攻防世界 ——crypto

原创

Captain Hammer 于 2019-10-17 17:07:35 发布 7136 收藏 22
分类专栏： CTF题解

本文链接：https://blog.csdn.net/vhkjhwbs/article/details/101026050

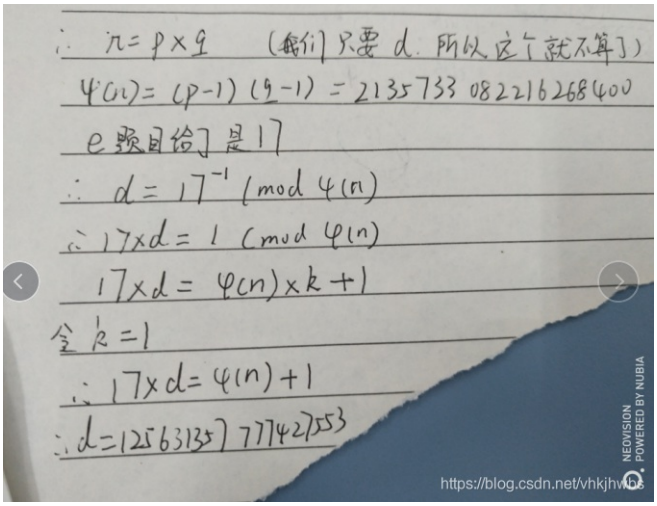

CTF题解 专栏收录该内容
11 篇文章 7 订阅
订阅专栏

目录

新手区部分题解：

1，easy_RSA

摘别人的图



或者用 脚本 python2：（装了 gmpy2库的可以用）

```
import gmpy2
p = 473398607161
q = 4511491
e = 17
s = (p-1)*(q-1)

d = gmpy2.invert(e,s)

print d
```

2，Normal_RSA

解压出来两个文件：flag.enc 和 pubkey.pem

```
pem文件：以前是利用公钥加密进行邮件安全的一个协议，而现在PEM这个协议仅仅在使用的就是.pem这种文件
格式，这种文件里面保存了keys和X.509证书，看到了吗数字证书和key都能保存。
而现在一般表示用Base64 encoded DER编码的证书，
这种证书的内容打开能够看到在”--BEGIN CERTIFICATE--” and “--END CERTIFICATE--”之间
```

```
enc文件：该.enc文件扩展用于通过在UUenconded格式文件，它们是加密的文件。
这也意味着这些ENC文件包含受保护的数据和保存在该格式中，所以未经授权的收视数据和复制可以被防止
```

本题考查 两个工具的使用 openssl （linux自带） 和 rsatool

这两个工具我是在 windows 下 安装使用的，当然可以在 linux中使用，个人喜好

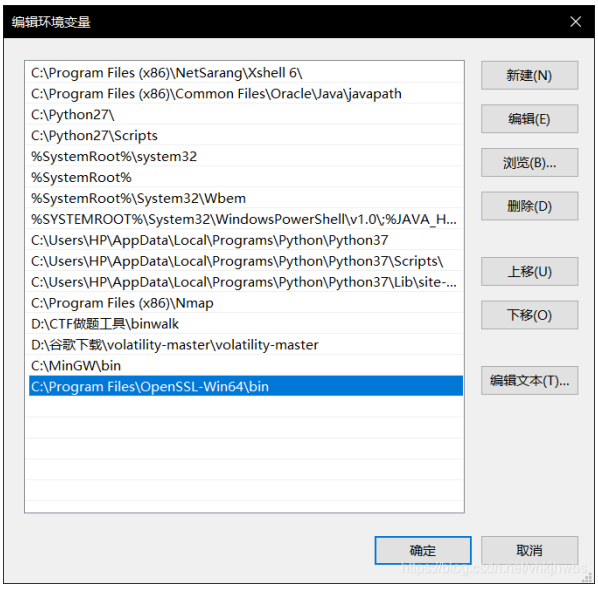
分享：openssl： https://pan.baidu.com/s/1wP73GPJZo-HITH50UA61YQ 提取码: v2ti

rsatools ： https://pan.baidu.com/s/16Hl_p1zv_ajZadkW5GFnzQ 提取码: 8qqr

其中 openssl 是exe文件 直接执行安装，默认安装到：C:\Program Files\OpenSSL-Win64

然后把 C:\Program Files\OpenSSL-Win64\bin 添加到 环境变量中

我的电脑 右键 》 属性 》高级系统设置 》环境变量 》系统变量 中 双击 path 》 新建 》把复制的路径添加进去 确定



一个是加密过的的flag.enc，另一个是公钥pubkey.pem。我们需要用四个步骤拿到flag

把 flag.enc 和 pubkey.pem 复制到 rsatools 文件夹中， 打开cmd

1，提取 pubkey.pem 中的 参数：

```
openssl rsa -pubin -text -modulus -in warmup -in pubkey.pem
```

```
:\Users\HP\Desktop\不常用软件\加密解密工具\RSA\rsatools(生成私钥)>openssl rsa -pubin -text -modulus -in warmup -in pubk
y.pem
SA Public-Key: (256 bit)
odulus:
    00:c2:63:6a:e5:c3:d8:e4:3f:fb:97:ab:09:02:8f:
    1a:ac:6c:0b:f6:cd:3d:70:eb:ca:28:1b:ff:e9:7f:
    be:30:dd
xponent: 65537 (0x10001)
odulus=C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD
riting RSA key
----BEGIN PUBLIC KEY-----
DwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMJjauXD2OQ/+5erCQKPGqxsC/bNPXDr
igb/+1/vjDdAgMBAAE=
----END PUBLIC KEY-----

:\Users\HP\Desktop\不常用软件\加密解密工具\RSA\rsatools(生成私钥)>
```


对得到的Modulus（16）进制的转化为十进制：

```
879243482641324068752761405144999371450508936656025929924181716470424916584 61
```

然后把 这个数分解 为两个 互质数 就是 q和 p

这里需要用到 yafu 中的 factor (kali自带)，我偏向于在windows中 玩

分享 源码，python脚本 linux也可以用，

https://pan.baidu.com/s/1TQqM8naEyVKhh101kz_igQ 提取码: rcb9

```
==== sieving in progress (1 thread):   36224 relations needed ====
====          Press ctrl-c to abort and save state          ====


SIQS elapsed time = 1.8158 seconds.
Total factoring time = 1.8501 seconds


***factors found***

P39 = 319576316814478949870590164193048041239
P39 = 275127860351348928173285174381581152299

ans = 1

C:\Users\HP\Desktop\不常用软件\加密解密工具\RSA\wafu(大数分解)>
```



```
275127860351348928173285174381581152299
319576316814478949870590164193048041239
```

