

# HGAME 2022 Week1 writeup by ek1ng

## HGAME 2022 Week1 writeup by ek1ng

### WEB

easy\_auth

蛛蛛...嘿嘿♥我的蛛蛛

Tetris plus

Fujiwara Tofu Shop

### MISC

欢迎欢迎！热烈欢迎！

这个压缩包有点麻烦

好康的流量

群青(其实是幽灵东京)

### CRYPTO

Dancing Line

Easy RSA

Matryoshka

English Novel

### IOT

饭卡的uno

### RE

easyasm

flagchecker

### PWN

test nc

## WEB

### easy\_auth

easy\_auth的意思是简单的身份验证，题目描述admin在todo上记录了重要内容即flag，考察的是网页登录使用JWT鉴权的知识，当Token的signature采用弱校验方式也就是secret为空时出现漏洞。

首先我们点击给出的网站看一下，如果直接访问登陆后页面会因为没有token跳转到login.html，题目在login.html中提供了登录和注册功能，那么第一步的话我们先注册一个账户登陆进去看一下，提示success后我们登录注册的账号。

不安全 | http://adminisdoingwhat.mjclouds.com/login.html

应用 CyberChef CTF在线工具-CTF...

用户名/手机号/邮箱	密码	login
admin1211	admin1211	注册

登录成功后由于这是我们注册的账户，里面什么都没有很正常，然后我们在浏览器的调试工具devtools里查看cookie，发现网页使用JWT的方式进行登录鉴权，那么JWT的鉴权上应该是存在漏洞的，我们需要将Token的形式伪装成admin的身份，通过校验。

The screenshot shows the Network tab of the DevTools interface. A cookie named 'token' is listed, with its value being a long JWT token. The token is highlighted with a yellow background in the screenshot.

Token是分为三部分的，分别是JWT一共由三部分组成，分别为头部(header),载荷(payload)，签证(signature)。三部分用.进行分割，header中是声明类型和加密的算法，payload中是标准中注册的声明和公共、私有声明，签证部分是将base64编码后的header与base64编码后的payload通过符号相连，然后通过header中声明的加密方式进行secret组合加密。

我们在jwt.io这样的jwt解密网站上可以查看Token信息

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJRCI6MSwiVXNlck5hbWUiOiJhZG1pbisIlBob25lIjoiIiwiRW1haWwiOiIiLCJleHAiOjE2NDI5NjU5NDEsImlzcyI6Ik1KY2xvdWRzIn0.mpQl6FCoXtA9RbwGXqAV7IRbmKpGSIALyUYAcUd1Fc8
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM &amp; TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYLOAD: DATA

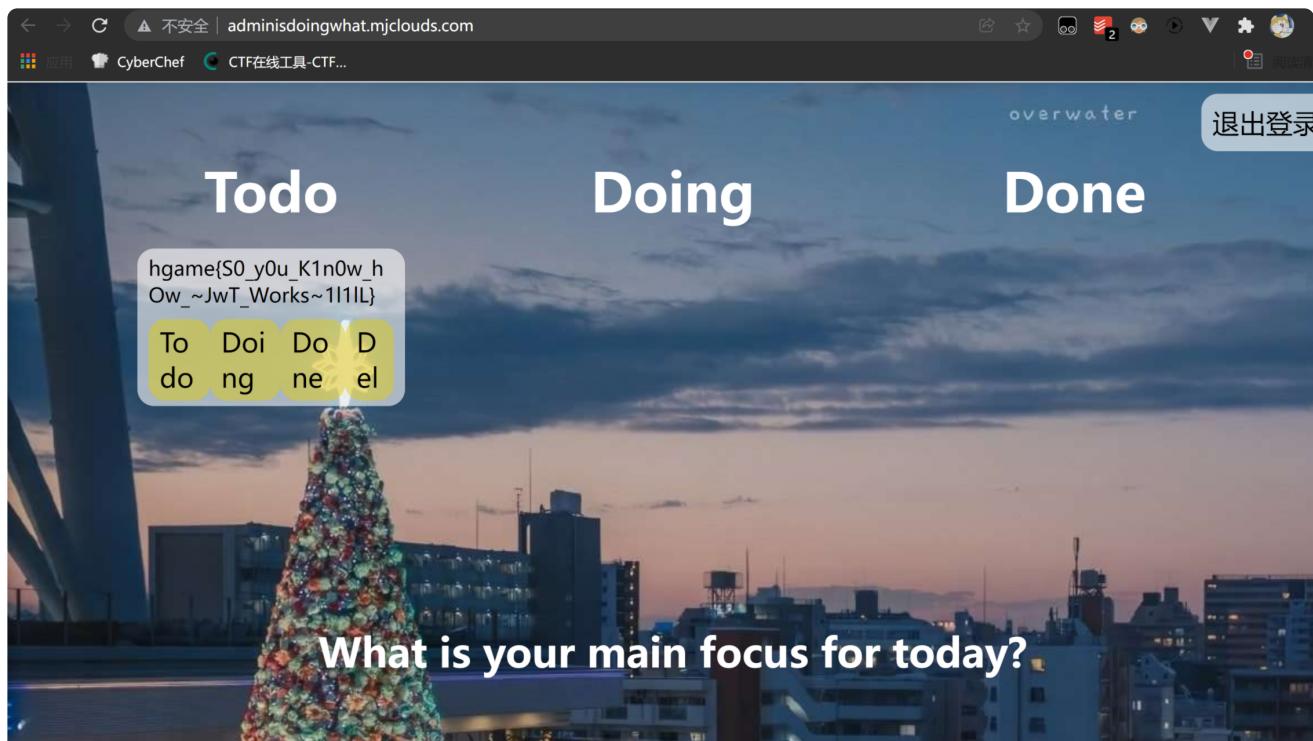
```
{  
  "ID": 1,  
  "UserName": "admin",  
  "Phone": "",  
  "Email": "",  
  "exp": 1642965941,  
  "iss": "MJclouds"  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
    
 )  secret base64 encoded
```

