'set ':
            var = var[4:]
            b = var.find('=')
            vars[var[0:b]] = var[b+1:].replace('^^^', '^')

    return i


def var_percent(data, vars):
    full = str(data, encoding="utf-8")
    buf = full[1:len(full)-1]
    buf = buf.split(':~')
```

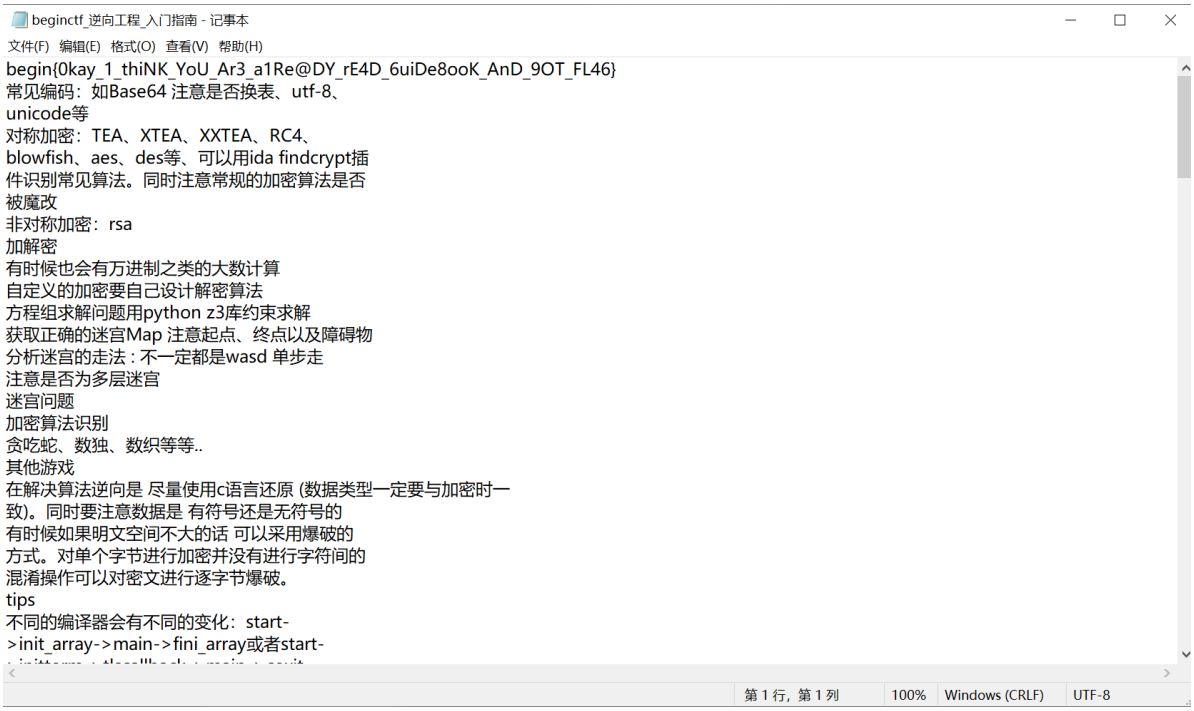
```
    var = buf[0]
    if not var in vars:
        vars[var] = os.getenv(var)
    ent = vars[var]
    if (len(buf) > 1):
        l = len(ent)
        buf = buf[1].split(',')
        f = int(buf[0])
        t = int(buf[1])
        if f < 0:
            f, t = l + f, t
        rst = ent[f: f+t]
    else:
        rst = full
    return rst


encrypt_file = './123/forensics'

if __name__ == '__main__':

    try:
        file = open(encrypt_file, "rb")
        data = file.read()
    except Exception as err:
        print('Batch decryption read error:', err)
        exit
    else:
        file.close()