知道两个素数，随机定义大素数e,求出密钥文件 private.pem （python2）

```
python rsatool.py -o private.pem -e 65537 -p 275127860351348928173285174381581152299 -q 3195763168144789498
```

最后解密：

```
openssl rsautl -decrypt -in flag.enc -inkey private.pem
```

```
d =
1806799bd44ce649122b78b43060c786f8b77fb1593e0842da063ba0d8728bf1

p = 275127860351348928173285174381581152299 (0xcefbb2cf7e18a98ebedc36e3e7c3b02b)

q = 319576316814478949870590164193048041239 (0xf06c28e91c8922b9c236e23560c09717)

Saving PEM as private.pem

C:\Users\HP\Desktop\不常用软件\加密解密工具\RSA\rsatools(生成私钥)>openssl rsautl -decrypt -in flag.enc -inkey private.p
em
PCTF{256b_i5_m3dium}

C:\Users\HP\Desktop\不常用软件\加密解密工具\RSA\rsatools(生成私钥)>
```


openssl是一个功能强大的工具，https://www.cnblogs.com/yangxiaolan/p/6256838.html

3, 幂数加密

提示是 幂函数解密：

二进制幂数加密法

编辑 讨论

本词条缺少概述、信息栏，补充相关内容使词条更完整，还能快速升级，赶紧来编辑吧！

二进制数除了0和1的表示方法外，在由二进制转换成十进制的时候，还可以表示成2的N次方的形式。例如：

15=2^0+2^1+2^2+2^3

并且我们发现，任意的十进制数都可以用2^n或2^n+2^m+……的形式表示出来，可以表示的单元数由使用的max n来决定。

可表示的单元数=2^(n+1)-1

二进制幂数加密法就是应用这个原理，由于英文字母只有26个字母，由公式可知，只要2的0、1、2、3、4次幂就可以表示31个单元。通过用二进制幂数表示字母序号数来加密。例如

明文：donotpullallyoureggsinonebasket

字母序号：4 15 14 15 20 16 21 12 12 1 12 12 25 15 21 18 5 7 7 19 9 14 15 14 5 2 1 19 11 5 20

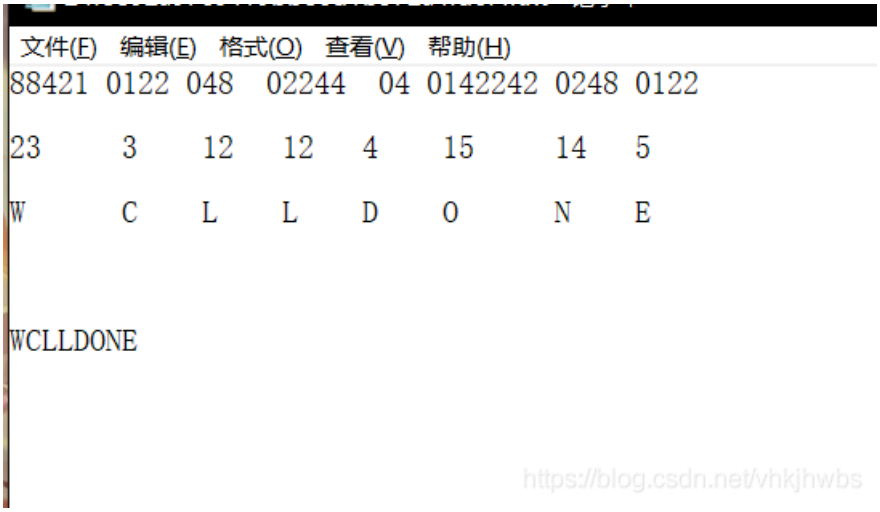
由于4=2^2 所以D加密过之后是2；15=2^0+2^1+2^2+2^3所以O加密后是0123。同理得到上述明文的加密后的密文

密文：2 0123/123 0123 24/4 024 23 23/0 23 23/034 0123 024 14/02 012 012 014/03 123 /0123 123 02/1 0 014 013 02 24

有点不太一样（因为这不是二进制的幂数加密）。之后搜到一种云影密码。是对01248组合构成的密文hhhh正好想要这个。

幂函数加密的密文 只有 01234 不可能有 8，所以表层不是 幂函数加密，

是 云影密码 ：



4,easy_ECC

直接上脚本：

```
import collections
import random

EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')

curve = EllipticCurve(
   'secp256k1',
   # Field characteristic.
```

```
   p=int(input('p=')),
   # Curve coefficients.
   a=int(input('a=')),
   b=int(input('b=')),
   # Base point.
   g=(int(input('Gx=')),
      int(input('Gy='))),
   # Subgroup order.
   n=int(input('k=')),
   # Subgroup cofactor.
   h=1,
)


# Modular arithmetic ##########################################################

def inverse_mod(k, p):
   """Returns the inverse of k modulo p.

  This function returns the only integer x such that (x * k) % p == 1.

  k must be non-zero and p must be a prime.
  """
   if k == 0:
       raise ZeroDivisionError('division by zero')

   if k < 0:
       # k ** -1 = p - (-k) ** -1 (mod p)
       return p - inverse_mod(-k, p)

   # Extended Euclidean algorithm.
   s, old_s = 0, 1
   t, old_t = 1, 0
   r, old_r = p, k

   while r != 0:
       quotient = old_r // r
       old_r, r = r, old_r - quotient * r
       old_s, s = s, old_s - quotient * s
       old_t, t = t, old_t - quotient * t

   gcd, x, y = old_r, old_s, old_t

   assert gcd == 1
   assert (k * x) % p == 1

   return x % p


# Functions that work on curve points #########################################

def is_on_curve(point):
   """Returns True if the given point lies on the elliptic curve."""
   if point is None:
       # None represents the point at infinity.
       return True

   x, y = point

   return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0
```

```
    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0


def point_neg(point):
    """Returns -point."""
    assert is_on_curve(point)

    if point is None:
        # -0 = 0
        return None

    x, y = point
    result = (x, -y % curve.p)

    assert is_on_curve(result)

    return result


def point_add(point1, point2):
    """Returns the result of point1 + point2 according to the group law."""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None

    if x1 == x2:
        # This is the case point1 == point2.
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # This is the case point1 != point2.
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

    x3 = m * m - x1 - x2
    y3 = y1 + m * (x3 - x1)
    result = (x3 % curve.p,
              -y3 % curve.p)

    assert is_on_curve(result)

    return result


def scalar_mult(k, point):
    """Returns k * point computed using the double and point_add algorithm."""
    assert is_on_curve(point)
```

```
    if k < 0:
        # k * point = -k * (-point)
        return scalar_mult(-k, point_neg(point))

    result = None
    addend = point

    while k:
        if k & 1:
            # Add.
            result = point_add(result, addend)

        # Double.
        addend = point_add(addend, addend)

        k >>= 1

    assert is_on_curve(result)

    return result


# Keypair generation and ECDHE ################################################

def make_keypair():
    """Generates a random private-public key pair."""
    private_key = curve.n
    public_key = scalar_mult(private_key, curve.g)

    return private_key, public_key



private_key, public_key = make_keypair()
print("private key:", hex(private_key))
print("public key: (0x{:x}, 0x{:x})".format(*public_key))
```

