hackme.inndy-crypto题解

一叶飘零 / 2018-08-15 17:09:14 / 浏览数 3554 技术文章 技术文章 顶(1) 踩(0)

前言

休假在家，竟然早上6点自然醒，不如做点数学题？
于是心血来潮打开

```
https://hackme.inndy.tw/scoreboard/
```

既然刷完了web，今天也尽量把crypto也刷完XD
做的过程中发现网上很难搜到题解，于是有了这篇文章
注：web全题解

```
http://skysec.top/2018/01/07/hackme%E7%BD%91%E7%AB%99%E8%BE%B9%E5%81%9A%E8%BE%B9%E8%AE%B0%E5%BD%95/
https://www.anquanke.com/post/id/156377
```

easy

题目给了一串16进制

```
526b78425233745561476c7a49476c7a4947566863336b7349484a705a3268305033303d
```

解了之后发现是Base64，写个脚本即可拿到flag

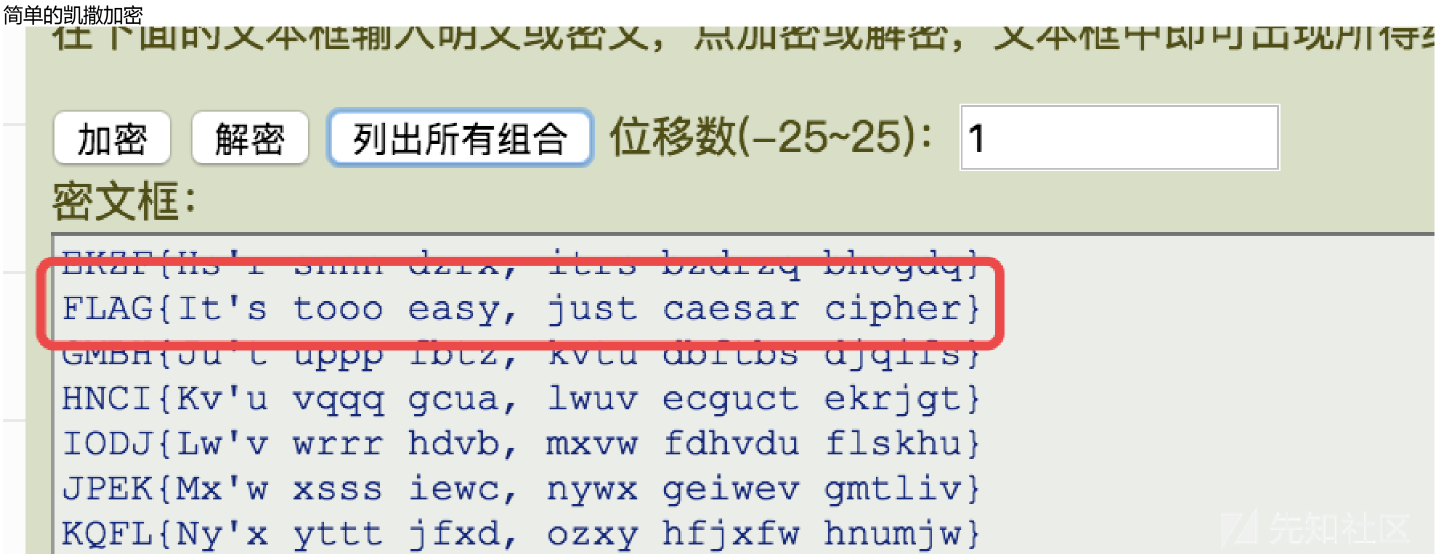
```
import base64
c = "526b78425233745561476c7a49476c7a4947566863336b7349484a705a3268305033303d"
print base64.b64decode(c.decode('hex'))
```

r u kidding

题目：

```
EKZF{Hs'r snnn dzrx, itrs bzdrzq bhogdq}
```

简单的凯撒加密



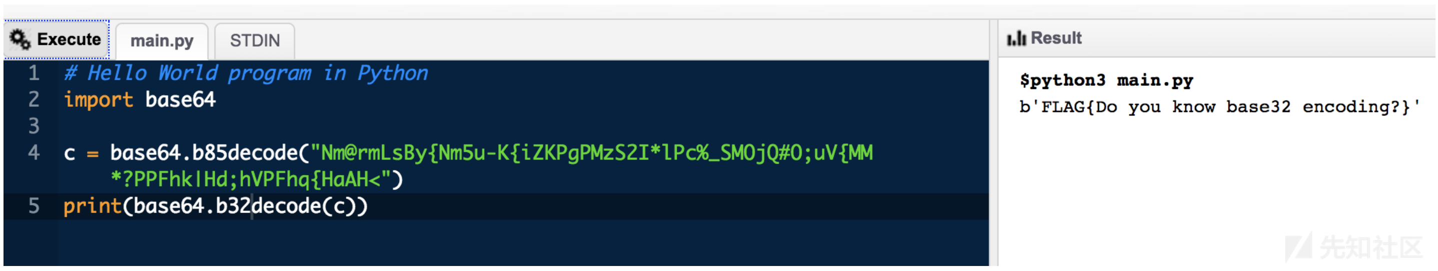
```
EKZF{Hs'r snnn dzrx, itrs bzdrzq bhogdq}
FLAG{It's tooo easy, just caesar cipher}
GMBH{Ju't uppp fbtz, kvtu dbftbs djqifs}
HNCI{Kv'u vqqq gcua, lwuv ecguct ekrjgt}
IODJ{Lw'v wrrr hdvb, mxvw fdhvdu flskhu}
JPEK{Mx'w xsss iewc, nywx geiwev gmtliv}
KQFL{Ny'x yttt jfxd, ozxy hfjxfw hnumjw}
```

not hard

题目信息：

```
Nm@rmLsBy{Nm5u-K{iZKPgPMzS2I*lPc%_SMOjQ#O;uV{MM*?PPFhk|Hd;hVPFhq{HaAH<
Tips: pydoc3 base64
```

随机想到py3的base85
于是尝试



```python
# Hello World program in Python
import base64

c = base64.b85decode("Nm@rmLsBy{Nm5u-K{iZKPgPMzS2I*lPc%_SM0jQ#0;uV{MM
*?PPFhk|Hd;hVPFhq{HaAH<")
print(base64.b32decode(c))
```

```
$python3 main.py
b'FLAG{Do you know base32 encoding?}'
```

