bmzctf-crypto writeup（二） (持续更新)

原创

dameow 已于 2022-02-21 09:42:33 修改 2438 收藏 1

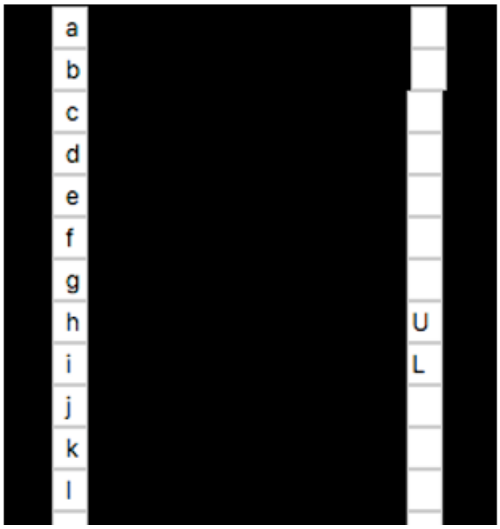
分类专栏： CTF 文章标签： web安全 网络安全 unctf

于 2022-01-10 10:32:08 首次发布

2018 HEBTUCTF lazy cipher

Looks like I'm too lazy to complete it. >_< Can you help me a little?

EULS LS I XCISSLXIC SHNEHNXH: EUH VWLXF YDAKN OAR MWBJS AZHD EUH CIPT GAQ! OLQUE, ING TAW BIT GLH. DWN, ING TAW'CC CLZH IE CHISE I KULCH. ING GTLNQ LN TAWD YHGS BINT THIDS ODAB NAK. KAWCG TAW YH KLCCLNQ EA EDIGH? ICC EUH GITS ODAB EULS GIT EA EUIE, OAD ANH XUINXH, MWSE ANH XUINXH, EA XABH YIXF UHDH ING EHCC AWD HNHBLHS EUIE EUHT BIT EIFH AWD CLZHS, YWE EUHT'CC NHZHD EIFH AWD OCIQ!OCIQ LS UHYEWXEO KLONZGSOOGTWYJIBHBS.



看到这样的乱乱的字母，首先要反应到用词频分析：https://quipqiup.com/

quipqiup beta3

quipqiup is a fast and automated cryptogram solver by Edwin Olson. It can solve simple substitution ciphers often found in newspapers, including puzzles like cryptoquips (in which word boundaries are preserved) and patristocrats (inwhi chwor dboun darie saren t).

Puzzle:

EULS LS I XCISSLXIC SHNEHNXH: EUH VWLXF YDAKN OAR MWBJS AZHD EUH CIPT GAQ! OLQUE, ING TAW BIT GLH. DWN, ING TAW'CC CLZH IE CHISE I KULCH. ING GTLNQ LN TAWD YHGS BINT THIDS ODAB NAK. KAWCG TAW YH KLCCLNQ EA EDIGH? ICC EUH GITS ODAB EULS GIT EA EUIE, OAD ANH XUINXH, MWSE ANH XUINXH, EA XABH YIXF UHDH ING EHCC AWD HNHBLHS EUIE EUHT BIT EIFH AWD CLZHS, YWE EUHT'CC NHZHD EIFH AWD OCIQ!OCIQ LS UHYEWXEO KLONZGSOOGTWYJIBHBS.

Clues: For example G=R QVW=THE

Solve

automatically selected statistics mode; you can override by using the drop down menu next to the solve button.

0 -1.898 THIS IS A CLASSICAL SENTENCE: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG! FIGHT, AND YOU MAY DIE. RUN, AND YOU'LL LIVE AT LEAST A WHILE. AND DYING IN YOUR BEDS MANY YEARS FROM NOW. WOULD YOU BE WILLING TO TRADE? ALL THE DAYS FROM THIS DAY TO THAT, FOR ONE CHANCE, JUST ONE CHANCE, TO COME BACK HERE AND TELL OUR ENEMIES THAT THEY MAY TAKE OUR LIVES, BUT THEY'LL NEVER TAKE OUR FLAG!FLAG IS HEBTUCTF WIFNVDSFFDYUBPAMEMS.