cyberpeace{19477226185390}

高手进阶区部分题题解

5, ENC

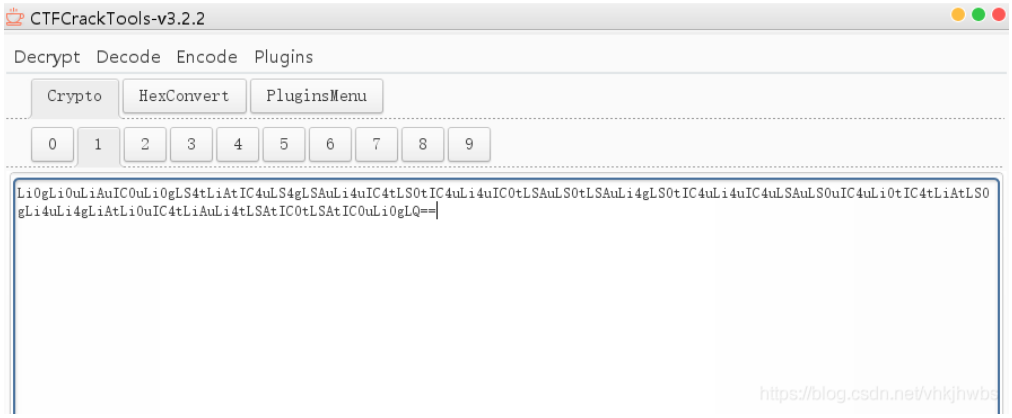
是一个没有后缀的文件，用winhex打开，发现 只有两种 字符串 ZERO 和 ONE

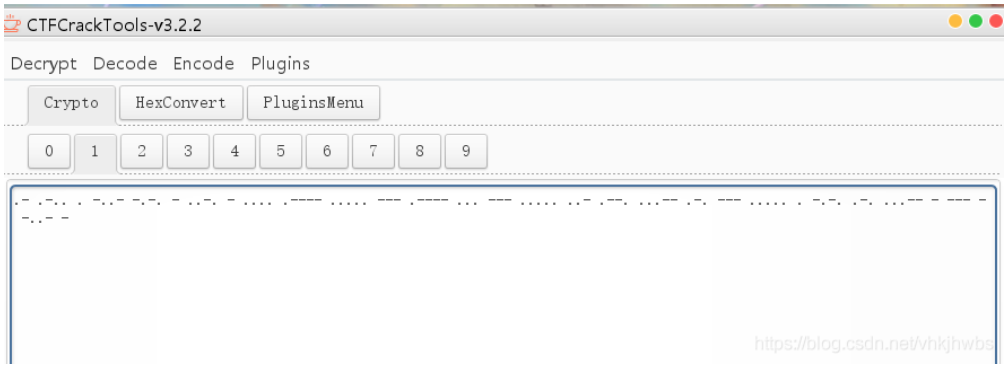
复制下来保存为 123.txt，然后用脚本将ZERO 替换为 0 ONE替换为 1：

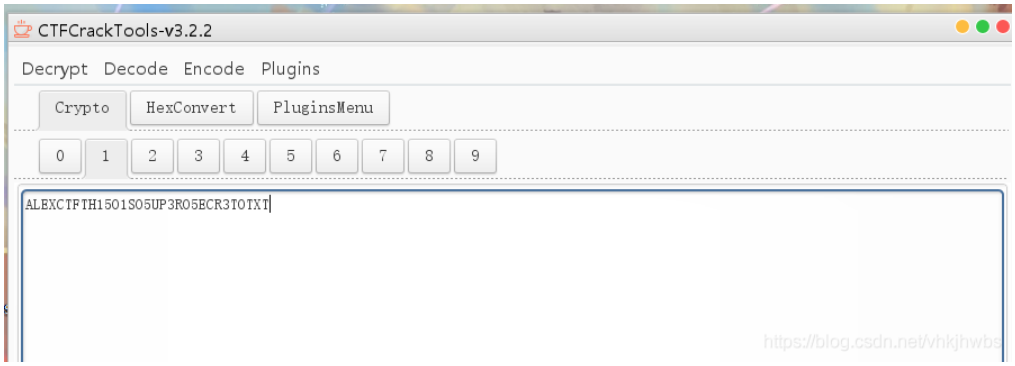
```
f = open("123.txt","r")
chars = f.read()
char = chars.split(" ")
string = ""
for c in char:
    if(c == "ZERO"):
        string += "0"
    elif(c == "ONE"):
        string +="1"
print(string)
f.close()
```

```
01001100011010010011000001100111010011000110100100110000011101010100110001101001010000010111010101001001010
```