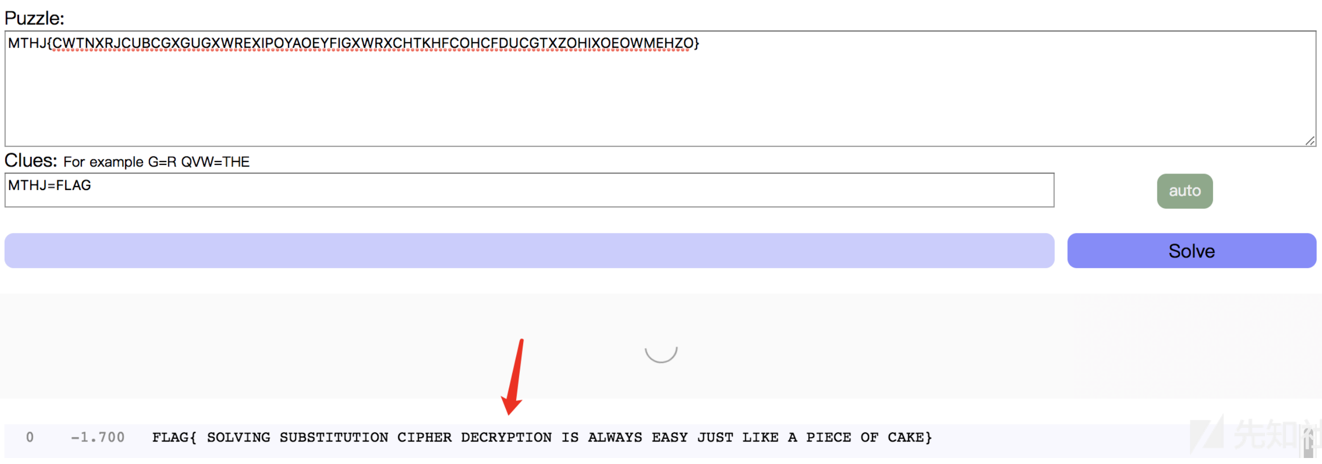
最后发现只是base85+base32
即可获得flag

classic cipher 1

题目如下

```
MTHJ{CWTNXRJCUBCGXGUGXWREXIPOYAOEYFIGXWRXCHTKHFCOHCFDUCGTXZOHIXOEOWMEHZO}
Solve this substitution cipher
```

直接凯撒遍历不行，于是直接使用工具



```
MTHJ{CWTNXRJCUBCGXGUGXWREXIPOYAOEYFIGXWRXCHTKHFCOHCFDUCGTXZOHIXOEOWMEHZO}
```

```
MTHJ=FLAG
```

```
0    -1.700    FLAG{ SOLVING SUBSTITUTION CIPHER DECRYPTION IS ALWAYS EASY JUST LIKE A PIECE OF CAKE}
```

可以得到flag
注：交的时候去掉空格

classic cipher 2

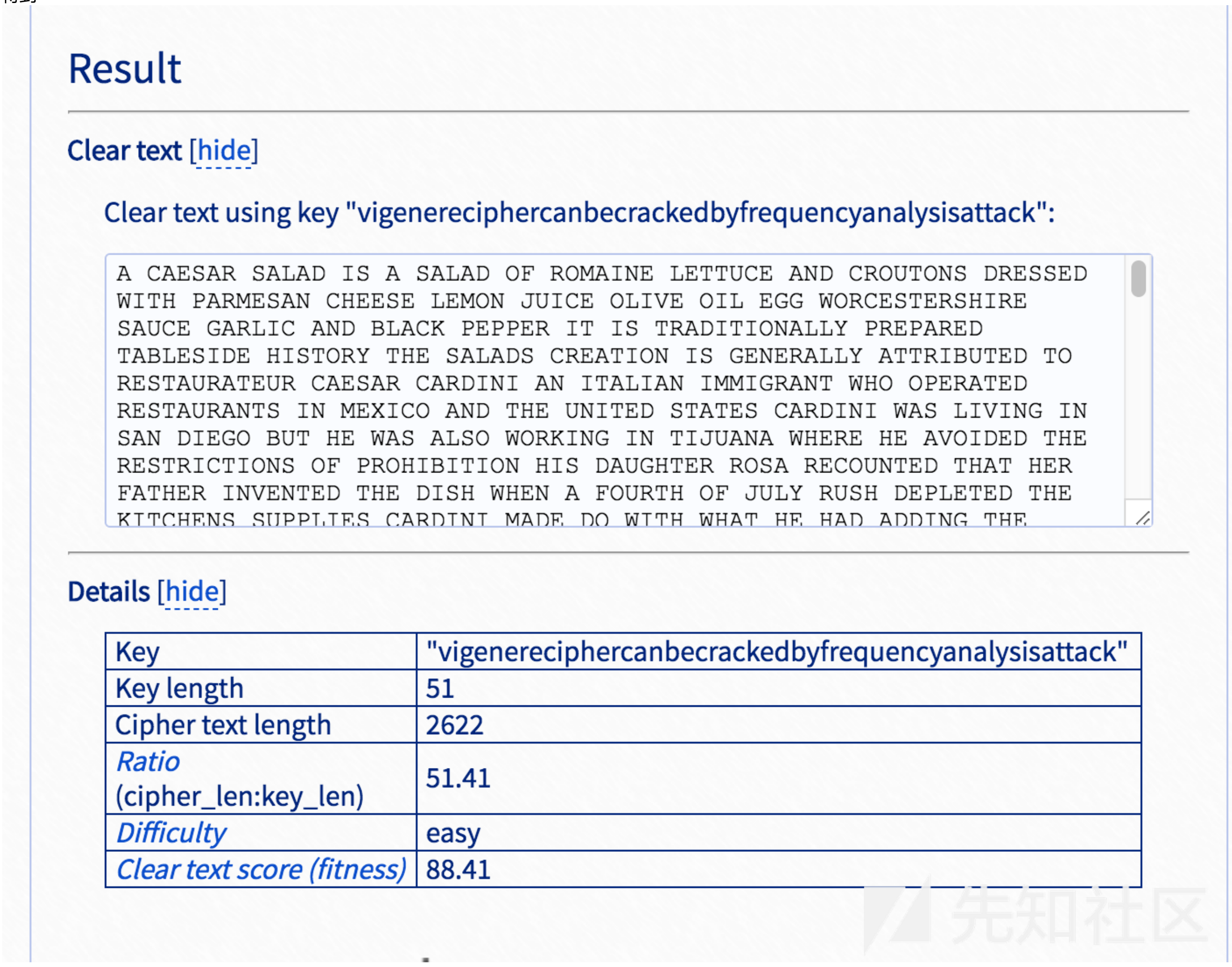
题目给了一个很长的vigenere cipher

```
V KGIFEI WCTPK MJ C SNMEF FF TYQDJLJ CIJNYPG YNQ CCMMBGNL WRGCNMJ AVXY TCZBLWRP CUFIUV LGWSQ KSNTI EFMIG MIY ERE OWJCXLTGBNPOVR WRYEM VHVCKC NOH DCAEU THQNJI MJ CW GTYDVTTMFIDLR
IRGZVZKH GESPGAXKI YKSGPVA KHG CEOBBX TVUUXVQL IF GPLWZSLER AVDMQHYGIU XQ ZTZXRWRNUIWI CCOWDS AFIHYHM NP GTNLTYF QEMBZRCXO ENS BTVVCBTK VVUTNVVCETU SR PFVNTS QHH GJC UAIECV ALAMXS
EKMLORV ARW NQKPRX KN FBR FZEIY FXU FJ NEI UPFQ UOEKTLY QF TBCUCXV ENIEI YI CDDPHVF TUF VGJTTSGWJMSJ SV JVBJGBVTTMF PAS WTUIROMX VBWR VGKDBRKGD GIEV YET PEWICW ZRLYRGGB TUE OGKP OHXG
A HYPZZL BJ AYNG GBWY FECMIVVD VRI NJRHYIDM WHRNLVED ASZVIGB MCNZ LU AVXY AJII OI YCD NEHKEG VRI GSYRRXYW JYCGR BF EFW BSBEXSKNZ BUWFMEK DG IOI TJES B RWDBGB SI DYWUMDCW FVYFS HLTW
ASIW MHCD OPKC VRMIPBTK XYG DVTL LLLKK GKJJI JEYX XUCR SUE SYV MSTXG A EKZAGV FECEF II JEIFIAJW TVSVKYUBLY NLUH WUG UAF A NFATV IG MHG C DV TIJWGERMG JSCWMAJWV UOTYXKZ INCKQFPRP
URBTP MX I UAXLAT MJVZEVRZRI ICJLFXIRT HKWFGBMQH DWFQ SUVQKLIF VPPKQGN MAE HVVO NEF RZRG EDYHJ CNQ ZSW EEGN XR BBI TYHFCOTYCRS EM XTSG TGD HVVO NEF WGEEMH HRU CLY JR WGPGBGDTC YYI
RCK SQMD EARC AV ZOEEYYYJLZLR GRIUIG ZECCDJJPN SE KXXUPBZTIT NS AGU YBRVCJA TY ZBLOYMMY WGIRO JWJZI ZVS NO MPKRKMEWF ATEGEWXVQL TUAE RSSWS TZEU DJ XXICEII CVS JSEVAVOW BFWKO PRUQ TW
KQLPVE PAJ OC QDQYHMEY EYYLRIQ IXKU KGVYKQNF SSORIPO EQDFTMMUM TNTKENSLL KQU CAXEUO JTOZR SZP XQCLKRT AAE TNVNVI SI CJFTO FYTCGP RRCTNWMVIM MHG PGIM MF ZZKGVTYI TKPUFV ERN DO GUBAPVH
RS JEGOURNNW SVSLRLIU KOBGGX ETGQZSPRX VO EPWC TATNMQJ RMV SHCKVPYL PAPQSZ KAETD WXGQQI UMJ FTWIOII CLRYW CMICDSUT QFCET XMQ PMT POYRSQF PBXCGC JN GRPLFZA BWL WCKGUU EPTHQFC IMYAFV
SIQRU DRBM EFW EGRVXSVOMANMEI JEWKT JEIFIAJ ACJ ORZSVFB YF YICRT CLCUOGGWA AN ABS UKGIJ MA XYI U KPYHZPIF EEWXHVOV VBGI KLQN XUG MRVGTLST JEVBPG SIKRYQIU AJWAL PVVTHDI NVAXOW ZIGHY
AULI ZGYNG TZ ZW TAFMXD DI OPK WGID EPL THXVP WVUL VYE HSRJFPX TSTXPRF CGTS LLV QLAEBAP YGQBI BMC FQBISIU EARTET URGCWLOEX RVU HSJ RPOQUNCV IFD FTRMOOMJ FL QRRA KDTTRPIRT XJV
TTKHHNYWB FHURQU AAEDTLAA UAXLAT MVZJMAMJ EPL IOI FTITJRCC CCOWDS BWVWICRT CPE NLW ADIAMXW TQ NVBK XB JVFTCPYC  KJOHHL VYEA GIUF MSCC HYKVUREEEO BWKSDXL LCDZZ GRQ QFVG BWHR R FOMFR
XRRKOXLFQ TW FENXYGB CNROGFQK DKXSUSIO GVR EMEKTPIPV VOQBC YZTJ FEUJMZJ MDAVRFGEATD KSVQ VTKICDDWTW BJ KLG APSEU GXVTX HFR GHEPQJJ SC JITCKLG N CLCKIJ STEAF GDBN KEMCPGL ROMTMEA
TXGRK QB WHBDTFH SYVGCGN ZEIGUIF RXLTCEMITXF QRC KUEYSMKSR PR KKEOC WXDF FJ WKVWGKRUGIYE LWJTBELC COZOTF JFV EZDBXFPS BS GQKILK GKFCXV JEL XUG NAEMPQSV ZETETJ MJVIIERJIFQI ALVTE VT
MPYETORW SGXB SV CRSGATVOY ZQ ASLFHNGVGI HEPXVVKI DJGRUIBOENCY HYYQE GS IEM YKT HPOZ CCYUSWD HK IOZMWVIEPP ACAWLH VIGFIINCS VRMV JQ F TSDWIEP UIGH XYFG KIFBLCB YZKWFMEKU BWHX RTE
RNYNJIHSIG XGYY IWAW GJMUTH RCFMJAEEY VRZ XN PRZVP KA IOSLIHG US DV AESHLD CSFYWB XB MGLY TSMKM TAVMETSV VKZRVKLGTTZW CCTRS ZGISKYRV PD YYI HYGVRC CNLW YL TWALM FQB WZOISPP GQWZLH
TQDQMIF VGIC SU QYXKIKLMMGB ETGD PWKAPXL MCI JUOX GLV IIO PUH GTOQVGG R CCOWDS TNEEYAVRVRE ZAYW NIJITMIQXN WL XUMJ WCTPK IOKSG ZSILRV SW VPKJKMCYW FWZSGIESLMV FHK TJO ZOMW GS
DEKVIHME C CEFEOP TGHXXSC FEH ENLRTQ CNLW DGZ MSBGG OKTWTRNMJI QQA HRU XIAFKCI
```