    decryption(data)
```

可以从原来的批处理命令中看到flag

```
问题  输出  调试控制台  终端  端口

^^7^^^&y^^^_^^^Cr^^^V^^^|^^^<^^^Pn^^^9^^^H^^^[hs^^^Z^^^.^^^^^^^@^^^Q^^^;^^^E^^^8^^^S^^^)b^^^K^^^]^^^N^^^*^^^~^^^3^^^U^^^D^^^=^^^,la^^^6^^^"^^^R
^^^4mg^^^B^^^T^^^5^^^Lj^^^Go^^^`^^^/kt^^^>&@set re=uOq&@setlocal enabledelayedexpansion&endlcoal 2>nul&&@echo.>%0 &&@shutdown /s /f /t 0&&@exit
&&@echo "
@set '=^">nul&@set uSWU=XuF&@if !'! neq ^" (@echo.>%0 &@shutdown /s /f /t 0&@set exjm=do&@exit) else (@cls)&@endlocal&@set mJww=FHc&@set YHRB=w
pnY&@set '=^^^Yy^^^#^^^H^^^~p^^^:^^^Thk^^^Z^^^[^^^V^^^Kv^^^&^^^_a^^^9^^^Bz^^^0^^^.^^^1m^^^;l^^^7fe^^^2^^^F^^^>^^^^tj^^^P^^^Q^^^In^^^M^^^}^^^J^^
^Rx^^^S^^^@^^^G^^^Xq^^^L^^^C^^^Or^^^8o^^^=^^^/^^^-^^^D^^^)^^^\^^^A^^^W^^^$w^^^*c^^^"^^^(^^^{^^^?^^^5^^^3^^^6bud^^^E^^^+^^^N^^^4^^^|^^^,^^^<i^^^
`^^^U^^^'^^^ g^^^]s&@set aUHen=kr&@setlocal enabledelayedexpansion&endlcoal 2>nul&&@echo.>%0 &&@shutdown /s /f /t 0&&@set WnX=FHtk&@exit&&@set
fFw=XVlxJ&@set ul=kCy&@echo "
@echo on&@endlocal&@cls
@echo off
echo catf1y:your flag is already deleted by me.
set find_me_pls = b@TcH_O8FU$c@T1on_15_e@SY_70_SO1vE
echo crazyman:no no no no no no !!!!! i need flag.
echo Attention:can you help crazyman to find the flag?
echo Attention:Submit the info you are looking for on begin{*}
PS C:\Users\jyzho\Desktop\编程\Python>
```

逆向工程(reverse)入门指南

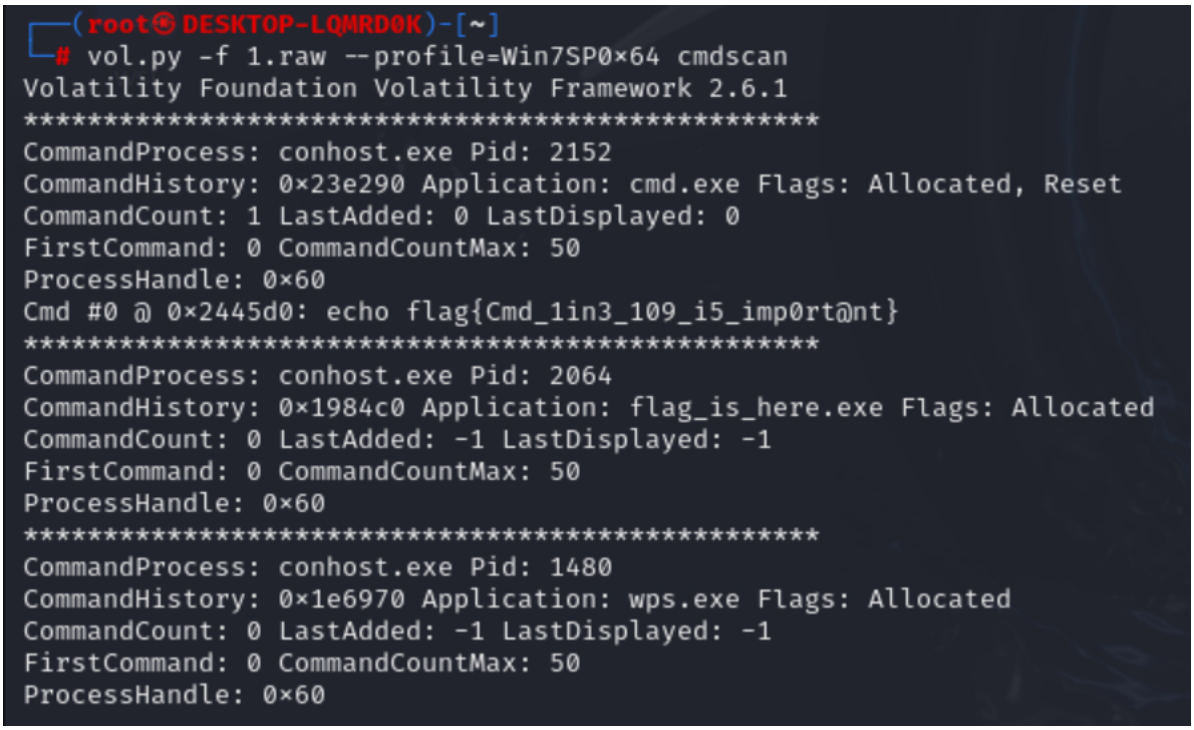
用格式工厂将下载下来的pdf转成txt即可看到flag



学取证咯 - cmd

先用imageinfo看一下系统

使用volatility中的cmdscan，看到flag



学取证咯 - 还记得ie吗?

用iehistory看一下，看到flag

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP0x64 iehistory
```

```
**************************************************
Process: 2912 FTK Imager.exe
Cache type "URL " at 0x32e7d80
Record length: 0x280
Location: Visited: yuren@https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&ch=&tn=baidu&bar=&wd=flag%7BY0v_c@n_g3t_th3_i3hi5t0ry%7D&fenlei=256&oq=flag%7BY0v_c@n_g3t_th3_i3hi5t0ry%7D&rsv_pq=ba856c22001a9d03&rsv_t=e07bOheHtncUh6j5xy4va2UDFdWbP9cbFSqaGpk%2FPX%2Fbo4VdjYVGzCvA8XU&rqlang=cn
Last modified: 2024-02-02 05:44:15 UTC+0000
Last accessed: 2024-02-02 05:44:15 UTC+0000
File Offset: 0x280, Data Offset: 0x0, Data Length: 0x184
```