直接出flag，根本不需要知道题目考察什么。

火眼金睛

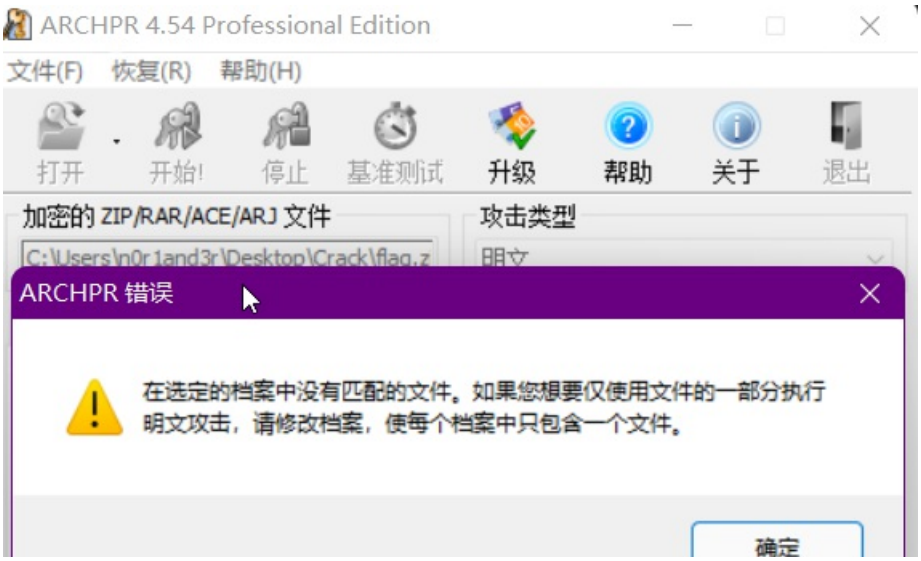
给了两个文件，是加密的。

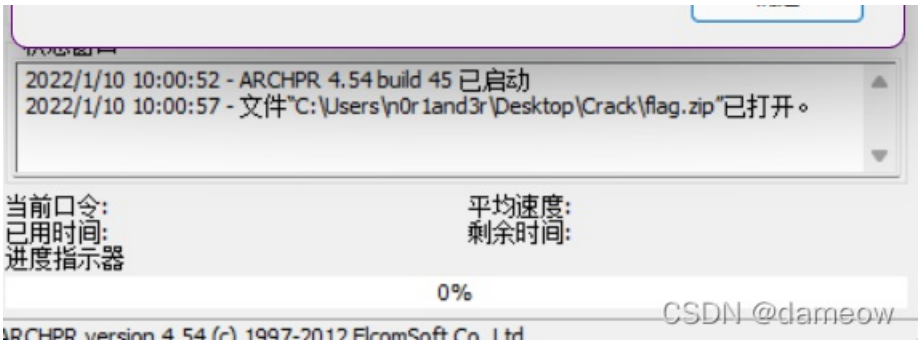
名称	压缩后大小	修改时间
flag.zip	287 502	2018-12-06 13:27
readme.txt	611	2018-12-06 13:21

7zip打不开，说明不是伪加密。
尝试爆破：

密码52181
发现里面还有一层。

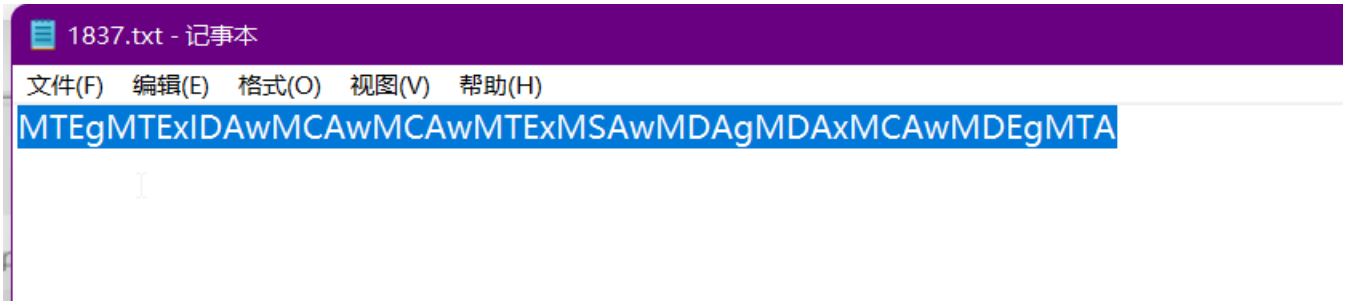
外面有一个readme.txt，里面也有一个readme.txt，所有考虑明文攻击。
（但是我就是不成功，看别人的也是明文攻击啊。。。）