在线工具解密

```
https://www.guballa.de/vigenere-solver
```

得到



Result

Clear text [hide]

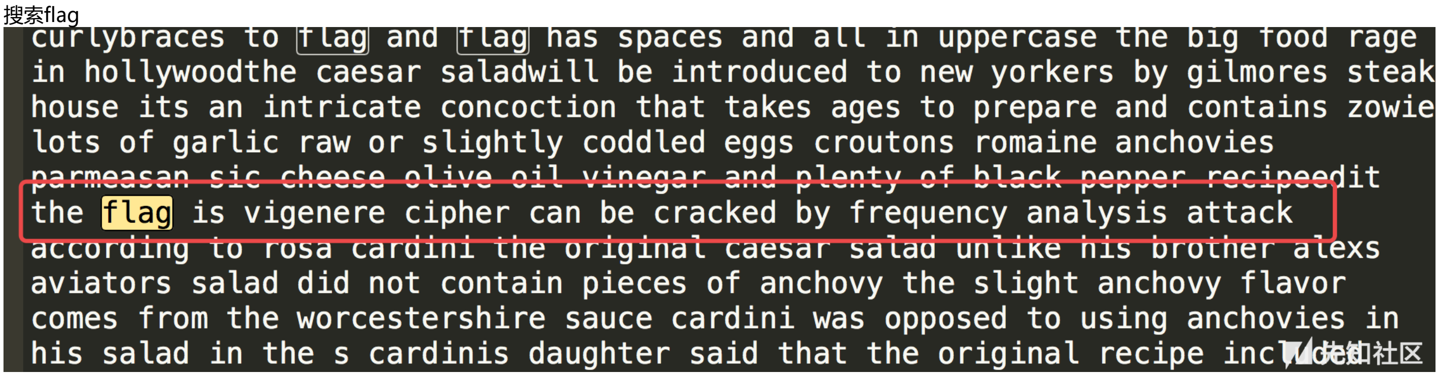
Clear text using key "vigenerecipherecanbecrackedbyfrequencyanalysisattack":

```
A CAESAR SALAD IS A SALAD OF ROMAINE LETTUCE AND CROUTONS DRESSED
WITH PARMESAN CHEESE LEMON JUICE OLIVE OIL EGG WORCESTERSHIRE
SAUCE GARLIC AND BLACK PEPPER IT IS TRADITIONALLY PREPARED
TABLESIDE HISTORY THE SALADS CREATION IS GENERALLY ATTRIBUTED TO
RESTAURATEUR CAESAR CARDINI AN ITALIAN IMMIGRANT WHO OPERATED
RESTAURANTS IN MEXICO AND THE UNITED STATES CARDINI WAS LIVING IN
SAN DIEGO BUT HE WAS ALSO WORKING IN TIJUANA WHERE HE AVOIDED THE
RESTRICTIONS OF PROHIBITION HIS DAUGHTER ROSA RECOUNTED THAT HER
FATHER INVENTED THE DISH WHEN A FOURTH OF JULY RUSH DEPLETED THE
KITCHENS SUPPLIES CARDINI MADE DO WITH WHAT HE HAD ADDING THE
```