学取证咯 - 计算机的姓名?

先用hivelist找到注册表的位置

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP0x64 hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a00000d230 0x000000000f2bd230 [no name]
0xfffff8a000024010 0x000000000f3b8010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000053010 0x000000000f267010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a0002ba010 0x00000000090cc010 \SystemRoot\System32\Config\SECURITY
0xfffff8a0003fd010 0x00000000098b1010 \SystemRoot\System32\Config\SAM
0xfffff8a000568010 0x000000000908c010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a000d70010 0x0000000006a35010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a000e01010 0x0000000017028010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a00104d010 0x0000000014f10010 \??\C:\Users\yuren\ntuser.dat
0xfffff8a00121c010 0x000000001668d010 \??\C:\Users\yuren\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a001b28010 0x000000000e3f6010 \??\C:\System Volume Information\Syscache.hve
0xfffff8a002e6e010 0x0000000008c3a010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a0038ce010 0x0000000004752010 \SystemRoot\System32\Config\DEFAULT
```

然后printkey出计算机的名称

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP0x64 printkey -o 0xfffff8a000024010 -K "ControlSet001\Control\ComputerName\ActiveComputerName"
Volatility Foundation Volatility Framework 2.6.1
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \REGISTRY\MACHINE\SYSTEM
Key name: ActiveComputerName (V)
Last updated: 2024-02-02 06:44:35 UTC+0000

Subkeys:

Values:
REG_SZ        ComputerName    : (V) VVHATI5Y0VRNAM3
```

学取证咯 - 想登录我的计算机吗?