然后转为 16进制 ， 再转为 asci，base64 摩斯密码







最后得到这个玩意，怎么提交都不对，TM的什么几把玩意，

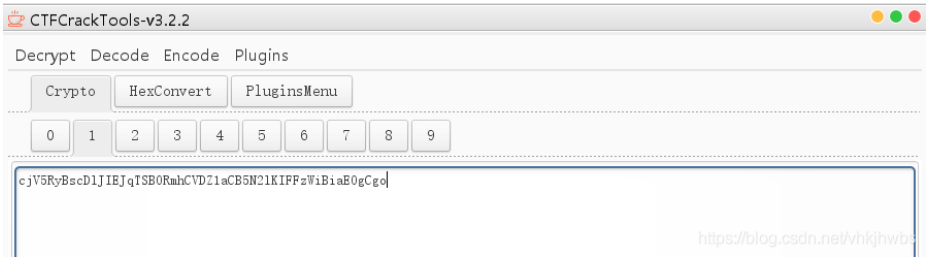
ALEXCTFTH15O1SO5UP3RO5ECR3TOTXT

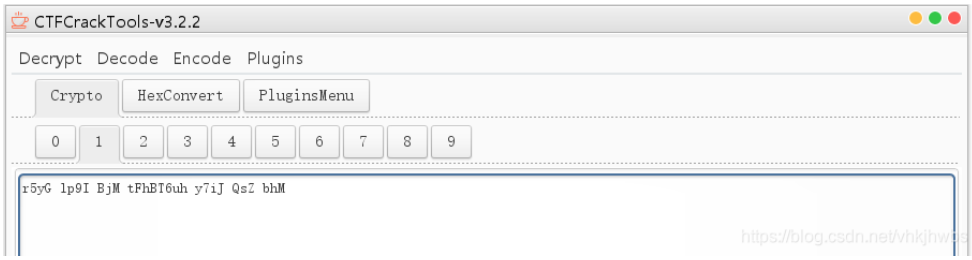
最后看了别人的writeup 说需要整理格式：

ALEXCTF{TH15_1S_5UP3R_5ECR3T_TXT}

6,告诉你个秘密

打开是两段十六进制，转为ascii 》 base64 》 键盘密码（我自己命名的，几个字符包围的字符）





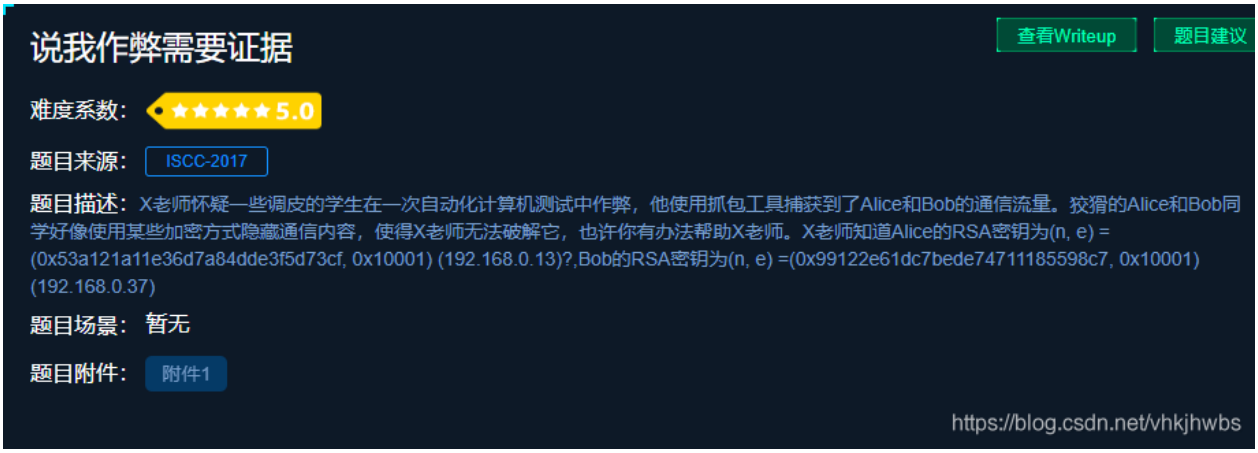
flag： TONGYUAN

7， Easy-one

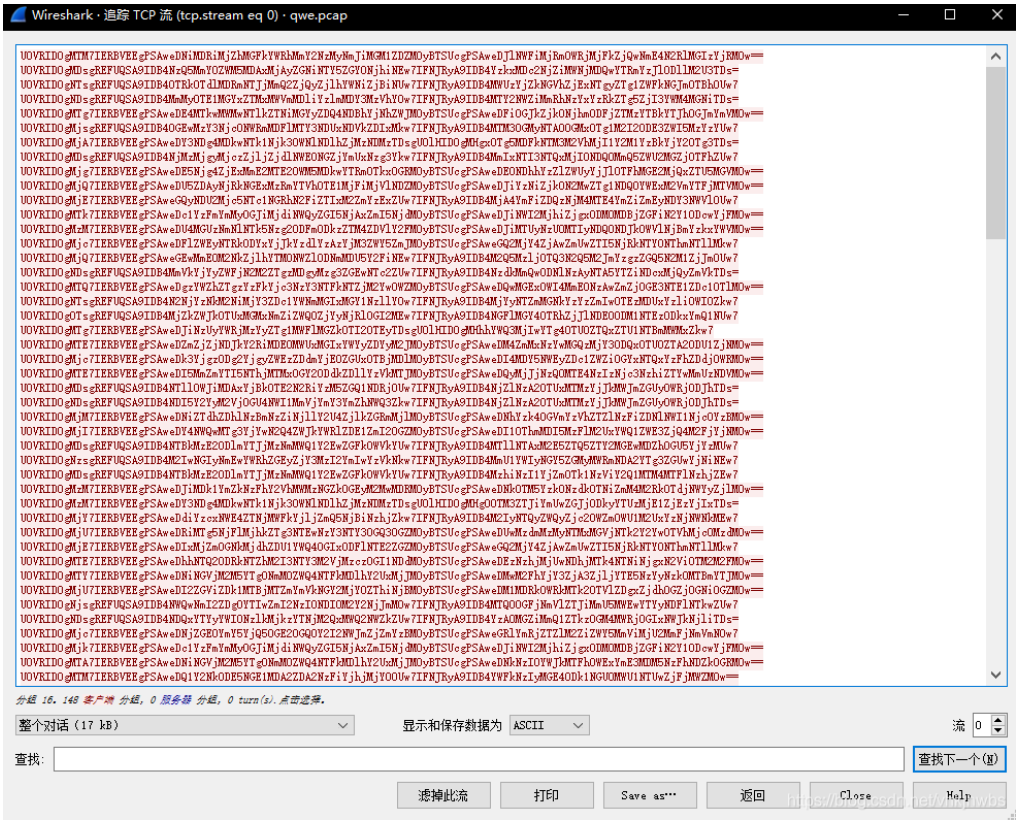
这个题我不会，附上一个大神的链接：

https://blog.csdn.net/zz_Caleb/article/details/89575430

8，说我作弊需要证据



追踪流：



是base64 ，找几条解密一下：

```
SEQ = 14; DATA = 0x83afae83c1db7776751d56c3f09fL; SIG = 0x400a19b82a4700ffc8a7515d7599L;

SEQ = 19; DATA = 0x75c1fbc28bb27b5d2db9601fb967L; SIG = 0x2b5b628bf8183400cdab7f5870b1L;

SEQ = 18; DATA = 0x181901c059de3b0f2d4840ab3aebL; SIG = 0x1b8bdf9468f81ce33a0da2a8bfbeL;

SEQ = 13; DATA = 0x3b04b26a0adada2f67326bb0c5d6L; SIG = 0x2e5ab24f9dc21df406a87de0b3b4L;
```