2022/1/10 10:00:52 - ARCHPR 4.54 build 45 已启动
2022/1/10 10:00:57 - 文件"C:\Users\n0r1and3r\Desktop\Crack\flag.zip"已打开。
当前口令:
平均速度:
已用时间:
剩余时间:
进度指示器
0%
CSDN @dameow

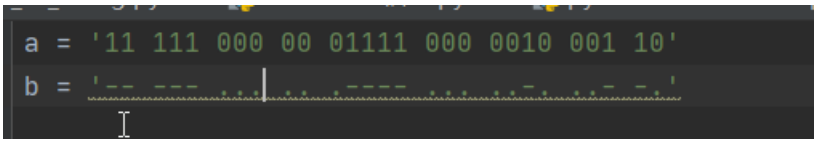
淦。。。

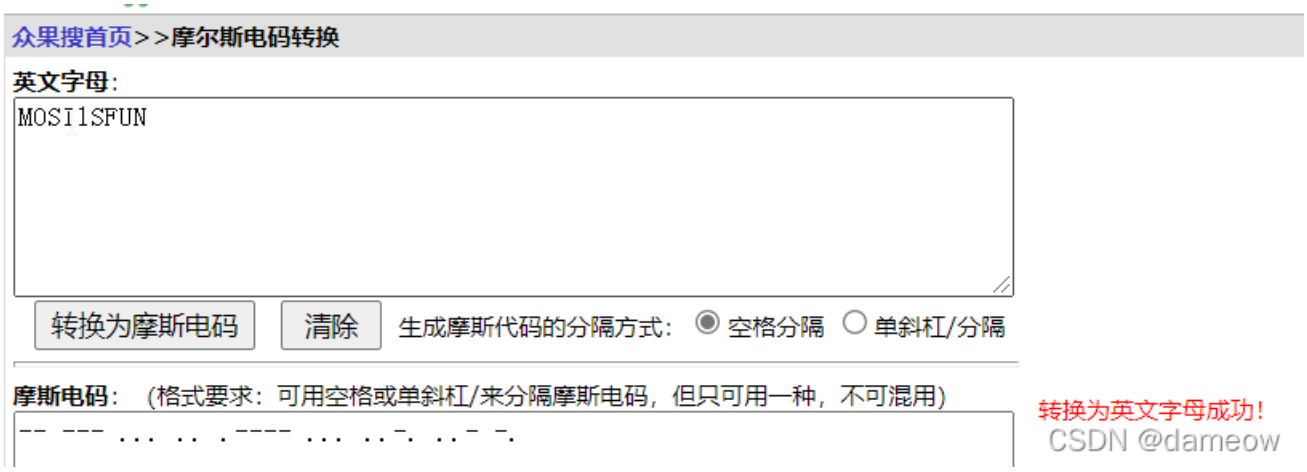
1837



拿去base64解密，得到二进制。

把二进制转字符串，转十进制怎么都不对，没想到是摩斯电码，摩斯电码是1837发明的，对应的题目。





flag{MOSI1SFUN}

Caesar

Can you untie it?