Details [hide]

Key	"vigenerecipherecanbecrackedbyfrequencyanalysisattack"
Key length	51
Cipher text length	2622
Ratio (cipher_len:key_len)	51.41
Difficulty	easy
Clear text score (fitness)	88.41

搜索flag



```
curlybraces to flag and flag has spaces and all in uppercase the big food rage
in hollywoodthe caesar saladwill be introduced to new yorkers by gilmores steak
house its an intricate concoction that takes ages to prepare and contains zowie
lots of garlic raw or slightly coddled eggs croutons romaine anchovies
parmeasan sic cheese olive oil vinegar and plenty of black pepper recipeedit
the flag is vigenere cipher can be cracked by frequency analysis attack
according to rosa cardini the original caesar salad unlike his brother alexs
aviators salad did not contain pieces of anchovy the slight anchovy flavor
comes from the worcestershire sauce cardini was opposed to using anchovies in
his salad in the s cardinis daughter said that the original recipe inc
```

得到答案

easy AES

下载后发现是.xz结尾
于是

```
xz -d 1.py.xz
```

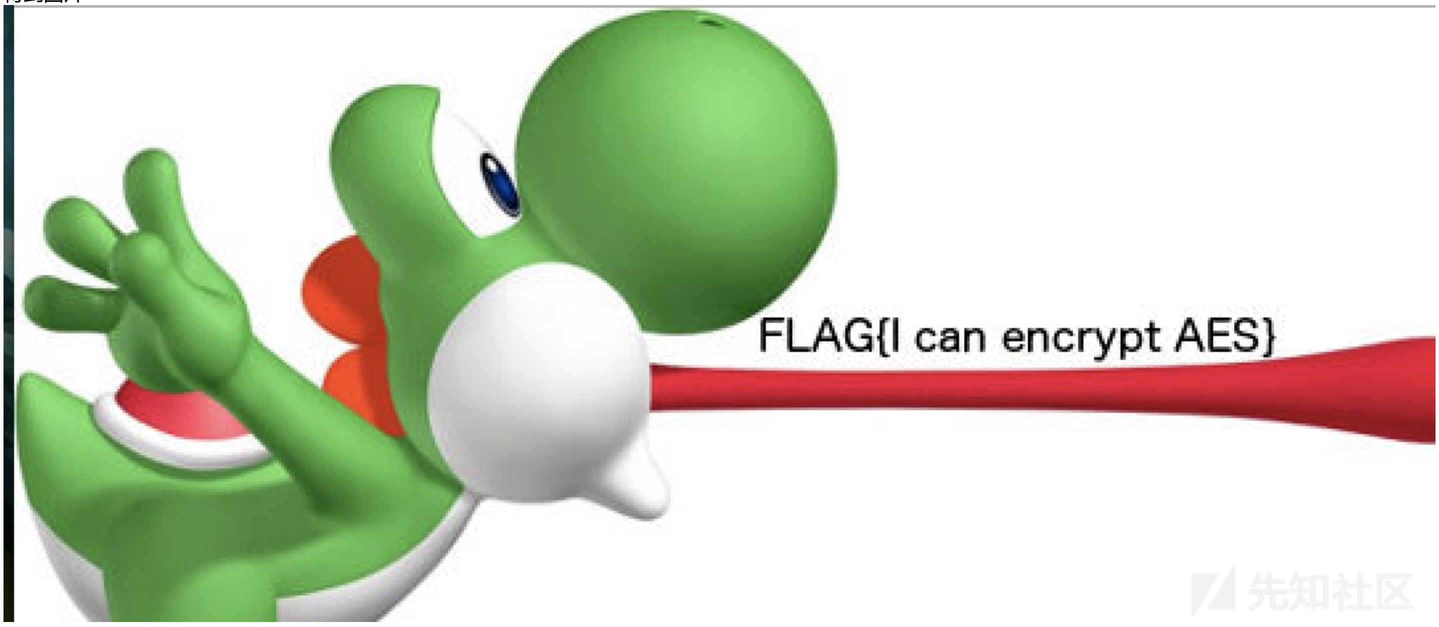
即可得到1.py

```
#!/usr/bin/env python3
import base64
from Crypto.Cipher import AES  # pip3 install pycrypto

def main(data):
    c = AES.new(b'Hello, World...!')
    plain_text = bytes.fromhex(input('What is your plain text? '))
    if c.encrypt(plain_text) != b'Good Plain Text!':
        print('Bad plain text')
        exit()

    c2 = AES.new(plain_text[::-1], mode=AES.MODE_CBC, IV=b'1234567887654321')

    decrypted = c2.decrypt(data)

    with open('output.jpg', 'wb') as fout:
        fout.write(decrypted)

main(base64.b64decode('.......'))
```

思路相当清晰：
1.第一轮密钥为`b'Hello, World...!'`
2.第一轮密文为`b'Good Plain Text!'`
3.解密即可得到`plain_text`
4.第二轮密钥为`plain_text`
5.直接解密输出图片即可
代码如下

```
import base64
from Crypto.Cipher import AES
c = AES.new(b'Hello, World...!')
plain_text = c.decrypt(b'Good Plain Text!')
c2 = AES.new(plain_text[::-1], mode=AES.MODE_CBC, IV=b'1234567887654321')
data = base64.b64decode('.......')
decrypted = c2.decrypt(data)
with open('output.jpg', 'wb') as fout:
    fout.write(decrypted)
```

得到图片



即可获得flag

one time padding

看到代码

```
<?php

/*
* one time padding encryption system
*
* we generate {$r = random_bytes()} which {strlen($r) == strlen($plaintext)}
* and encrypt it with {$r ^ $plaintext}, so no body can break our encryption!
*/

// return $len bytes random data without null byte
function random_bytes_not_null($len)
{
    $result = '';
    for($i = 0; $i < $len; $i++)
        $result .= chr(random_int(1, 255));
    return $result;
}

if(empty($_GET['issue_otp'])) {
    highlight_file(__file__);
    exit;
}

require('flag.php');

header('Content-Type: text/plain');

for($i = 0; $i < 20; $i++) {
    // X ^ 0 = X, so we want to avoid null byte to keep your secret safe :)
    $padding = random_bytes_not_null(strlen($flag));
    echo bin2hex($padding ^ $flag)."\n";
}
```

注意到每次加密都是使用`random_bytes_not_null`生成随机的key，然后与flag进行异或，正面突破显然无望
但是我们注意到一段注释

```
// X ^ 0 = X, so we want to avoid null byte to keep your secret safe :)
```

题目意思为随机key中不会存在0，那么意味着不会出现flag中的原字母
那么我们反过来想，只要爆破每一位，每一位从未出现过的，即flag
所以写出脚本如下：

```
import requests
import re
from bs4 import BeautifulSoup

url = "https://hackme.inndy.tw/otp/?issue_otp=a"

res_list = [[True] * 256 for i in range(50)]

for i in range(300):
    print i,res_list
    r = requests.get(url)
    soup = BeautifulSoup(r.text, "html.parser")
    text = str(soup)
    c_list = re.findall("[^\n]*\n", text)
    for j in c_list:
        j = j.replace('\n','')
        for k in range(1, len(j)/2+1):
            char_hex = "0x" + j[k * 2 - 2: k * 2]
            char_int = int(char_hex, 16)
            res_list[k - 1][char_int] = False

flag = ""
for i in range(50):
    for j in range(256):
        if res_list[i][j]:
            flag += chr(j)

print flag
```