解码得到序列号，数据和 签名（签名用来校验信息）

猜测应该是解密数据部分，便可以得到flag。

现在思路就清晰了，

题目中给了两个公钥，一个公钥对应 加密的数据，另一个是用来 加密校验码的保证数据的准确性

序列号，肯定是 解密出字符 的位置

首先 求解 两个公钥 对应的 私钥： （其中的 n = q * p ,n分解为 p 和q 需要用到 yafu（kali自带） 里面的 factor 命令，linux和win都有yafu这个工具，命令 linux： factor（n）；win： yafu.exe factor(n)）

```
from Crypto.PublicKey
import RSA import gmpy
n = long(3104649130901425335933838103517383)
e = long(65537)
p = 49662237675630289
q = 62515288803124247
d = long(gmpy.invert(e, (p-1)*(q-1)))
rsa = RSA.construct( (n, e, d) )
```

然后就可以拿私钥解密数据了：

给出完整的脚本： 其中的 zero_one是 保存的 tcp流数据

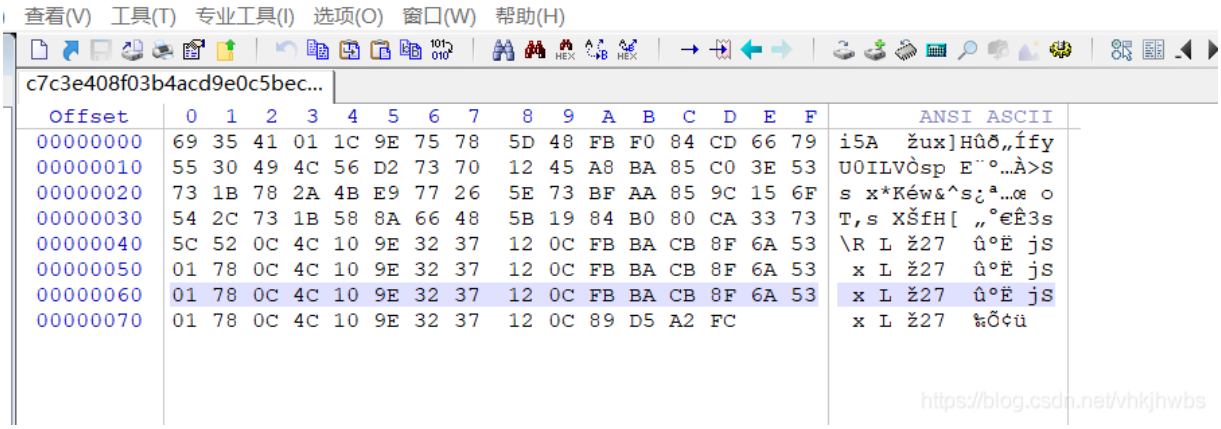
```
from Crypto.PublicKey import RSA
import gmpy2
import base64
#Alice's
A_n = 16962061390529489243049483333474767
A_p = 38456719616722997
A_q = 44106885765559411
#Bob's
B_n = 3104649130901425335933838103517383
B_p = 49662237675630289
B_q = 62515288803124247

A_phi = (A_p - 1) * (A_q - 1)
B_phi = (B_p - 1) * (B_q - 1)

e = 65537

A_d = int(gmpy2.invert(e, A_phi))
B_d = int(gmpy2.invert(e, B_phi))

A_rsa = RSA.construct( (A_n, e, A_d) )
B_rsa = RSA.construct( (B_n, e, B_d) )

flag = {}
with open('zero_one') as f:
    for s in f.readlines():
        line = str(base64.b64decode(s), encoding = 'utf8')
        seq = int(line.split(';')[0].split(' ')[2])
        data = int(line.split('0x')[1].split('L;')[0], 16)
        sig = int(line.split('0x')[2].rstrip('L;\n'), 16)
        decry = B_rsa.decrypt(data)
        signcheck = A_rsa.sign(decry, '')[0]

        if signcheck == sig:
            flag[seq] = chr(decry)
dic = sorted(flag.items(), key = lambda item:item[0])  #对字典按键值进行排序，返回值为列表
print(dic)
f = ''
for i in dic:
    f += i[1]
print(f)
```

拿到flag：flag{n0th1ng_t0_533_h3r3_m0v3_0n}

9.x_xor_md5

下载下来是一个无后缀名的文件，用hex打开看到十六进制的内容中有几行是重复的，加上题目是 异或 xor

怀疑 重复的内容就是 xor key



用 xor key 去和每一行的数据进行 异或运算，

```
['01', '78', '0C', '4C', '10', '9E', '32', '37', '12', '0C', 'FB', 'BA', 'CB', '8F', '6A', '53']
```

然后得到：

```
['68', '4d', '4d', '4d', '0c', '00', '47', '4f', '4f', '44', '00', '4a', '4f', '42', '0c', '2a',
'54', '48', '45', '00', '46', '4c', '41', '47', '00', '49', '53', '00', '4e', '4f', '54', '00',
'72', '63', '74', '66', '5b', '77', '45', '11', '4c', '7f', '44', '10', '4e', '13', '7f', '3c',
'55', '54', '7f', '57', '48', '14', '54', '7f', '49', '15', '7f', '0a', '4b', '45', '59', '20',
'5d', '2a', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00',
'00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00',
'00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00',
'00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '72', '6f', '69', '73']
```

再将其 转为字符：得到有ctf字样的乱码，说明思路没有出错，还需要进一步操作

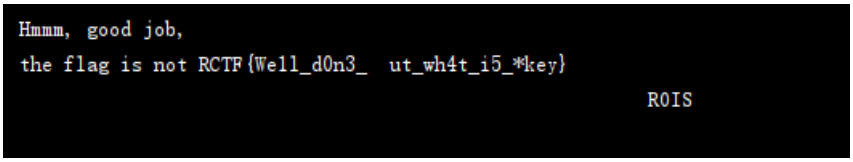
审查得到的 16 进制列表数据，发现 不应该出现0x00，0x00是绝对意义上的空，而空格 是 0x20