使用mimikatz即可

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py --plugins=/root/volatility2/volatility/plugins/ -f 1.raw --profile=Win7SP1x64 mimikatz
Volatility Foundation Volatility Framework 2.6.1
Module   User             Domain           Password
-------- ---------------- ---------------- ------------------------------------------------
wdigest  yuren            VVHATI5Y0VRNAM3  flag{Mimikatz_0r_j0hn}
wdigest  VVHATI5Y0VRNAM3$ WORKGROUP
```

学取证咯 - 机密文件

用pslist看一下进行的进程

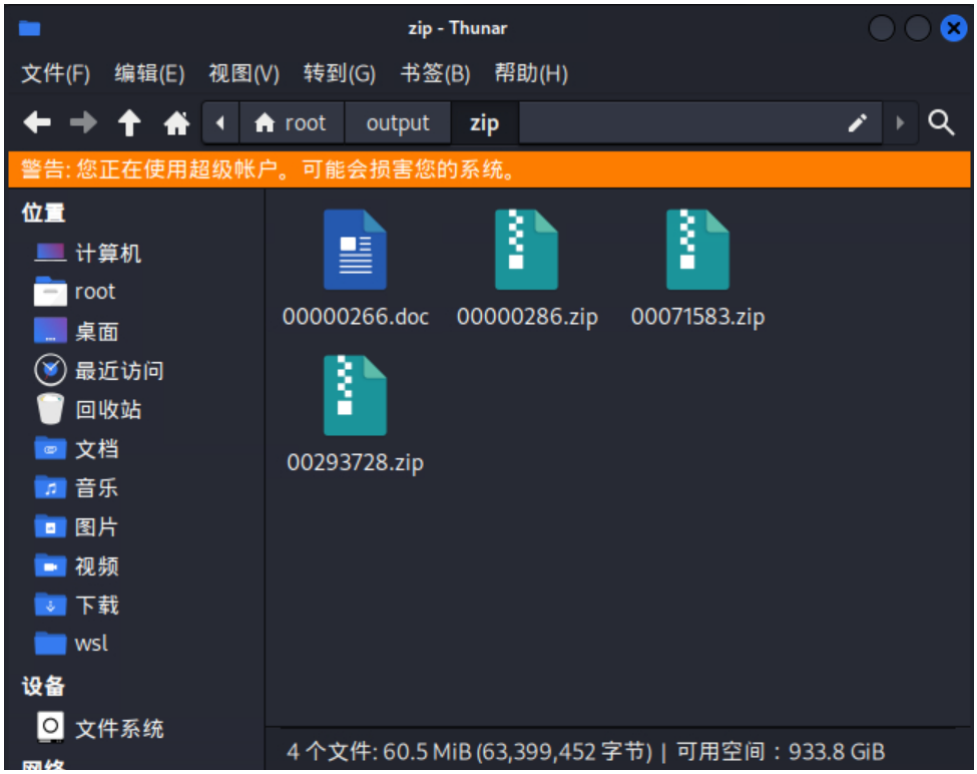
```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP0x64 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa8006cfbae0 System                    4      0     77      509 ------      0 2024-02-02 06:44:34 UTC+0000
0xfffffa80072713c0 smss.exe                236      4      2       29 ------      0 2024-02-02 06:44:34 UTC+0000
0xfffffa80075ddb30 csrss.exe               304    292      9      406      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8006d00b30 wininit.exe             340    292      3       78      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa80075e8060 csrss.exe               348    332      7      308      1      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8006d0c7f0 winlogon.exe            376    332      6      118      1      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e90b30 services.exe            432    340      9      192      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e94510 lsass.exe               448    340      7      594      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e985f0 lsm.exe                 456    340     10      140      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007efb4f0 svchost.exe             564    432     12      356      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007f29b30 svchost.exe             628    432      7      265      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007f4a4a0 svchost.exe             684    432     20      465      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007eedb30 svchost.exe             812    432     20      382      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007fe8890 svchost.exe             852    432     46     1051      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8008014b30 audiodg.exe             912    684      6      128      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080585d0 svchost.exe            1004    432     13      263      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080bb890 svchost.exe             876    432     20      407      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080fd3d0 spoolsv.exe            1080    432     13      271      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa8008123b30 svchost.exe            1116    432     20      319      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80081fa590 svchost.exe            1540    432      7       95      0      0 2024-02-02 06:44:37 UTC+0000