其实最外层循环100次左右就够了，怕有人很非，所以写了300次= =
要是二维数组True不止50个。。对不起，你是大非酋。。。写1000吧
最后得到flag

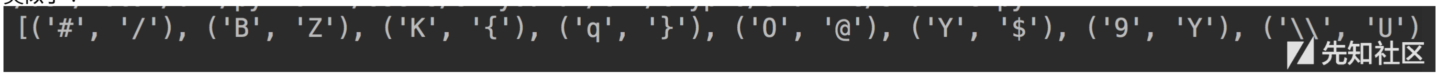
shuffle

拿到代码

```
import random
import string
```

```
characters = ''.join(map(chr, range(0x20, 0x7f)))

with open('plain.txt', 'r') as fin:
   plaintext = fin.read()

mapping = list(characters)
random.shuffle(mapping)
mapping = ''.join(mapping)

T = str.maketrans(characters, mapping)

with open('crypted.txt', 'w') as fout:
   fout.write(plaintext.translate(T))

plain = list(plaintext)
random.shuffle(plain)
suffled_plaintext = ''.join(plain)

with open('plain.txt', 'w') as frandom:
   frandom.write(suffled_plaintext)
```

代码很清晰：
1.将明文随机替换加密，保存为crypted.txt
2.将明文打乱，保存为plain.txt
故此我们只要根据plain.txt和crypted.txt计算出字频和对应的字符
然后替换一遍即可
类似于：

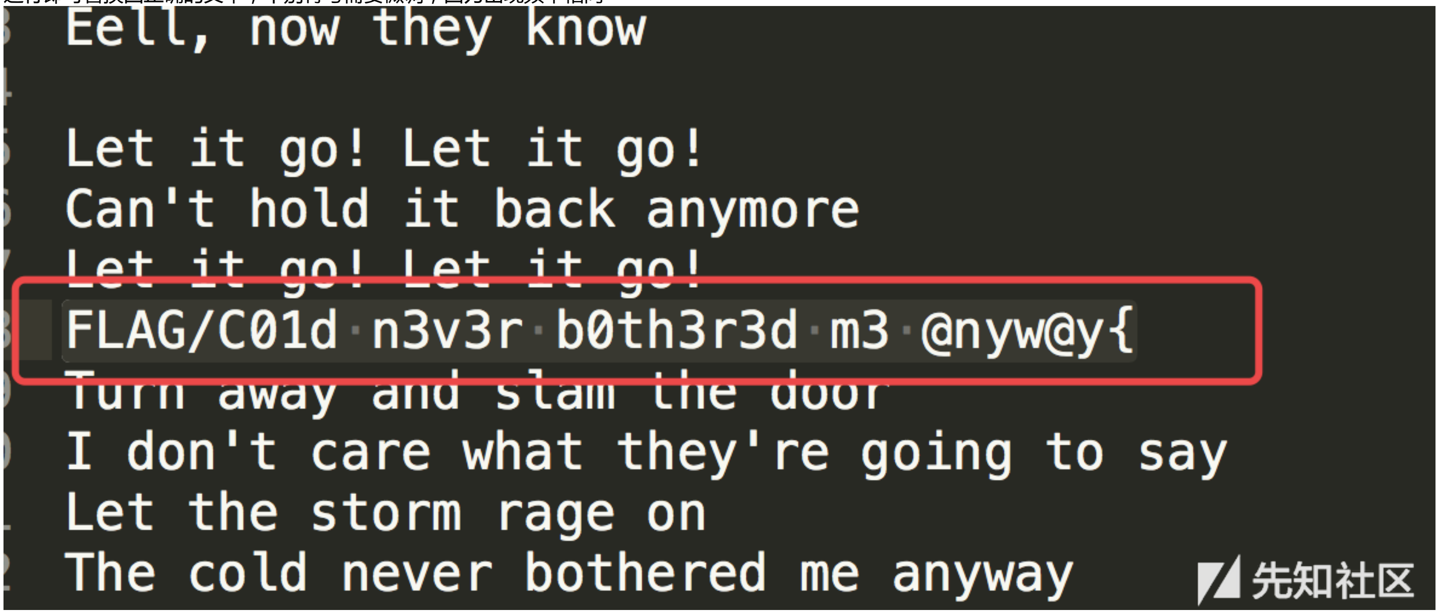


```
[('#', '/'), ('B', 'Z'), ('K', '{'), ('q', '}'), ('O', '@'), ('Y', '$'), ('9', 'Y'), ('\\', 'U')
```

随机写了个冗余的代码（想到哪里写到哪里= =别介意，没优化）

```
f1 = open('./crypted.txt')
cry_list=[0 for i in range(300)]
cry_content = f1.read()
list1=[]
for i in cry_content:
   cry_list[ord(i)]+=1
for i in range(len(cry_list)):
   if cry_list[i]!=0:
       list1.append((cry_list[i],i))
list2=[]
f2 = open('./plain.txt')
plain_list=[0 for i in range(300)]
plain_content = f2.read()
for i in plain_content:
   plain_list[ord(i)]+=1
for i in range(len(plain_list)):
   if plain_list[i]!=0:
       list2.append((plain_list[i],i))
res1 = sorted(list1)
res2 = sorted(list2)
res = []
for i in range(len(res1)):
   cry_chr = chr(int(res1[i][1]))
   plain_chr = chr(int(res2[i][1]))
   res.append((cry_chr,plain_chr))
f3 = open('./crypted.txt')
flag_content = f3.read()
res_content = ""
for i in flag_content:
   flag = False
   for j in range(len(res)):
       if i == res[j][0]:
           res_content+=res[j][1]
           flag = True
           break
   if flag == False:
       res_content+=i
print res_content
```

运行即可替换回正确的文本，个别符号需要微调，因为出现频率相同


```
Eell, now they know

Let it go! Let it go!
Can't hold it back anymore
Let it go! Let it go!
FLAG/C01d n3v3r b0th3r3d m3 @nyw@y{
Turn away and slam the door
I don't care what they're going to say
Let the storm rage on
The cold never bothered me anyway
```

故此得到flag

login as admin 2

拿到源码分析一下，看到关键函数

```
function load_user()
{
   global $secret, $error;

   if(empty($_COOKIE['user'])) {
       return null;
   }

   list($sig, $serialized) = explode('#', base64_decode($_COOKIE['user']), 2);

   if(md5(md5($secret).$serialized) !== $sig) {
       $error = 'Invalid session';
       return false;
   }

   parse_str($serialized, $user);
   return $user;
}
```

发现需要

```
md5(md5($secret).$serialized) === $sig
```

而

```
$serialized = http_build_query($user);
$sig = md5(md5($secret).$serialized);
$all = base64_encode("{$sig}#{$serialized}");
setcookie('user', $all, time()+3600);
```

现在我们的cookie中，user为

```
NmJjYjljOTE1NTk3NWE1M2U5NTFiMGI1MGYxMzc0ODAjbmFtZT1ndWVzdCZhZG1pbj0w
```