所以猜测应该是 所有数据还要和 0x20 进行 异或运算

得到，

```
['48', '6d', '6d', '6d', '2c', '20', '67', '6f', '6f', '64', '20', '6a', '6f', '62', '2c',
'a', '74', '68', '65', '20', '66', '6c', '61', '67', '20', '69', '73', '20', '6e', '6f',
'74', '20', '52', '43', '54', '46', '7b', '57', '65', '31', '6c', '5f', '64', '30', '6e',
'33', '5f', '1c', '75', '74', '5f', '77', '68', '34', '74', '5f', '69', '35', '5f', '2a',
'6b', '65', '79', '0', '7d', 'a', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '52', '4f', '49', '53']
```

转为字符：提交不对，看到 { }中间 u前面有一个空格，并且*key应该是 *key*



在 16 进制列表中先找到 key 前面的 字符 的16进制为 2a ， 确定key后面的字符也应该是 2a

而， 0x00与0x2a异或后 就是 0x2a ， 所以找到 字符u 前面的字符 1c 与 2a进行 异或运算得到 36

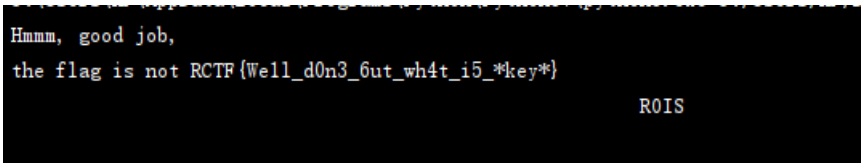
所以最后的结果 就是将 1c 改为 36 ， 将00 改为 2a

```
['48', '6d', '6d', '6d', '2c', '20', '67', '6f', '6f', '64', '20', '6a', '6f', '62', '2c', 'a', '74', '68', '65', '20', '66', '6c',
'61', '67', '20', '69', '73', '20', '6e', '6f', '74', '20', '52', '43', '54', '46', '7b', '57', '65', '31', '6c', '5f', '64', '30',
'6e', '33', '5f', '1c', '75', '74', '5f', '77', '68', '34', '74', '5f', '69', '35', '5f', '2a', '6b', '65', '79', '0', '7d', 'a', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '52', '4f', '49', '53']
```

改完之后得到：

```
['48', '6d', '6d', '6d', '2c', '20', '67', '6f', '6f', '64', '20', '6a', '6f', '62', '2c', '0a',
'74', '68', '65', '20', '66', '6c', '61', '67', '20', '69', '73', '20', '6e', '6f', '74', '20',
'52', '43', '54', '46', '7b', '57', '65', '31', '6c', '5f', '64', '30', '6e', '33', '5f', '36',
'75', '74', '5f', '77', '68', '34', '74', '5f', '69', '35', '5f', '2a', '6b', '65', '79', '2a',
'7d', '0a', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '20',
'20', '20', '20', '20', '20', '20', '20', '20', '20', '20', '52', '4f', '49', '53']
```

转为字符串：



提交还是不对，

题目提示还有 md5

而且字符串最后一行是一串不能打印的字符，怀疑最后一行的16进制 与md5有关，

将 最后一行的 16 进制复制下来 与 0x20 进行异或 再进行md5解码 得到 that

```
m = ['01','78','0C','4C','10','9E','32','37','12','0C','FB','BA','CB','8F','6A','53']
zz = []
for i in range(0,len(m)):
 x = int(m[i],16)
 zz.append(hex(x^32)[2:])
for i in zz:
    print(i,end="")
```

```
21582c6c30be1217322cdb9aebaf4a73
```

最后得到 flag：

```
RCTF{We1l_d0n3_6ut_wh4t_i5_that}
```

处理脚本：

```
file1 = "693541011C9E75785D48FBF084CD66795530494C56D273701245A8BA85C03E53731B782A4BE977265E73BFAA859C156F54

file = []
#先将文件的16进制字符串两个字符一组制成列表
for i in range(0,len(file1),2):
    file.append(file1[i:i+2])
xor1 = "01780C4C109E3237120CFBBACB8F6A53"
xor = []
#将 异或的对象制成一个列表
for i in range(0,len(xor1),2):
    xor.append(xor1[i:i+2])
result = []
string = ""
print(xor)
for i in range(0,len(file)):
    s = int("0x" + file[i],16) ^ int("0x" + xor[i%16],16) ^ int("0x20",16)
    string += chr(s)
    result.append(hex(s)[2:])
print(result)
print(string)
```

修改数据后的处理脚本：

```
n = ['48', '6d', '6d', '6d', '2c', '20', '67', '6f', '6f', '64', '20', '6a', '6f', '62', '2c', '0a','74', '
for i in n :
    print(chr(int(i,16)),end="")
```

10，easy_crypto

给了一个密文 enc.txt 和　一个加密的脚本

根据加密脚本，写出解密脚本

```
get buf unsign s[256]

get buf t[256]

we have key:hello world

we have flag:??????????????????????????????


for i:0 to 256

set s[i]:i

for i:0 to 256
    set t[i]:key[(i)mod(key.lenth)]

for i:0 to 256
    set j:(j+s[i]+t[i])mod(256)
        swap:s[i],s[j]

for m:0 to 37
    set i:(i + 1)mod(256)
    set j:(j + S[i])mod(256)
    swap:s[i],s[j]
    set x:(s[i] + (s[j]mod(256))mod(256))
    set flag[m]:flag[m]^s[x]

fprint flagx to file
```

解密脚本

```
# -*- coding: utf-8 -*-
f = open('C:/Users/HP/Desktop/enc.txt','r',encoding='ISO-8859-1')
c = f.read()
t = []
key = 'hello world'
ch = ''
j = 0 #初始化
s = list(range(256)) #创建有序列表
for i in range(256):
    j = (j + s[i] + ord(key[i % len(key)])) % 256
    s[i],s[j] = s[j],s[i]
i = 0 #初始化
j = 0 #初始化
for r in c:
    i = (i + 1)  % 256
    j = (j + s[i])  % 256
    s[i], s[j] = s[j], s[i]
    x = (s[i] + (s[j] % 256)) % 256
    ch += chr(ord(r) ^ s[x])
print(ch)
```