ch\at]ZW+S$)6Q#k

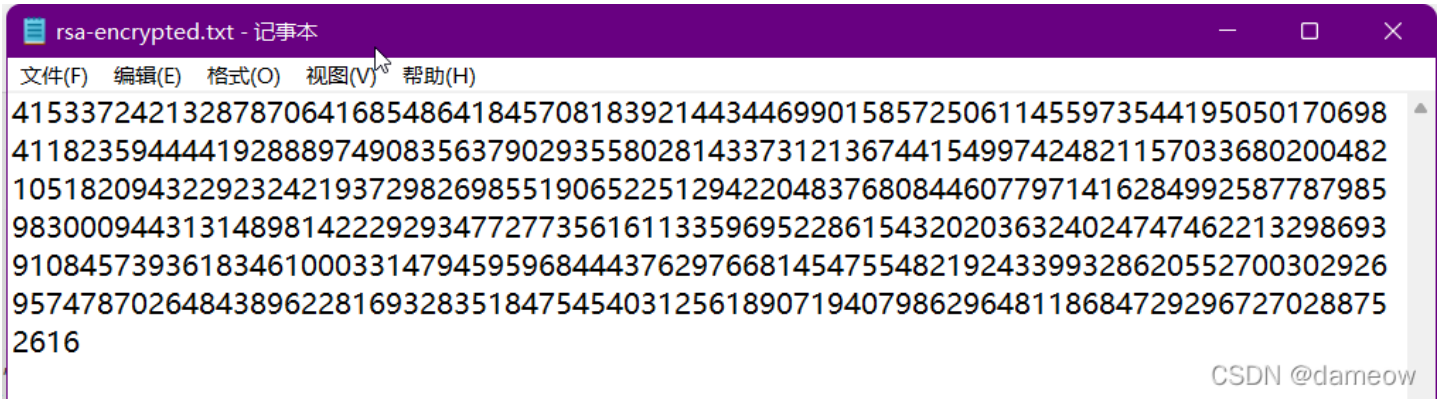
Tip：flag格式为flag{xxx}

变异凯撒，编译的凯撒一般都是按顺序递增或者递减移位，查看前面四位和flag的关系，c到f移位3，h到l移位5，\到a移位5，a到g移位6，说明是递增。脚本：

```
a = 'ch\\at]ZW+S$)6Q#k'
b = ''
for i in range(len(a)):
    b += chr(ord(a[i])+ 3 + i)
print(b)
```

反斜杠需要转义。
flag{eca6_17Ea4}

rsa

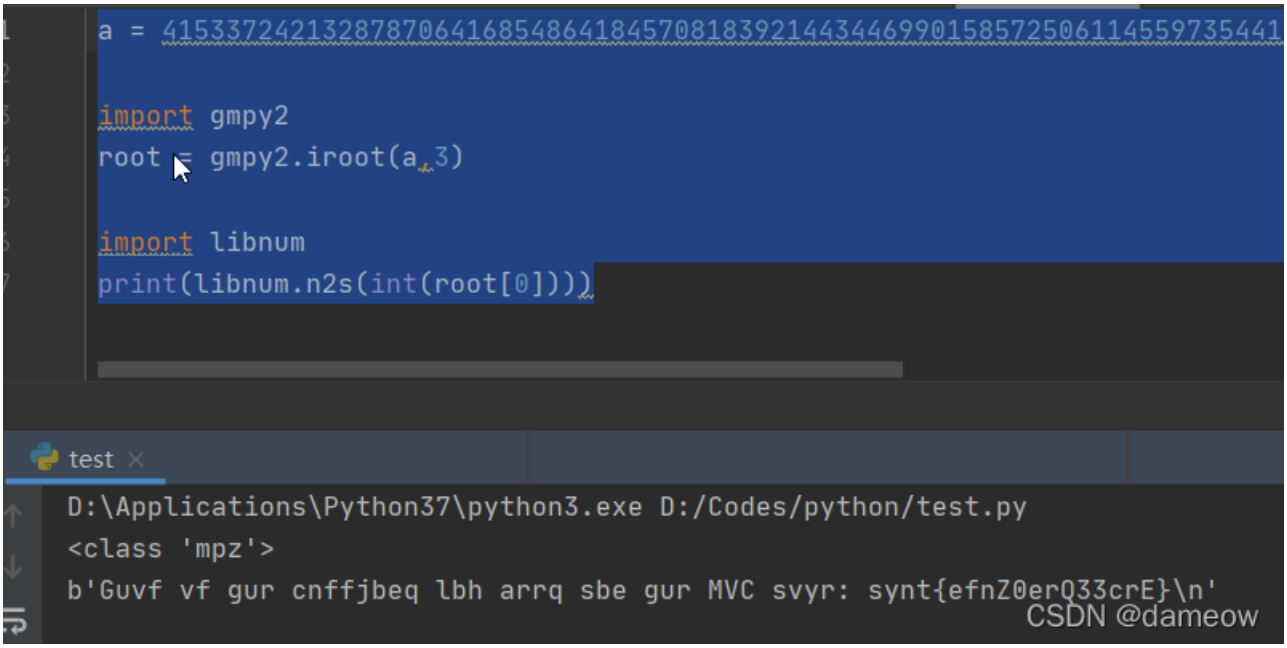


从文件名可以看出，这是密文。
只有密文是不能解密rsa的。
从加密函数c = m^e (mod n)来看，除非e很小，造成低加密指数攻击，并且，m^e<n，就直接不断开根号爆破这个e，得到明文m。
爆破e脚本：