0xfffffa8008354060 taskhost.exe           1800    432     10      176      1      0 2024-02-02 06:44:54 UTC+0000
0xfffffa8008374b30 taskeng.exe            1852    852      5       78      0      0 2024-02-02 06:44:55 UTC+0000
0xfffffa8008373520 dwm.exe                1860    812      4       72      1      0 2024-02-02 06:44:55 UTC+0000
0xfffffa8008388b30 explorer.exe           1880   1844     33      898      1      0 2024-02-02 06:44:55 UTC+0000
0xfffffa80083d9880 YunDetectServi         1276   1880     22      255      1      1 2024-02-02 06:44:55 UTC+0000
0xfffffa8007f6e060 GoogleCrashHan         1620   2040      8       92      0      1 2024-02-02 06:44:57 UTC+0000
0xfffffa8008082580 GoogleCrashHan         1648   2040      7       82      0      0 2024-02-02 06:44:57 UTC+0000
0xfffffa8007f6c410 SearchIndexer.         1040    432     15      618      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa8008521b30 SearchProtocol          760   1040      8      277      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa8008547460 SearchFilterHo         1532   1040      7       98      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa800857c6e0 cmd.exe                2144   1880      1       20      1      0 2024-02-02 06:45:04 UTC+0000
0xfffffa8008582150 conhost.exe            2152    348      3       59      1      0 2024-02-02 06:45:04 UTC+0000
0xfffffa8008591b30 WmiPrvSE.exe           2260    564      8      119      0      0 2024-02-02 06:45:37 UTC+0000
0xfffffa8007ef89c0 iexplore.exe           2304   1880     21      634      1      1 2024-02-02 06:46:01 UTC+0000
0xfffffa80083fe4b0 iexplore.exe           2392   2304     24      631      1      1 2024-02-02 06:46:02 UTC+0000
0xfffffa8007db8060 svchost.exe            3048    432     12      151      0      0 2024-02-02 06:46:37 UTC+0000
0xfffffa8007f50060 sppsvc.exe             1500    432      6      153      0      0 2024-02-02 06:46:38 UTC+0000
0xfffffa8007fcab30 svchost.exe             968    432     12      330      0      0 2024-02-02 06:46:38 UTC+0000
0xfffffa800855f060 flag_is_here.e         1596   1880      1       15      1      1 2024-02-02 06:47:02 UTC+0000
0xfffffa80082f25f0 conhost.exe            2064    348      2       59      1      0 2024-02-02 06:47:02 UTC+0000
0xfffffa8007652b30 wps.exe                 748   1880      1       16      1      1 2024-02-02 06:47:04 UTC+0000
0xfffffa8007653b30 conhost.exe            1480    348      2       58      1      0 2024-02-02 06:47:04 UTC+0000
0xfffffa8007f71060 FTK Imager.exe         2912   1880     18      343      1      0 2024-02-02 06:47:11 UTC+0000
0xfffffa8007f70060 WmiPrvSE.exe           2988    564      9      177      0      0 2024-02-02 06:47:24 UTC+0000
```

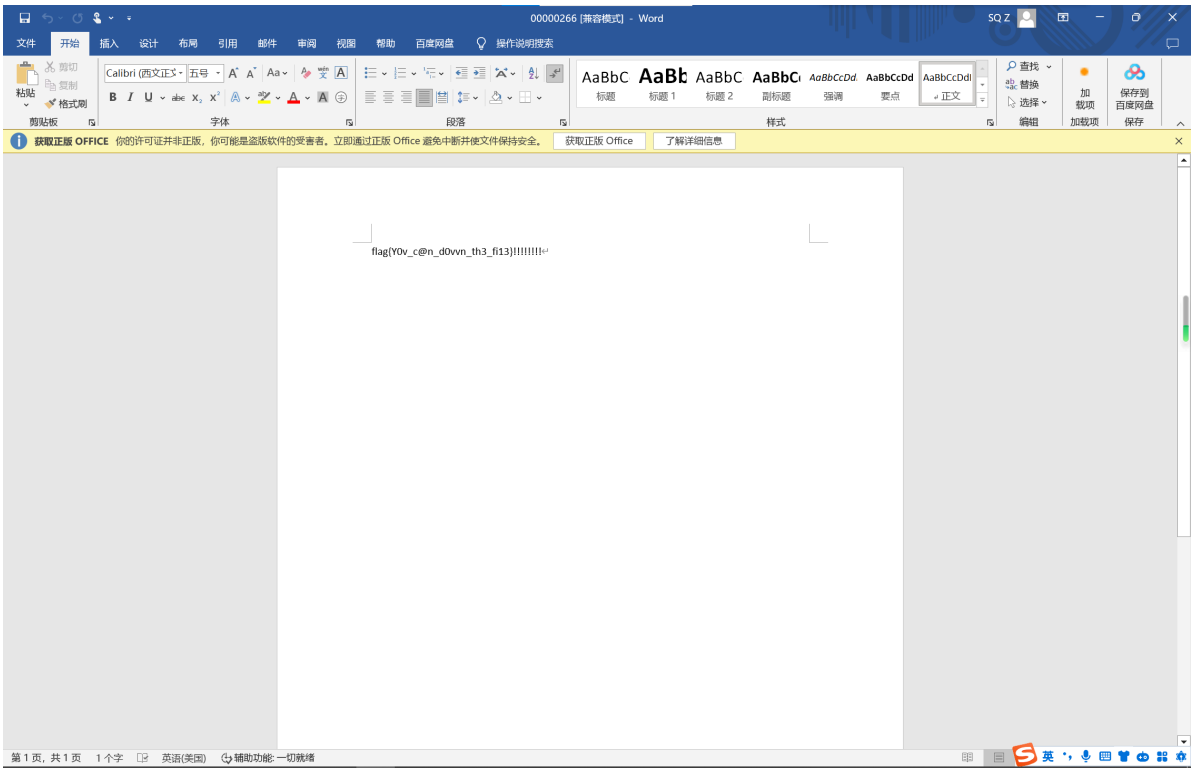
发现有个wps.exe的进程，把它dump下来

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP0x64 memdump -p 748 --dump-dir=/root
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
Writing wps.exe [   748] to 748.dmp
```