得到：

EIS{55a0a84f86a6ad40006f014619577ad3}

11，cr2-many-time-secrets

没思路，是多字节xor运算，

附上一篇 大牛的博客：https://pequalsnp-team.github.io/writeups/CR2

```
import binascii


def dec(msg, key):
    '''
    Simple char-by-char XOR with a key (Vigenere, Vernam, OTP)
    '''
    m = ""
    for i in range(0, len(key)):
        m += chr(msg[i] ^ ord(key[i]))
    return m


#######################################

lines = []

with open("msg", "r") as f:
    # Read lines from file and decode Hex
    ls = f.readlines()
    for l in ls:
        lines.append(binascii.unhexlify(l[:-1]))

# Step 1: Decode each line with the known key
k = "ALEXCTF{"
mes = []
for l in lines:
    m = dec(l, k)
    mes.append(m)
print(mes)

# Step 2: Guess some part of the first message 'Dear Fri'
k = "Dear Friend, "
m = dec(lines[0], k)
print(m)

# Step 3: Decode each line with the new known key
k = "ALEXCTF{HERE_"
mes = []
for l in lines:
    m = dec(l, k)
    mes.append(m)
print(mes)

# Step 4: Guess some part of the last message 'ncryption sc'
k = 'ncryption scheme '
```

```
m = dec(lines[-1], k)
print(m)

# Step 5: Decode each line with the new known key
k = "ALEXCTF{HERE_GOES_"
mes = []
for l in lines:
    m = dec(l, k)
    mes.append(m)
print(mes)

# Step 6: Guess all the second message 'sed One time pad e'
# the third message is 'n scheme, I heard '
# so we can retrive the complete key
k = 'sed One time pad encryptio'
m = dec(lines[2], k)
print(m)

'''
['Dear Fri', 'nderstoo', 'sed One ', 'n scheme', 'is the o', 'hod that', ' proven ', 'ever if ', 'cure, Le'
ALEXCTF{HERE_
['Dear Friend, ', 'nderstood my ', 'sed One time ', 'n scheme, I h', 'is the only e', 'hod that is m', ' pr
ALEXCTF{HERE_GOES
['Dear Friend, This ', 'nderstood my mista', 'sed One time pad e', 'n scheme, I heard ', 'is the only encry
ALEXCTF{HERE_GOES_THE_KEY}
'''
```

12，oldDevier

以前遇到过这种类型的题，是 RSA攻击方法中的一种，**RSA低加密指数广播攻击（模数n、密文c不同，加密指数e相同）**

至于什么是 **RSA低加密指数广播攻击，自行百度，**

如果想更详细的的掌握 RSA 的各种攻击方法，可以看看 这篇博客https://www.zhihu.com/people/ou-yang-shen-pai/posts

（python 2 ）

```
#!/usr/bin/python
#coding:utf-8

import gmpy2
import time
from Crypto.Util.number import long_to_bytes

file = [{"c": 736606757474117146172206513324291608049550591366325033008274746538367689397041147655074839484
{"c": 219628253233004691517959202898868865627909427715468585008421798065664357671038039788851487721393054 84
{"c": 656968942027406695783598339058358528657008761904811014118770058419379269523540507781154435516929038 23
{"c": 450824616804451351845249388271353639063674154155180582179033897379761597127186724858437981311412547 81
{"c": 229661056702912823355888430182441615527644863731179428659669040761911223374355425532767439388176867 29
{"c": 179633130634050457429681369162198383521355617853895343812629792645853978968444708790236865085403551 60
{"c": 165241753470902945038057065397370532098611767959756387302268314080050748256048294831013154094822779 70
{"c": 155857717344883510394566313940404977595686794295106192197661917808076753617418592904907324511126487 76
{"c": 896512342163769405004421684452337916334747802912481503283281322505073255852423966064874628488414074 67
{"c": 135609457565430230085293881084469408471378530384370952445730358885312885773708290656663200693978983 94

# 读入 e, n, c
c = []
n = []
for f in file:
    c.append(f["c"])
    n.append(f["n"])
e = 10


def CRT(items):
    N = reduce(lambda x, y: x * y, (i[1] for i in items))
    result = 0
    for a, n in items:
        m = N / n
        d, r, s = gmpy2.gcdext(n, m)
        if d != 1: raise Exception("Input not pairwise co-prime")
        result += a * s * m
    return result % N, N


data = zip(c, n)
x, n = CRT(data)
m = gmpy2.iroot(gmpy2.mpz(x), e)[0].digits()
print long_to_bytes(m)
```

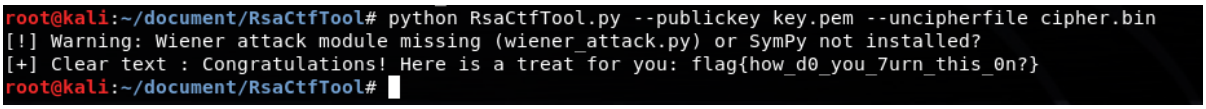
13，wtc_rsa_bbq

给了个密文和公钥，直接用 RsaCtfTool 直接爆破就行

kali没有，需要自行去 github下载使用（需要先安装依赖包）

命令：

```
python RsaCtfTool.py --publickey key.pem --uncipherfile cipher.bin
```



得到：flag{how_d0_you_7urn_this_0n?}

14，cr4-poor-rsa

给了个无后缀的文件，先用winhex查看一下，发现是一个压缩包，后缀改为 zip解压

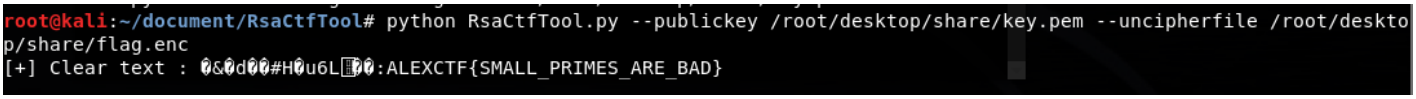
得到两个文件 flag.b64（密文） 和 key.pub（公钥）

先处理flag.b64（将flag.b64中的内容进行解 base64操作）

使用 notepad++ 打开 flag.b64文件使用 插件中的 MIME Tools 中的 base64 decode 将文件内容解密，然后另保存为 flag.enc 文件(这里不能直接复制粘贴保存，会丢失数据)

然后使用 RsaCtfTool 工具进行破解：

```
python RsaCtfTool.py --publickey key.pem --uncipherfile cipher.bin
```