经过多次尝试后发现将token的ID改为1，UserName的改为admin，将浏览器中暂存的token更改成我们修改后的token我们就可以以admin的形式成功登录，拿到flag啦。

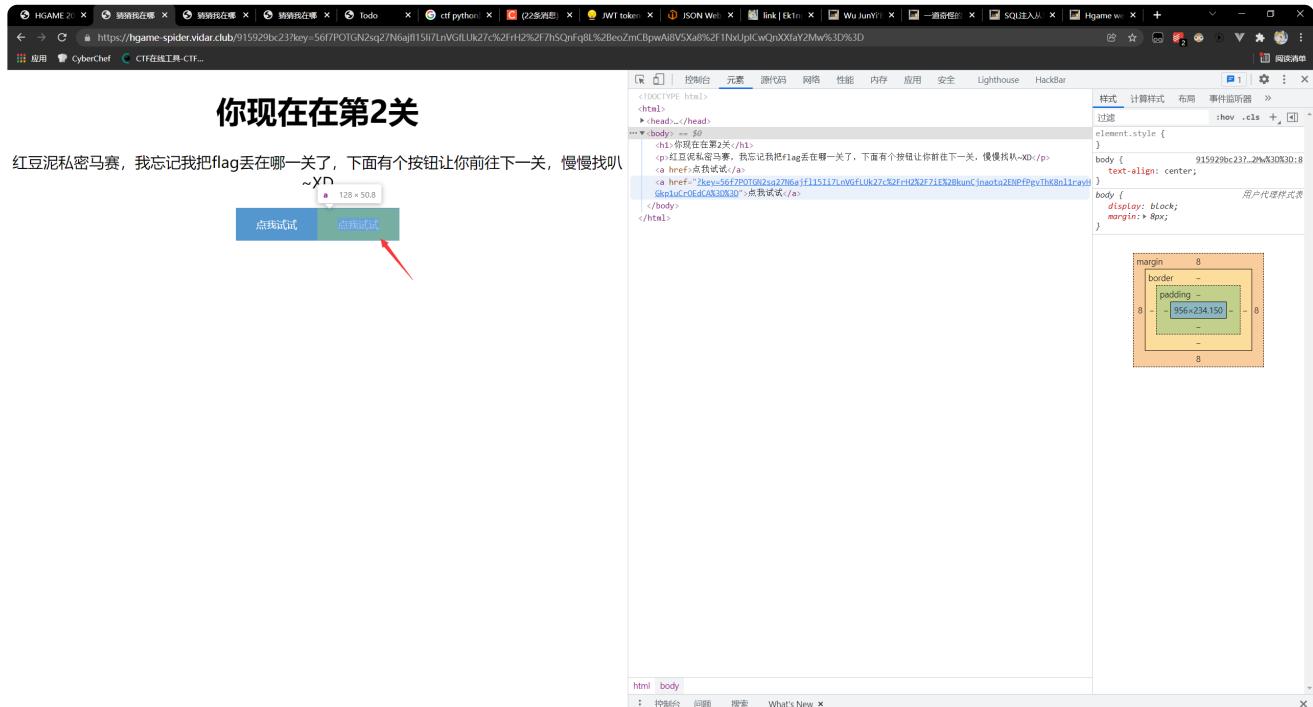
此外需要注意的是，我也尝试了爆破Token，但是注册Token的有效期网站提示只有3分钟，而爆破jwt token 需要好几个小时才能跑完。