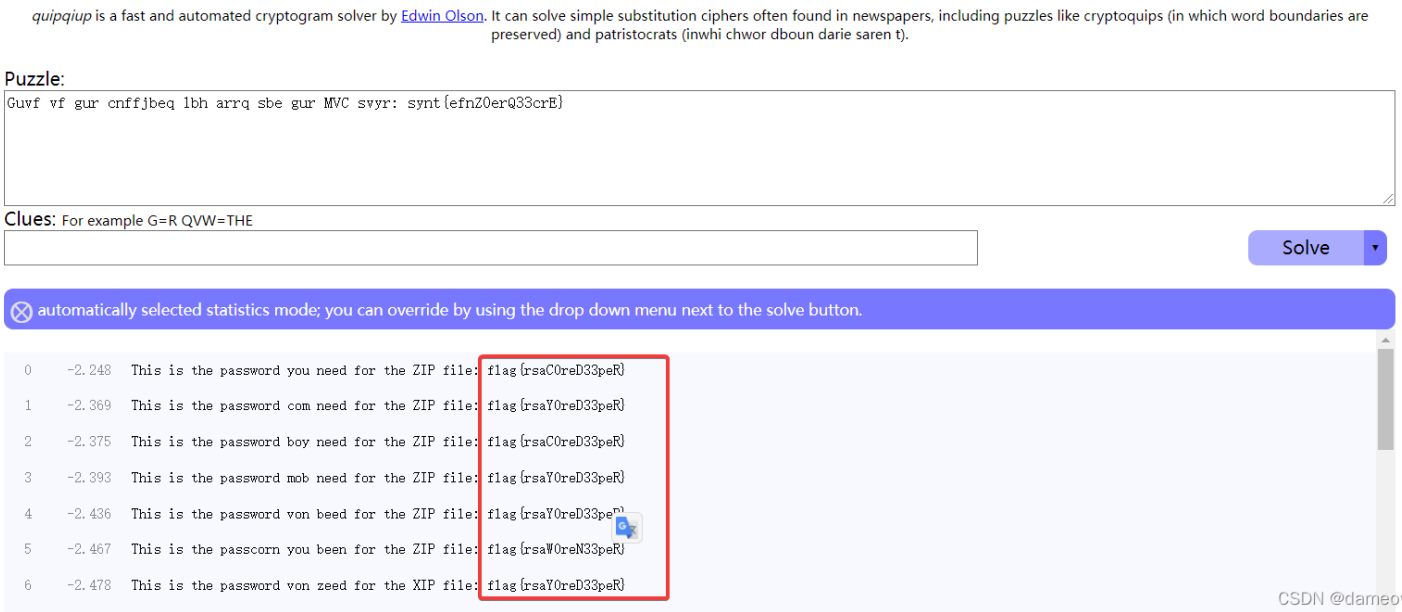
```
a = 41533724213287870641685486418457081839214434469901585725061145597354419505017069841182359444419288897490835637902935580281433731213674415499742482115703368020048210518209432292324219372982698551906522512942204837680844607797141628499259877879859830009443131489814222929347727735616113359695228615432020363240247474622132986939108457393618346100033147945959684443762976681454755482192433993286205527003029269574787026484389622816932835184754540312561890719407986296481186847292967270288752616

import gmpy2
for e in range(2,100):
    root = gmpy2.iroot(a,e)
    if root[1] == True:
        print(e)
```