得到flag:ALEXCTF{SMALL_PRIMES_ARE_BAD}

15,flag_in_your_hand1

给了两个 文件 index.html 和 一个js文件 ，考察js代码审计能力

首先借助浏览器来运行js 程序。用浏览器打开`index.html`，分析 js 代码: 首先无论在 token 输入框中输入什么字符串，`getFlag()` 都会算出一个 hash 值，

实际上是`showFlag()`函数中 ic 的值决定了 hash 值即 flag 是否正确。

那么在`script-min.js`中找到 ic 取值的函数 ck() ，找到一个 token 使得 ck()中`ic =true`即可。

token 是`[118, 104, 102, 120, 117, 108, 119, 124, 48,123,101,120]`每个数字减3 得到的ascii码所对应的字符，

即`security-xbu`

可以利用浏览器的js 调试功能跟踪变量，逻辑梳理的会更快一些

在 token 处输入`security-xbu` ，点击Get flag!

出现`You got the flag below!!`，

以及flag： `RenIbyd8Fgg5hawvQm7TDQ`

16,cr3-what-is-this-encryption

给了rsa解密的相关参数，直接用脚本解密就行了

```
import libnum
from Crypto.Util.number import long_to_bytes

c = 0x7fe1a4f743675d1987d25d38111fae0f78bbea6852cba5beda47db76d119a3efe24cb04b9449f53becd43b0b46e269826a983

e = int("0x6d1fdab4ce3217b3fc32c9ed480a31d067fd57d93a9ab52b472dc393ab7852fbcb11abbebfd6aaae8032db1316dc22d3

q = int("0xa6055ec186de51800ddd6fcbf0192384ff42d707a55f57af4fcfb0d1dc7bd97055e8275cd4b78ec63c5d592f567c6639
p = int("0xfa0f9463ea0a93b929c099320d31c277e0b0dbc65b189ed76124f5a1218f5d91fd0102a4c8de11f28be5e4d0ae91ab31
n = q*p

d = libnum.invmod(e, (p - 1) * (q - 1))
m = pow(c, d, n)   # m 的十进制形式
string = long_to_bytes(m)  # m明文
print(string)  # 结果为 b‘ m ’ 的形式
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python37\pyt
b'ALEXCTF{RS4_I5_E55ENT1AL_T0_D0_BY_H4ND}'

Process finished with exit code 0
```

flag： ALEXCTF{RS4_I5_E55ENT1AL_T0_D0_BY_H4ND}

17，safer_than_rot13

解压得到 cry100文件，复制文件内容尝试进行 rot13解密，没发现什么

在尝试 词频分析，利用在线工具：https://quipqiup.com/

得到flag：no_this_is_not_crypto_my_dear

```
Y MAY BE ??ST AS DAN?ERO?S. ALMOST ALL ARE COMMON-BORN, SIM?LE FOL? WHO HAD NEVER BEE
 WAR. YO?R FLA? IS "NO THIS IS NOT CRY?TO MY DEAR", IN LOWERCASE, WITH ?NDERSCORES IN

Y MAY BE ?UST AS ?ANGEROUS. ALMOST ALL ARE COMMON-BORN, SIM?LE FOL? WHO HA? NEVER BEE
) WAR. YOUR FLAG IS "NO THIS IS NOT CRY?TO MY ?EAR", IN LOWERCASE, WITH UN?ERSCORES I

Y MAY BE ?UST AS DANGEROUS. ALMOST ALL ARE ?OMMON-BORN, SIM?LE FOL? WHO HAD NEVER BEE
) WAR. YOUR FLAG IS "NO THIS IS NOT ?RY?TO MY DEAR", IN LOWER?ASE, WITH UNDERS?ORES I

Y MAY BE ?UST AS DANGEROUS. ALMOST ALL ARE COMMON-BORN, SIM?LE FOL? WHO HAD NE?ER BEE
) WAR. YOUR FLAG IS "NO THIS IS NOT CRY?TO MY DEAR", IN LOWERCASE, WITH UNDERSCORES I

Y MAY BE ?UST AS DANGEROUS. A?MOST A?? ARE COMMON-BORN, SIM??E FO?? WHO HAD NEVER BEE
```


18，flag_in_your_hand

给了两个 文件 index.html 和 一个js文件 ， 考察js代码审计能力

首先借助浏览器来运行js 程序。用浏览器打开`index.html`，分析 js 代码: 首先无论在 token 输入框中输入什么字符串，`getFlag()` 都会算出一个 hash 值，

实际上是`showFlag()`函数中 ic 的值决定了 hash 值即 flag 是否正确。

那么在`script-min.js`中找到 ic 取值的函数 ck() ，找到一个 token 使得 ck()中`ic =true`即可。

token 是`[118, 104, 102, 120, 117, 108, 119, 124, 48,123,101,120]`每个数字减3 得到的ascii码所对应的字符，

即`security-xbu`

可以利用浏览器的js 调试功能跟踪变量，逻辑梳理的会更快一些

在 token 处输入`security-xbu` ，点击Get flag!

出现`You got the flag below!!`，

以及flag：`RenIbyd8Fgg5hawvQm7TDQ`

19，Decode_The_File

这题跟 攻防世界的 misc中的 base64stego 一样，都是利用 base64加密的特性进行数据的加密填充

利用base64 加密的特性，将flag进行编码分割，填充在每一行的最后

（简要原理，ascii码是用8位二进制表示一个字符的，而base64是用6位二进制表示一个字符，将明文字符转化为二进制后再每6位划分成 一个 “字节”，然后将每个字节转化为一个字符，就变成了base64密文，而在base64的密文中加密是利用，每一段密文的最后4位二进制是不影响明文的，可以将flag转化为二进制后拆分隐藏在每一段的最后4位二进制中）

解密：

将文件重命名为 stego.txt

```
import base64

b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt', 'rb') as f:   #stego.txt 为在base64密文中加密后的密文
    flag = ''
    bin_str = ''
    for line in f.readlines():
        stegb64 = str(line, "utf-8").strip("\n")
        rowb64 = str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\n")
        offset = abs(b64chars.index(stegb64.replace('=', '')[-1]) - b64chars.index(rowb64.replace('=', '')[
        equalnum = stegb64.count('=')  # no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
for i in range(0, len(bin_str), 8):
    print(chr(int(bin_str[i:i + 8], 2)),end='')
```

刷不下去了，就到这儿吧