## 蜘蛛...嘿咻♥我的蜘蛛

题目名字和题目描述在没开始做题前的话我们可以通过我的蜘蛛正在满地找头这个题目描述猜测会考察HTTP请求头响应头的知识，题目需要使用python脚本进行自动化点击（或者手动点100个）以及HTTP响应头的知识。

我们点击进入给出的网站后出题人提示flag不知道被丢在哪一关了，我们尝试点击按钮后发现，第n关跳转第n+1关有n个按钮只一个按钮是正确的，通过f12我们可以看到哪个按钮会是正确的，但是我也想知道flag会在哪一关呀然后就这么尝试点了100关我还真的点到了...



然后提示flag就在当前页面时，查看http请求头的内容就能发现flag啦，写题解的时候觉得靠手点始终不是个办法，毕竟这次是因为才100关我点了3分钟就看到了但是如果设置的多一些的话就会比较绝望的所以说正儿八经的解法应该是用python脚本去点的，然后我就写了个python脚本，重新做了一遍这题，python代码如下

```
from time import sleep
from selenium import webdriver

# 打开浏览器
wd = webdriver.Chrome(r'chromedriver.exe')
wd.get('https://hgame-spider.vidar.club/915929bc23')
# 让程序停1s保证打开页面后再自动点击
sleep(1)

# 自动点击
for i in range(100):
    # 点击href值不为空的按钮
    for link in wd.find_elements_by_xpath("//*[@href]"):
        try:
            if link.get_attribute('href') != '':
                wd.get(link.get_attribute('href'))
        except:
```

pass

```
# 打开浏览器
wd = webdriver.Chrome(r'chromedriver.exe')
wd.get('https://hgame-spider.vidar.club/915929bc23')
# 让程序停1s保证打开页面后再自动点击
sleep(1)

# 自动点击
for i in range(100):
    # 点击href值不为空的按钮
    for link in wd.find_elements_by_xpath("//*[@href]"):
        try:
            if link.get_attribute('href') != '':
                wd.get(link.get_attribute('href'))
        except:
            pass

for i in range(100)
```

运行: main > C:\Users\76104\Desktop\HGAME2022\week1\web\venv\Scripts\python.exe C:/Users/76104/Desktop/HGAME2022/week1/web/autoClick/main.py  
C:\Users\76104\Desktop\HGAME2022\week1\web\autoClick\main.py:4: DeprecationWarning: executable\_path has been deprecated, please pass in  
wd = webdriver.Chrome(r'chromedriver.exe')  
C:\Users\76104\Desktop\HGAME2022\week1\web\autoClick\main.py:9: DeprecationWarning: find\_elements\_by\_\* commands are deprecated. Please u  
for link in wd.find\_elements\_by\_xpath("//\*[@href]"):  
进程已结束,退出代码0

运行完python脚本后我们发现从100关直接跳转到了这个页面，看起来flag就被落（藏）在了这里



找到flag落下的地方后，按钮点击也是不会有什么事情发生，f12查看一下前端源码也是啥也没有。

The screenshot shows a browser window with the URL <http://hgame-spider.vidar.club/915929bc23?key=56f7POTGN2sq27N6ajfl15li7LnVGfLUk27c%2FrH2%2F7goEhL7cpzDNha7asgT%2F%2FDVW6Bzs...>. The page content is:

# 我好像在就是把flag落在这里了欸~ 快帮我找找x

红豆泥私密马赛，我忘记我把flag丢在哪一关了，下面有个按钮让你前往下一关，慢慢找叭~XD

[点我试试](#)