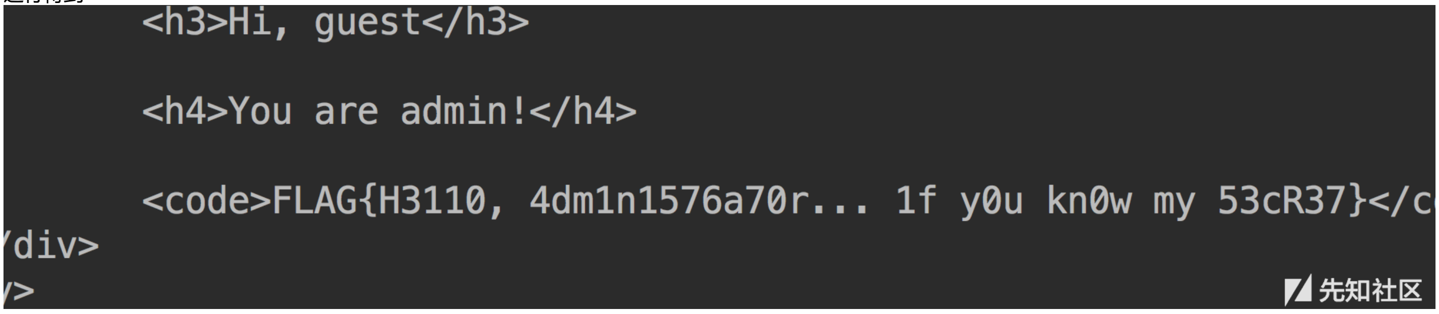
解码

```
6bcb9c9155975a53e951b0b50f137480#name=guest&admin=0
```

如此一来：
1.我们知道md5(salt.data)的值即sig
2.我们可以控制data
3.哈希长度拓展攻击即可
于是构造脚本

```
import hashpumpy
import base64
import requests

url = 'https://hackme.inndy.tw/login2/'
tmp = hashpumpy.hashpump('6bcb9c9155975a53e951b0b50f137480', 'name=guest&admin=0', 'name=guest&admin=1', 32)
payload = base64.b64encode(tmp[0]+'#'+tmp[1])
cookie = {
   'user':payload
}
r =requests.get(url=url,cookies=cookie)
print r.content
```

运行得到



```
        <h3>Hi, guest</h3>

        <h4>You are admin!</h4>

        <code>FLAG{H3110, 4dm1n1576a70r... 1f y0u kn0w my 53cR37}</c
/div>
/>
```

即可获得flag

login as admin 5

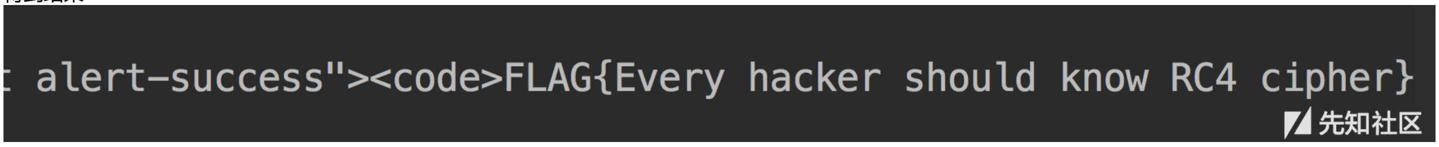
我们看到关键代码

```
function set_session($user)
{
   global $cipher;
   $cookie = base64_encode($cipher->encrypt(json_encode($user)));
   setcookie('user5', $cookie, time() + 60 * 60, '/', 'hackme.inndy.tw', true, true);
}
function restore_session()
{
   global $cipher;
   global $user;
   $data = $cipher->decrypt(base64_decode($_COOKIE['user5']));
   $user = json_decode($data, true);
}
```

发现加解密都用的rc4，然后明文直接使用了json
而我们可以知道json，又知道密文，那么可以反推rc4生成的流密钥
然后利用生成的流密钥，即可伪造消息
脚本如下

```
import base64
import urllib
import requests

c = base64.b64decode('U/osUbnY8nSrWz4WPwKSwWPzKq9tOIQ9eCWnN5E+')
plain = '{"name":"guest","admin":false}'
res = ''
for i in range(len(c)):
   res += chr(ord(c[i])^ord(plain[i]))
need = '{"name":"guest","admin":true}'
payload = ''
for i in range(len(need)):
   payload += chr(ord(need[i])^ord(res[i]))
payload = urllib.quote(base64.b64encode(payload))
cookie = {
   'user5':payload
}
url = "https://hackme.inndy.tw/login5/"
r = requests.get(url=url,cookies=cookie)
print r.content
```

得到结果



```
t alert-success"><code>FLAG{Every hacker should know RC4 cipher}
```

xor

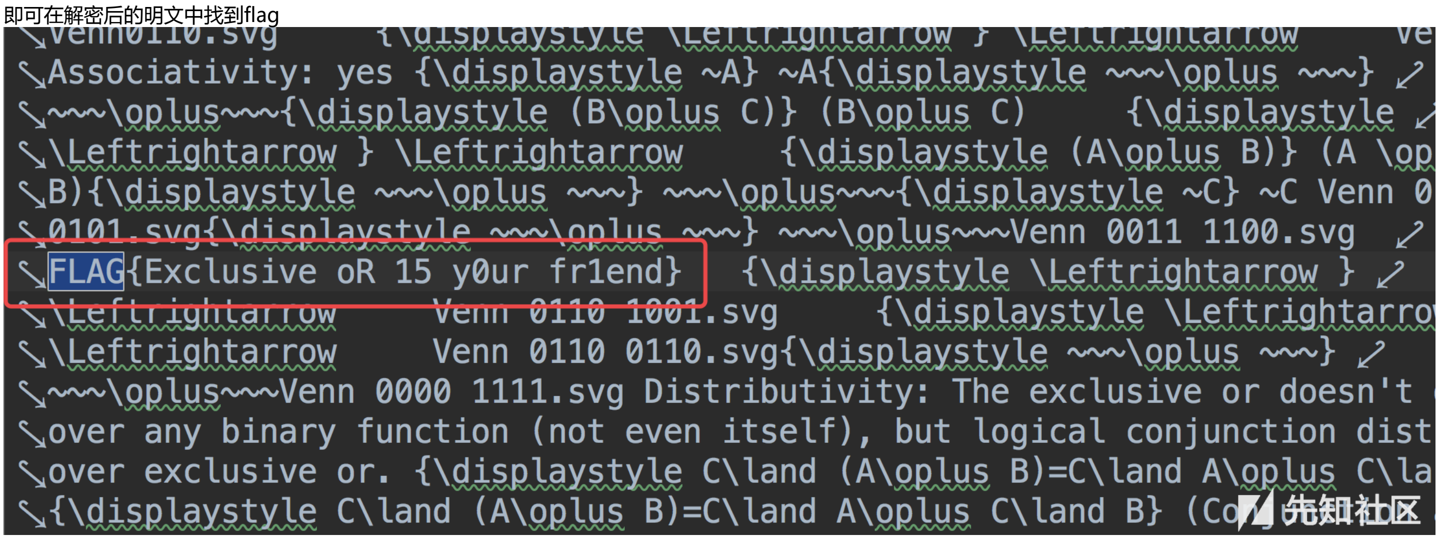
运行github开源的xortool脚本

```
G:\python2.7\Scripts>python xortool -c 20 xor
The most probable key lengths:
  1:   8.6%
  3:   10.6%
  6:   9.4%
  9:   21.8%
 12:   7.1%
 15:   6.2%
 18:   14.1%
 27:   9.7%
 36:   7.1%
 45:   5.4%
Key-length can be 3*n
1 possible key(s) of length 9:
hackmepls
Found 1 plaintexts with 95.0%+ printable characters
See files filename-key.csv, filename-char_used-perc_printable.csv
```