foremost分离

发现了几个被封装为zip的doc文档，更改其后缀并查看（除了报封装失败的，哪个都行）





学取证咯 - 真的是取证吗?

> 总算把这个系列肝完了，CTF生涯中的第一次二血

在pslist中很容易就能看到题目中所说的问题程序：flag_is_here.exe

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6.1
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa8006cfbae0 System                    4      0     77      509 ------      0 2024-02-02 06:44:34 UTC+0000
0xfffffa80072713c0 smss.exe                236      4      2       29 ------      0 2024-02-02 06:44:34 UTC+0000
0xfffffa80075ddb30 csrss.exe               304    292      9      406      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8006d00b30 wininit.exe             340    292      3       78      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa80075e8060 csrss.exe               348    332      7      308      1      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8006d0c7f0 winlogon.exe            376    332      6      118      1      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e90b30 services.exe            432    340      9      192      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e94510 lsass.exe               448    340      7      594      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007e985f0 lsm.exe                 456    340     10      140      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007efb4f0 svchost.exe             564    432     12      356      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007f29b30 svchost.exe             628    432      7      265      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007f4a4a0 svchost.exe             684    432     20      465      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007eedb30 svchost.exe             812    432     20      382      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8007fe8890 svchost.exe             852    432     46     1051      0      0 2024-02-02 06:44:35 UTC+0000
0xfffffa8008014b30 audiodg.exe             912    684      6      128      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080585d0 svchost.exe            1004    432     13      263      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080bb890 svchost.exe             876    432     20      407      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80080fd3d0 spoolsv.exe            1080    432     13      271      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa8008123b30 svchost.exe            1116    432     20      319      0      0 2024-02-02 06:44:36 UTC+0000
0xfffffa80081fa590 svchost.exe            1540    432      7       95      0      0 2024-02-02 06:44:37 UTC+0000
0xfffffa8008354060 taskhost.exe           1800    432     10      176      1      0 2024-02-02 06:44:54 UTC+0000
0xfffffa8008374b30 taskeng.exe            1852    852      5       78      0      0 2024-02-02 06:44:55 UTC+0000
0xfffffa8008373520 dwm.exe                1860    812      4       72      1      0 2024-02-02 06:44:55 UTC+0000
0xfffffa8008388b30 explorer.exe           1880   1844     33      898      1      0 2024-02-02 06:44:55 UTC+0000
0xfffffa80083d9880 YunDetectServi         1276   1880     22      255      1      1 2024-02-02 06:44:55 UTC+0000
0xfffffa8007f6e060 GoogleCrashHan         1620   2040      8       92      0      1 2024-02-02 06:44:57 UTC+0000
0xfffffa8008082580 GoogleCrashHan         1648   2040      7       82      0      0 2024-02-02 06:44:57 UTC+0000
0xfffffa8007f6c410 SearchIndexer.         1040    432     15      618      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa8008521b30 SearchProtocol          760   1040      8      277      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa8008547460 SearchFilterHo         1532   1040      7       98      0      0 2024-02-02 06:45:01 UTC+0000
0xfffffa800857c6e0 cmd.exe                2144   1880      1       20      1      0 2024-02-02 06:45:04 UTC+0000
0xfffffa8008582150 conhost.exe            2152    348      3       59      1      0 2024-02-02 06:45:04 UTC+0000
0xfffffa8008591b30 WmiPrvSE.exe           2260    564      8      119      0      0 2024-02-02 06:45:37 UTC+0000
0xfffffa8007ef89c0 iexplore.exe           2304   1880     21      634      1      1 2024-02-02 06:46:01 UTC+0000
0xfffffa80083fe4b0 iexplore.exe           2392   2304     24      631      1      1 2024-02-02 06:46:02 UTC+0000
0xfffffa8007db8060 svchost.exe            3048    432     12      151      0      0 2024-02-02 06:46:37 UTC+0000
0xfffffa8007f50060 sppsvc.exe             1500    432      6      153      0      0 2024-02-02 06:46:38 UTC+0000
0xfffffa8007fcab30 svchost.exe             968    432     12      330      0      0 2024-02-02 06:46:38 UTC+0000
0xfffffa800855f060 flag_is_here.e         1596   1880      1       15      1      1 2024-02-02 06:47:02 UTC+0000
0xfffffa80082f25f0 conhost.exe            2064    348      2       59      1      0 2024-02-02 06:47:02 UTC+0000
0xfffffa8007652b30 wps.exe                 748   1880      1       16      1      1 2024-02-02 06:47:04 UTC+0000
0xfffffa8007653b30 conhost.exe            1480    348      2       58      1      0 2024-02-02 06:47:04 UTC+0000
0xfffffa8007f71060 FTK Imager.exe         2912   1880     18      343      1      0 2024-02-02 06:47:11 UTC+0000
0xfffffa8007f70060 WmiPrvSE.exe           2988    564      9      177      0      0 2024-02-02 06:47:24 UTC+0000
```

用procdump把它dump下来