The browser's developer tools are open, showing the source code and element inspector. The source code pane shows:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body> == $0
    <h1>我好像在就是把flag落在这里了欸~ 快帮我找找x</h1>
    <p>红豆泥私密马赛，我忘记我把flag丢在哪一关了，下面有个按钮让你前往下一关，慢慢找叭~XD</p>
    <a href="#">点我试试</a>
  </body>
</html>
```

The element inspector shows the body element with the following styles:

```
body {
  text-align: center;
}
body {
  display: block;
  margin: 8px;
}
```

A bounding box is drawn around the body element, highlighting its dimensions: 629.200x214.637.

结合题目我的蛛蛛正在满地找头，在HTTP响应头中我们发现了自定义的fi4g这个响应头，就找到flag啦

The screenshot shows a browser window with the same URL as the previous screenshot. The page content is identical.

The browser's developer tools are open, showing the network tab. A request for [点我试试](#) is selected, showing the following response headers:

名称	标头	载荷	预览	响应	启动器	时间
915929bc23?key=56f7POTGN...	n					18
favicon.ico						

**响应标头**

```
auth0r: asjdf
content-length: 831
content-type: text/html; charset=utf-8
date: Thu, 27 Jan 2022 07:41:29 GMT
fi4g: hgame(d76358d92407d56f48a7dd72e34497de5d78
8abc46b267e2a88b3f974f564b63)
welcome-to-hgame: See you next week!
x-api-apid: 1308188104
x-api-funcname: helloworld-1642513741
x-api-httphost: nil
x-api-id: api-6p0hmfp8t
x-api-requestid: bf7c8d0f30b8bc31f96b9bce7d1f79f
3
x-api-serviceid: service-kjbkqayp
```

hggame {d76358d92407d56f48a7dd72e34497de5d788abc46b267e2a88b3f974f564b63}

# Tetris plus

Tetris plus意思是加强版俄罗斯方块，点击页面发现是个小游戏，题目描述告诉我们需要达到3000分，解题需要用到JS审计、Base64、JSfuck编码的知识。

题目提示了3000分就会给flag，然后简单玩了玩游戏感觉挺有意思的，F12查看源代码后，在checking.js文件中找到了校验分数的部分。

HIGH SCORE  
79

# Hextris

Play!

FACEBOOK    TWITTER

```
66 // remove block from array
67 // hex blocks as deleted
68 hex.blocks[arr[0]][arr[1]].deleted = 1;
69 deletedBlocks.push(hex.blocks[arr[0]][arr[1]]);
70 }

71 }

72 // add scores
73 var now = hex.lastCombos < settings.comboTime ?
74 if (now - hex.lastCombo < settings.comboTime) {
75 settings.comboTime = (1 / settings.creationSpeed*modifier) * (waveone.nextGen / 16.666667) * 3;
76 hex.lastCombo += 1;
77 hex.lastCombo = now;
78 var coords = findCenterOfBlocks(deletedBlocks);
79 hex.texts.push(new Text(coords['x'],coords['y'],"x" + hex.comboMultiplier.toString(),"bold 0","#fff",fadeUp));
80 } else {
81 settings.comboTime = 240;
82 hex.lastCombo = now;
83 hex.comboMultiplier = 1;
84 }

85 var adder = deleting.length * hex.comboMultiplier;
86 hex.texts.push(new Text(hex.x,hex.y,"+" + adder.toString(),"bold 0 ",deletedBlocks[0].color,fadeUpAndOut));
87 hex.lastBlock = deletedBlocks[0].color;
88 score += adder;
89 }