得到key：hackmepls
运行脚本解密

```
f1 = open("xor","rb")
key = "hackmepls"
f3 = open("flagtest.txt","wb")
# key = f2.read().replace(" ", "")
# key = "47 6F 6F 64 4C 75 63 6B 54 6F 59 6F 75".replace(" ", "").decode("hex")
flag = f1.read()
flag_length = len(flag)
key_length = len(key)
flag_res = ""
for i in range(0,flag_length):
   xor_str = chr(ord(flag[i])^ord(key[i%key_length]))
   flag_res += xor_str
f3.write(flag_res)
f3.close()
```

即可在解密后的明文中找到flag



```
Venn0110.svg    {\displaystyle \Leftrightarrow } \Leftrightarrow    Ve
Associativity: yes {\displaystyle ~A} ~A{\displaystyle ~~~\oplus ~~~}
~~~\oplus~~~{\displaystyle (B\oplus C)} (B\oplus C)     {\displaystyle
\Leftrightarrow } \Leftrightarrow     {\displaystyle (A\oplus B)} (A \op
B){\displaystyle ~~~\oplus ~~~} ~~~\oplus~~~{\displaystyle ~C} ~C Venn 0
0101.svg{\displaystyle ~~~\oplus ~~~} ~~~\oplus~~~Venn 0011 1100.svg
FLAG{Exclusive oR 15 y0ur fr1end}   {\displaystyle \Leftrightarrow }
\Leftrightarrow     Venn 0110 1001.svg     {\displaystyle \Leftrightarro
\Leftrightarrow     Venn 0110 0110.svg{\displaystyle ~~~\oplus ~~~}
~~~\oplus~~~Venn 0000 1111.svg Distributivity: The exclusive or doesn't
over any binary function (not even itself), but logical conjunction dist
over exclusive or. {\displaystyle C\land (A\oplus B)=C\land A\oplus C\la
{\displaystyle C\land (A\oplus B)=C\land A\oplus C\land B} (C
```

得到flag

emoji

拿到题目后丢进

```
https://tool.lu/js/
```

解密，得到

```
console.log((function() {
   if (typeof(require) == 'undefined') return '(´■ω■`)';
   var code = require('process').argv[2];
   if (!code) return '(´■ω■`)';
   String.prototype.zpad = function(l) {
       return this.length < l ? '0' + this.zpad(l - 1) : this
   };

   function encrypt(data) {
       return '"' + (Array.prototype.slice.call(data).map((e) = > e.charCodeAt(0)).map((e) = > (e * 0xb1 + 0x1b) & 0xff).map((
   }
   var crypted = ".......";
   if (JSON.parse(encrypt(code)) != crypted) return '(´■ω■`)';
   try {
       eval(code)
   } catch (e) {
       return '(´■ω■`)'
   }
   return '(*´∀`)~♥'
})())
```

观察到关键代码

```
var crypted = ".......";
   if (JSON.parse(encrypt(code)) != crypted) return '(´■ω■`)';
   try {
       eval(code)
   } catch (e) {
       return '(´■ω■`)'
   }
```

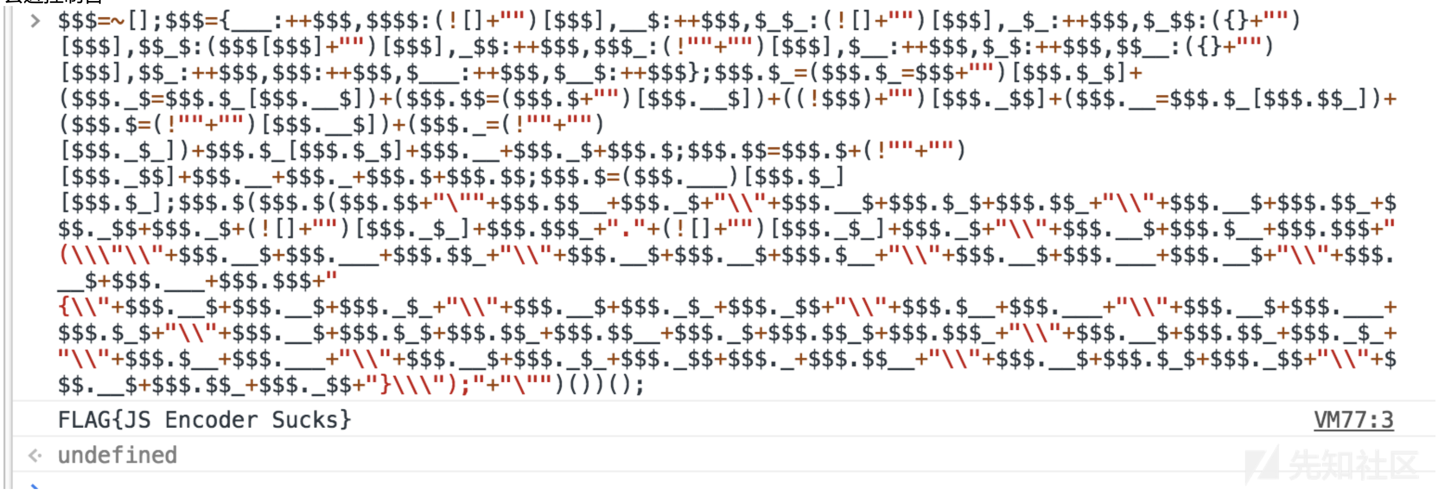
关键点应该是解密crypted去得到code
跟到加密函数，发现直接爆破即可，于是写出脚本

```
def crack(n):
   for i in range(256):
       if (i * 0xb1 + 0x1b) & 0xff == n:
           return i

crypted=u'......'
res = [crack(ord(i)) for i in crypted]
code = ''
for j in res:
   code += chr(j)
print code
```

得到代码

```
$$$=~[];$$$={___:++$$$,$$$$:(![]+"")[$$$],__$:++$$$,$_$_:(![]+"")[$$$],_$_:++$$$,$_$$:({}+"")[$$$],$$_$:($$$[$$$]+"")[$$$],_$$
```

丢进控制台



```
> $$$=~[];$$$={___:++$$$,$$$$:(![]+"")[$$$],__$:++$$$,$_$_:(![]+"")[$$$],_$_:++$$$,$_$$:({}+"")
  [$$$],$$_$:($$$[$$$]+"")[$$$],_$$:++$$$,$$$_:(!""+"")[$$$],$__:++$$$,$_$:++$$$,$$__:({}+"")
  [$$$],$$_:++$$$,$$$:++$$$,$___:++$$$,$__$:++$$$};$$$.$_=($$$.$_=$$$+"")[$$$.$_$]+
  ($$$._$=$$$.$_[$$$.__$])+($$$.$$=($$$.$+"")[$$$.__$])+((!$$$)+"")[$$$._$$]+($$$.__=$$$.$_[$$$.$$_])+
  ($$$.$=(!""+"")[$$$.__$])+($$$._=(!""+"")
  [$$$._$_])+$$$.$_[$$$.$_$]+$$$.__+$$$._$+$$$.$;$$$.$$=$$$.$+(!""+"")
  [$$$._$$]+$$$.__+$$$._+$$$.$+$$$.$$;$$$.$=($$$.___)[$$$.$_]
  [$$$.$_];$$$.$($$$.$($$$.$$+"\""+$$$.$$__+$$$._$+"\\"+$$$.__$+$$$.$_$+$$$.$$_+"\\"+$$$.__$+$$$.$$_+$
  $$._$$+$$$._$+(![]+"")[$$$._$_]+$$$.$$$_+"."+(![]+"")[$$$._$_]+$$$._$+"\\"+$$$.__$+$$$.$__+$$$.$$$+"
  (\\\"\\"+$$$.__$+$$$.___+$$$.$$_+"\\"+$$$.__$+$$$.__$+$$$.$__+"\\"+$$$.__$+$$$.___+$$$.__$+"\\"+$$$.
  __$+$$$.___+$$$.$$$+"
  {\\"+$$$.__$+$$$.__$+$$$._$_+"\\"+$$$.__$+$$$._$_+$$$._$$+"\\"+$$$.$__+$$$.___+"\\"+$$$.__$+$$$.___+
  $$$.$_$+"\\"+$$$.__$+$$$.$_$+$$$.$$_+$$$.$$__+$$$._$+$$$.$$_$+$$$.$$$_+"\\"+$$$.__$+$$$.$$_+$$$._$_+
  "\\"+$$$.$__+$$$.___+"\\"+$$$.__$+$$$._$_+$$$._$$+$$$._+$$$.$$__+"\\"+$$$.__$+$$$.$_$+$$$._$$+"\\"+$
  $$.__$+$$$.$$_+$$$._$$+"}\\\");"+"\"")())();
  FLAG{JS Encoder Sucks}                                                                    VM77:3
< undefined
```