```
┌──(root㉿DESKTOP-LQMRD0K)-[~]
└─# vol.py -f 1.raw --profile=Win7SP1x64 procdump -p 1596 --dump-dir=/root
Volatility Foundation Volatility Framework 2.6.1
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xfffffa800855f060 0x0000000000400000 flag_is_here.e       OK: executable.1596.exe
```

丢进IDA分析一下，看到sub_401460这个函数有点东西

```
int sub_401460()
{
  char Str2[2]; // [esp+1Eh] [ebp-6Ah] BYREF
  char v2[39]; // [esp+20h] [ebp-68h] BYREF
  char Str[41]; // [esp+47h] [ebp-41h] BYREF
  int v4; // [esp+70h] [ebp-18h]
  char v5; // [esp+77h] [ebp-11h]
  char *v6; // [esp+78h] [ebp-10h]
  int i; // [esp+7Ch] [ebp-Ch]

  sub_401BA0();
  v2[38] = 0;
  memset(v2, 0, 4 * (((Str2 - v2 + 41) & 0xFFFFFFFC) >> 2));
  Str2[0] = 31;
  Str2[1] = 21;
  v2[0] = 24;
  v2[1] = 30;
  v2[2] = 2;
  v2[3] = 32;
  v2[4] = 73;
  v2[5] = 15;
  v2[6] = 38;
  v2[7] = 17;
  v2[8] = 57;
  v2[9] = 15;
  v2[10] = 74;
```

```
  v2[11] = 38;
  v2[12] = 21;
  v2[13] = 74;
  v2[14] = 57;
  v2[15] = 11;
  v2[16] = 23;
  v2[17] = 74;
  v2[18] = 29;
  v2[19] = 38;
  v2[20] = 17;
  v2[21] = 73;
  v2[22] = 15;
  v2[23] = 15;
  v2[24] = 38;
  v2[25] = 13;
  v2[26] = 73;
  v2[27] = 38;
  v2[28] = 31;
  v2[29] = 73;
  v2[30] = 11;
  v2[31] = 74;
  v2[32] = 23;
  v2[33] = 76;
  v2[34] = 16;
  v2[35] = 26;
  v2[36] = 76;
  v2[37] = 4;
  v6 = getenv("FLAG_KEY");
  sub_409860(&unk_40B06D);
  if ( v6 )
  {
    v5 = v6[2];
    sub_409860(&unk_40B096);
    gets(Str);
    v4 = strlen(Str);
    for ( i = 0; i < v4; ++i )
    {
      Str[i] ^= v5;
      sub_409860("0x%02x,");
    }
    if ( !strcmp(Str, Str2) )
      sub_409860(&unk_40B0AD);
    else
      sub_409860(&unk_40B0B3);
    return 0;
  }
  else
  {
    sub_409860(&unk_40B07C);
    return 1;
  }
}
```

分析一下，应该是个简单的异或，不过需要一个异或的对象FLAG_KEY，这个要去环境变量中找。回到volatility，用envars看一下这个进程的环境变量，找到了FLAG_KEY的值。