结果是3，e=3。
那么解密脚本：

```
a = 41533724213287870641685486418457081839214434469901585725061145597354419505017069841182359444419288897490835637902935580281433731213674415499742482115703368020048210518209432292324219372982698551906522512942204837680844607797141628499258778798598300094431314898142229293477277356161133596952286154320203632402474746221329869391084573936183461000331479459596844437629766814547554821924339932862055270030292695747870264843896228169328351847545403125618907194079862964811868472929672702887526 16

import gmpy2
root = gmpy2.iroot(a,3)

import libnum
print(libnum.n2s(int(root[0])))
```



得到的synt{efnZ0erQ33crE}并不是flag，拿去词频分析，怎么都不对。



原来是凯撒密码，移位13。

凯撒密码加密解密

Guvf vf gur cnffjbeq lbh arrq sbe gur MVC svyr: synt{efnZ0erQ33crE}

位移 13 加密 解密

This is the password you need for the ZIP file: flag{rsaM0reD33peR}

CSDN @dameow

技协杯-Crypto1

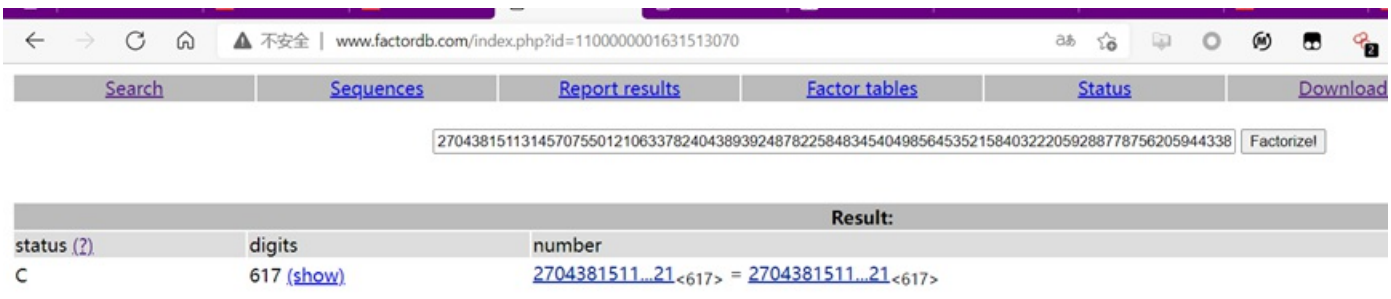
考察：

- RSA低加密指数攻击
- Python脚本

题目给了如下信息：

N=
270438151131457075501210633782404389392487822584834540498564535215840322205928877875620594433800386
102905985756220629164853071840914095545385148542024988735472977731032393169542396416673368943133678
457728450230657214391345013104265900193558445247569200783070053249295260634254876422094563550232302
084678607747427850993179013989638671065824767403260827469532622165040201110958176136563477220831950
761438471890350350630217575743825576634794421900506959271081984201919432718757983806840965926809306
442001364364074727786308110004560735488361968903753508273569069535596026273924005244986826420187174
473022973254323181772 21
c=551046197285606898107283024676650283848098926639864466516013084135553632241090727641369586261735 5
936646132852462993038758543109874471031857950404905411061547902438936991547678694613605323702583239
366069488873885610625249288097306336647968725558401446623024736147226016692153463491573070792980934
874297761853307340675918803389596872299837245872674279024432457686193702739949816046783820511697654
136850492399268165051567370669797284490139086244295253355986745048375485230065337234499266780950266
44921098893
e = 5

考虑过把模数n因数分解，但是没能分解出来。在线分解：factordb.com


不安全 | www.factordb.com/index.php?id=1100000001631513070
Search
Sequences
Report results
Factor tables
Status
Download
27043815113145707550121063378240438939248782258483454049856453521584032220592887787562059443380038610290598575622062916485307184091409554538514854202498873547297773103239316954239641667336894313367845772845023065721439134501310426590019355844524756920078307005324929526063425487642209456355023230208467860774742785099317901398963867106582476740326082746953262216504020111095817613656347722083195076143847189035035063021757574382557663479442190050695927108198420191943271875798380684096592680930644200136436407472778630811000456073548836196890375350827356906953559602627392400524498682642018717447302297325432318177221
c=551046197285606898107283024676650283848098926639864466516013084135553632241090727641369586261735 5
936646132852462993038758543109874471031857950404905411061547902438936991547678694613605323702583239
366069488873885610625249288097306336647968725558401446623024736147226016692153463491573070792980934
874297761853307340675918803389596872299837245872674279024432457686193702739949816046783820511697654
136850492399268165051567370669797284490139086244295253355986745048375485230065337234499266780950266
44921098893
Factorize!