即可得到flag

multilayer

解题脚本

```
import base64
from Crypto.Util import number
n=0x80dd2dec6684d43bd8f2115c88717386b2053bdb554a12d52840380af48088b7f1f71c3d3840ef4615af318bbe261d2d2d90616c0d2dcb6414e05c706f
e=0xcf98d5
lines = open('encrypted').readlines()
data = base64.b64decode(lines[3].strip())
def xor(a, b):
   res=''
   for i in range(len(a)):
       res+=chr(ord(a[i])^ord(b[i]))
   return res
dec = {}
for i in range(0x10000):
   x = b'%.4x' % i
   v = number.bytes_to_long(x)
   dec[pow(v, e, n)] = x
raw = b''
for i in range(256, len(data), 256):
   prev = data[i-256:i]
   curr = int(xor(prev, data[i:i+256]).encode('hex'), 16)
   raw += dec[curr]
data = raw.decode('hex')
r = number.inverse(17, 251)
for key in range(0,256):
   output=''
   res=''
   for i in data:
       key = (key * 0xc8763 + 9487) % 0x10000000000000000
       output+=chr((ord(i) ^ key) & 0xff)
   for i in output:
       res += chr((ord(i)*r)%251)
   if res[4:5]=='{' and res[-2:] == '}\n':
       print res
       break
```

详细题解:
https://xz.aliyun.com/t/2627

ffa

解题脚本

```
from z3 import *
from primefac import *
import libnum
M=349579051431173103963525574908108980776346966102045838681986112083541754544269
```

```
p=2406701218042089783949967107308390697287009568247069459848190153714938375512 38
```

```
p=240670121804208978394996710730839069728700956824706945984819015371493837551238
q=63385828825643452682833619835670889340533854879683013984056508942989973395315
z=213932962252915797768584248464896200082707350140827098890648372492180142394587
m=282832747915637398142431587525135167098126503327259369230840635687863475396299
x=254732859357467931957861825273244795556693016657393159194417526480484204095858
y=261877836792399836452074575192123520294695871579540257591169122727176542734080

a, b, c = BitVecs('a b c', 262)
s = Solver()
s.add(UGT(a, pow(2, 256, m)))
s.add(ULT(a, pow(2, 257, m)))
s.add(UGT(b, pow(2, 256, m)))
s.add(ULT(b, pow(2, 257, m)))
s.add(UGT(c, pow(2, 256, m)))
s.add(ULT(c, pow(2, 257, m)))
s.add(x == (a + b * 3) % m)
s.add(y == (b - c * 5) % m)
s.add(z == (a + c * 8) % m)
while s.check() == sat:
   A,B= s.model()[a].as_long(),s.model()[b].as_long()
   if gcd(A,B) == 1:
       break
s1,s2,tmp = libnum.xgcd(A, B)
if s1<0:
   s1 = - s1
   p = modinv(p, M)
   if p<0:
       p+=M
elif s2<0:
   s2 = - s2
   q = modinv(q, M)
   if q<0:
       q+=M
m=(pow(p,s1,M)*pow(q,s2,M)) % M
print libnum.n2s(m)
```

详细题解：
https://www.anquanke.com/post/id/156915

后记

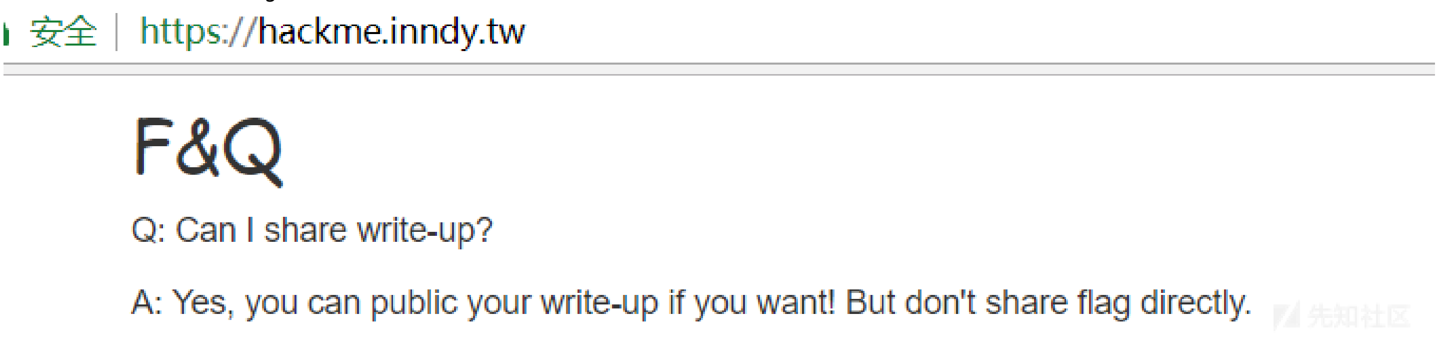
欢迎师傅们讨论，菜鸡献丑了！

点击收藏 | 0 关注 | 1
上一篇：利用Craft CMS SEOma... 下一篇：通过 how2heap 复习堆利用...

1. 3 条回复

findneo 2018-08-15 19:11:02

作者好像不是很高兴share flag ：）



安全 | https://hackme.inndy.tw

F&Q

Q: Can I share write-up?

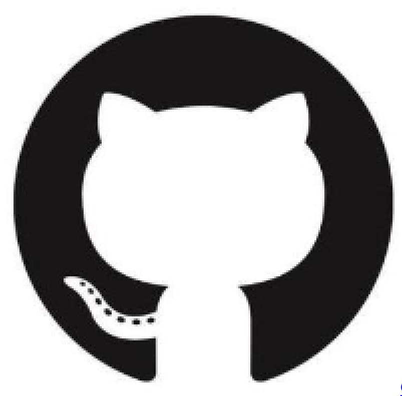
A: Yes, you can public your write-up if you want! But don't share flag directly.

1 回复Ta

一叶飘零 2018-08-15 20:32:52

@findneo 谢谢提醒，已将每题最后的flag删掉。但是我觉得flag给不给都无所谓：）思路和方法才是最重要的

2 回复Ta

chybeta 2018-08-15 21:16:37

感谢分享。

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板