90 if (score >= 2000 && !window.winned) {
91 winned = true;
92 alert("2nd0h2y0osozkLzoopvo14!abfmaXkuobvvIz1ho3mib7eb7kkch")
93 }
94 
```

发现两个地方，首先是未被注释的部分，当score变量的值大于3000时，页面会弹窗将ZmxhZyDosozkvLzooqvol4/otbfmnaXkuobvvIzlho3mib7mib7lkKch进行Base64解码后的内容给玩家，那么我们直接Base64解码看看，发现这是假的flag。（万恶的出题人即便你玩到了3000分也不会给你flag hhhh）

< 返回

AmanCTF - BASE64编码解码

在线BASE64编码解码

ZmxhZyDosozkvLzooqv0l4/otbfmnaXkuobvvIzlho3mib7mib7lkKch

解密

flag 貌似被藏起来了，再找找吧！

那么另一个可疑的地方就是下面注释的部分了，经过搜索我们发现这是JSfuck编码，JSfuck编码只需要在控制台中运行就可以解码了，放在控制台中跑一下然后我们就得到了真的flag。

The screenshot shows the DevTools Network tab with a single large request to 'game.summ3r.top/Tetris/index.html'. The response body is filled with a massive amount of encoded data, appearing as a continuous string of characters. A blue highlight box surrounds the beginning of this data, containing the text: 'alert("hgame(jsfuck\_1s\_S0\_fUu1n)")'.

## Fujiwara Tofu Shop

题目意思是想成为车神需要做一些事情吧大概hhh用的是头文字D的背景，考察的是HTTP请求头、响应头的知识，以及能够使用Postman或者Burpsuite这类工具。

一开始我还没看懂什么叫去一趟qiumingshan.net，我还以为是需要访问qiumingshan.net什么的，然后发现这个qiumingshan.net的DNS解析不到对应主机总之就是非常迷惑然后问了问出题人发现原来想复杂了，访问的其实就是题目给出的shop.summ3r.top，那么根据描述去一趟qiumingshan.net，我们给HTTP请求头添加Referer，Referer的作用是在客户端向web服务器发送请求时，告诉服务器该网页是从哪个页面链接过来的，服务器因此可以对访客信息做一些处理统计什么的工作。

Referer: qiumingshan.net

Burp Suite Professional v2020.8 - Temporary Project - licensed to surferxyz

Burp 项目 测试器 重发器 窗口 帮助

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × 2 × 3 × 4 × ...

发送 取消 < | > | ? 目标: http://shop.summ3r.top

**请求**

Raw Headers Hex

```
1 GET / HTTP/1.1
2 Host: shop.summ3r.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/84.0.4147.105 Safari/537.36
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;
v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Referer: qiumingshan.net
11
12
```

**响应**

Raw Headers Hex Render



只有借助AE86才能拿到车神通行证 (Hachi-Roku)

? ⚙️ ← → Search... 没有匹配 In Pretty 完成 557字节 | 17毫秒

网页又提示需要借助AE86才能成为车神，所以将User-Agent的内容改为Hachi-Roku

User-Agent的作用是向访问网站提供你所使用的浏览器类型、操作系统及版本、CPU类型、浏览器渲染引擎、浏览器语言、浏览器插件等信息的标识，我们原先发送请求时默认会带上的，更改为Hachi-Roku后我们就得到了进一步提示。

User-Agent: Hachi-Roku

Burp Suite Professional v2020.8 - Temporary Project - licensed to surferxyz

Burp 项目 测试器 重发器 窗口 帮助

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × 2 × 3 × 4 × ...

发送 取消 < | > | ?

目标: http://shop.summ3r.top

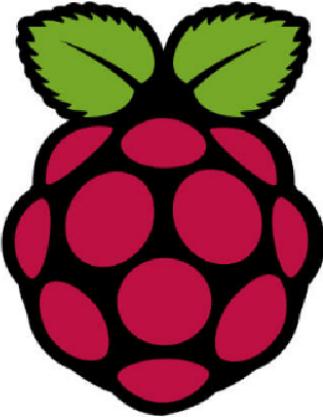
**请求**

Raw Headers Hex

```
1 GET / HTTP/1.1
2 Host: shop.summ3r.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Hachi-Roku
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Referer: qiumingshan.net
11
12
```

**响应**

Raw Headers Hex Render



86的副驾上应该放一盒树莓（Raspberry）味的曲奇

题目提示放一盒Raspberry味的曲奇，HTTP请求会使用Cookie做持久化访问，然后我们看了看响应头发现响应头中Cookie有个叫flavor的字段,初始值是Strawberry，恰好符合提示，我们用请求头将其设置为Raspberry就可以了

Cookie: flavor=Raspberry

Burp Suite Professional v2020.8 - Temporary Project - licensed to surferxyz

Burp 项目 测试器 重发器 窗口 帮助

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × 2 × 3 × 4 × ...

发送 取消 < | > | ?

请求

Raw Headers Hex