Result:		
status (?)	digits	number
C	617 (show)	2704381511...21<617> = 2704381511...21<617>

考虑另外的思路，看到e很小，相对于模数n来说很小，想到低加密指数。
我们知道，RSA加密函数如下：

$$C = m^e \bmod n$$

其中，c为密文，m为明文，e为加密指数，n为模数。 如果m的e次方，小于模数n，即：

$$C = m^e < n$$

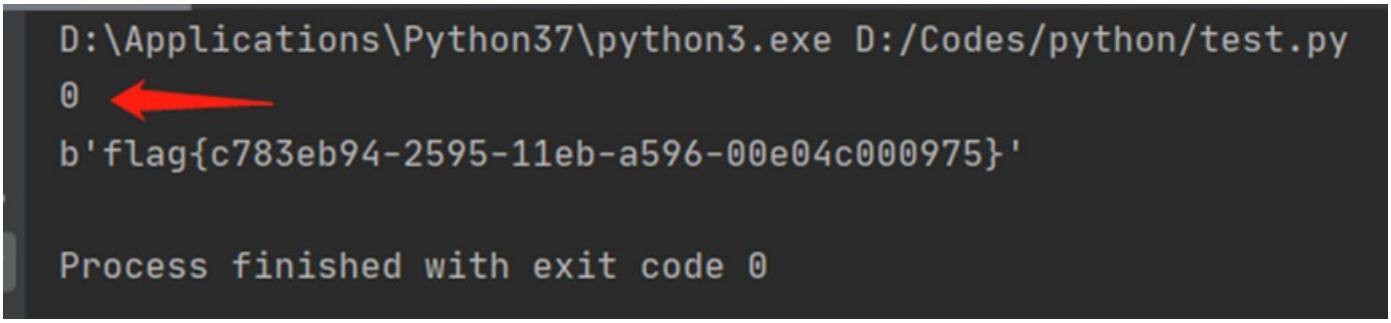
那么会出现直接对密文c开e次方，便可得出明文m。 如果m的e次方，大于模数n，即：

$$C = m^e > n$$

那么必然存在一个整数k，使得： $C = m^e + kn$ 那么，可以爆破这个k值，同时不断地开e次方，直到能开出来。 脚本如下：

```
import libnum
import gmpy2

n = 270438151131457075501210633782404389392487822584834540498564535215840322205928877875620594433800386102905985
756220629164853071840914095545385148542024988735472977731032393169542396416673368943133678457728450230657214391345013104265900193558445247569200783070053249295260634254876422094563550232302084678607747427850993179013989638671065824767403260827469532622165040201110958176136563477220831950761438471890350350630217575743825576634794421900506959271081984201919432718757983806840965926809306442001364364074727786308110004560735488361968903753508273569069535596026273924005244986826420187174473022973254323181772211
e = 5
flag = 0
c = 551046197285606898107283024676650283848098926639864466516013084135553632241090727641369586261735593664613285246299303875854310987447103185795040490541106154790243893699154767869461360532370258323936606948887385610625249288097306336647968725558401446623024736147226016692153463491573070792980934874297761853307340675918803389596872299837245872674279024432457686193702739949816046783820511697654136850492399268165051567370669797284490139086244295253355986745048375485230065337234499266780950266449210988931
for k in range(200000000):
    if gmpy2.iroot(c + n * k, e)[1] == 1:
        '''
        这个函数是开n次方根，gmpy2.iroot（x,n），x的n次方根。
        返回值[0]是方根，[1]是布尔值。
        如a = gmpy2.iroot(4,2)，则返回值a[0]=2,a[1]=TRUE。
        '''
        m = gmpy2.iroot(c + n * k, e)[0]
        print(k)
        print(libnum.n2s(int(m))) # gmpy2库的数据格式是元组，元素格式是mpz，需要转成int再转string
        break
```

在2亿次循环中，爆破这个k值。



K的值是0，说明m的e次方，是小于模数n的。
flag{c783eb94-2595-11eb-a596-00e04c000975}