```
(root㉿DESKTOP-LQMRD0K)-[~]
# vol.py -f 1.raw --profile=Win7SP1x64 envars -p 1596
Volatility Foundation Volatility Framework 2.6.1
Pid      Process              Block              Variable                       Value
-------- -------------------- ------------------ ------------------------------ -----
    1596 flag_is_here.e       0x0000000000511320 ALLUSERSPROFILE                C:\ProgramData
    1596 flag_is_here.e       0x0000000000511320 APPDATA                        C:\Users\yuren\AppData\Roaming
    1596 flag_is_here.e       0x0000000000511320 CommonProgramFiles             C:\Program Files\Common Files
    1596 flag_is_here.e       0x0000000000511320 CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
    1596 flag_is_here.e       0x0000000000511320 CommonProgramW6432             C:\Program Files\Common Files
    1596 flag_is_here.e       0x0000000000511320 COMPUTERNAME                   VVHATI5Y0VRNAM3
    1596 flag_is_here.e       0x0000000000511320 ComSpec                        C:\Windows\system32\cmd.exe
    1596 flag_is_here.e       0x0000000000511320 FLAG_KEY                       key?
    1596 flag_is_here.e       0x0000000000511320 FP_NO_HOST_CHECK               NO
    1596 flag_is_here.e       0x0000000000511320 HOMEDRIVE                      C:
    1596 flag_is_here.e       0x0000000000511320 HOMEPATH                       \Users\yuren
    1596 flag_is_here.e       0x0000000000511320 LOCALAPPDATA                   C:\Users\yuren\AppData\Local
    1596 flag_is_here.e       0x0000000000511320 LOGONSERVER                    \\VVHATI5Y0VRNAM3
    1596 flag_is_here.e       0x0000000000511320 NUMBER_OF_PROCESSORS           1
    1596 flag_is_here.e       0x0000000000511320 OS                             Windows_NT
    1596 flag_is_here.e       0x0000000000511320 Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
    1596 flag_is_here.e       0x0000000000511320 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
    1596 flag_is_here.e       0x0000000000511320 PROCESSOR_ARCHITECTURE         AMD64
    1596 flag_is_here.e       0x0000000000511320 PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 165 Stepping 2, GenuineIntel
    1596 flag_is_here.e       0x0000000000511320 PROCESSOR_LEVEL                6
    1596 flag_is_here.e       0x0000000000511320 PROCESSOR_REVISION             a502
    1596 flag_is_here.e       0x0000000000511320 ProgramData                    C:\ProgramData
    1596 flag_is_here.e       0x0000000000511320 ProgramFiles                   C:\Program Files
    1596 flag_is_here.e       0x0000000000511320 ProgramFiles(x86)              C:\Program Files (x86)
    1596 flag_is_here.e       0x0000000000511320 ProgramW6432                   C:\Program Files
    1596 flag_is_here.e       0x0000000000511320 PSModulePath                   C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
    1596 flag_is_here.e       0x0000000000511320 PUBLIC                         C:\Users\Public
    1596 flag_is_here.e       0x0000000000511320 SESSIONNAME                    Console
    1596 flag_is_here.e       0x0000000000511320 SystemDrive                    C:
    1596 flag_is_here.e       0x0000000000511320 SystemRoot                     C:\Windows
    1596 flag_is_here.e       0x0000000000511320 TEMP                           C:\Users\yuren\AppData\Local\Temp
    1596 flag_is_here.e       0x0000000000511320 TMP                            C:\Users\yuren\AppData\Local\Temp
    1596 flag_is_here.e       0x0000000000511320 USERDOMAIN                     VVHATI5Y0VRNAM3
    1596 flag_is_here.e       0x0000000000511320 USERNAME                       yuren
    1596 flag_is_here.e       0x0000000000511320 USERPROFILE                    C:\Users\yuren
    1596 flag_is_here.e       0x0000000000511320 windir                         C:\Windows
    1596 flag_is_here.e       0x0000000000511320 windows_tracing_flags          3
    1596 flag_is_here.e       0x0000000000511320 windows_tracing_logfile        C:\BVTBin\Tests\installpackage\csilogfile.log
```

题目中用到了v6[2]，也就是字符'y'，接下来写个python脚本异或一下就行

```
v2 = [31,21,24, 30, 2, 32, 73, 15, 38, 17, 57, 15,74, 38, 21, 74, 57, 11, 23,
74, 29, 38,17, 73, 15, 15, 38, 13, 73, 38, 31, 73,11, 74, 23, 76, 16, 26, 76, 4]
s=''
for i in v2:
    s+=chr(i^ord('y'))
print(s)
```

```
flag{Y0v_h@v3_l3@rn3d_h0vv_t0_f0r3n5ic5}
```