```
1 GET / HTTP/1.1
2 Host: shop.summ3r.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Hachi-Roku
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Cookie: flavor=Raspberry
11 Referer: qiumingshan.net
12
13
```

响应

Raw Headers Hex Render

名称	值
HTTP/1.1	200 OK
Content-Type	text/html; charset=utf-8
Gasoline	0
Server	gin-gonic/gin v1.7.7
Set-Cookie	flavor=Strawberry; Path=/; Domain=localhost; Max-Age=3600
Date	Sun, 23 Jan 2022 09:43:12 GMT
Content-Length	309
Connection	关闭

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>□□□□</title>
6 </head>
7 <body>
8   
9   <p>86□□□□□□□□□□Raspberry□□□□</p>
10 </body>
11 </html>
```

没有匹配

Search... Pretty

没有匹配

Search... Pretty

更改后作者再次提示我们需要将汽油加到100， 我们可以在响应头中发现Gasoline:0, Gasoline应该是自定义在响应头中的，非常奇怪，我们翻译一下发现这个单词刚好是汽油的意思，所以说将这个值修改成100就可以了。

Gasoline: 100

不过我很奇怪的遇到在burpsuite给http请求头因为我中间发过一些什么奇奇怪怪的请求，添加Gasoline: 100无法得到响应，我还以为是不能这么加，但是搜了老半天也是没发现问题

Burp Suite Professional v2020.8 - Temporary Project - licensed to surferxyz

Burp 项目 测试器 重发器 窗口 帮助

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

4 × 5 × 6 × ...

发送 取消 < | > | ? 目标: http://shop.summ3r.top

请求 响应

Raw Params Headers Hex Raw Headers Hex Render

```
1 GET / HTTP/1.1
2 Host: shop.summ3r.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Hachi-Roku
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Cookie: flavor=Raspberry
11 Referer: qiumingshan.net
12 Gasoline: 100
13
```

① ⚙️ ⏪ ⏩ Search... 没有匹配 ⌂ Pretty

待机中

② ⚙️ ⏪ ⏩ Search... 没有匹配 ⌂ Pretty

所以说怀着疑惑的心情又在postman中尝试了，神奇的发现可以

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', 'Flows', and 'History'. The main area has tabs for 'New Collection' and 'New Request'. A 'GET' request is selected with the URL 'http://shop.summ3r.top/'. The 'Headers' tab is active, showing the following configuration:

Key	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.0
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Referer	qiumingshan.net
User-Agent	Hachi-Roku
Cookie	flavor=Raspberry
Gasoline	100

The 'Body' tab shows the response content as:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <title>暮勇且商店</title>
7 </head>
8
9 <body>
10   
11   <p>哪怕成了车停，也请让请求从本地发出来才最重要到 Ding !</p>
12 </body>
13
14 </html>
```

然后再回到burpsuite中尝试了一下，大概是我之前发在加Gasoline: 100之前还添加了什么burpsuite有缓存啥的？？？总之重新试了一下发现可以了

Burp Suite Professional v2020.8 - Temporary Project - licensed to surferxyz

Burp 项目 测试器 重发器 窗口 帮助

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

4 × 5 × 6 × ...

发送 取消 < | > | ?

目标: http://shop.summ3r.top

请求

Raw Params Headers Hex

```
1 GET / HTTP/1.1
2 Host: shop.summ3r.top
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Hachi-Roku
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Connection: close
10 Cookie: flavor=Raspberry
11 Referer: qiumingshan.net
12 Gasoline: 100
13
14
```

响应

Raw Headers Hex Render



哪怕成了车神，也得让请求从本地发出来才能拿到 flag！

①⚙️ ⏪ ⏩ Search... 没有匹配 In Pretty 完成 570字节 | 75毫秒

这里作者提示要从本地访问页面，我们先尝试XFF，XFF是识别客户端最原始的IP地址的，它通过三个Proxy把请求转发给服务端，改成X-Forwarded-For: 127.0.0.1发现不行并且被出题人嘲讽了，然后做到这里我就卡住了，在问了出题人后也是得知本地访问的话其实有好几种请求头的方式，然后出题人把XFF禁用了所以在网上搜了搜发现了X-Real-IP也可以起同样的效果

X-Forwarded-For: 127.0.0.1

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

New Import Overview GET https://api.hduhel... New Collection [CONFLICT] GET New... + \*\*\*

No Environment

Save Send Cookies

My Workspace Collections + New Collection GET New Request

Params Authorization Headers (12) Body Pre-request Script Tests Settings

User-Agent: PostmanRuntime/7.29.0

Accept: \*/\*

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Referer: qiumingshan.net

User-Agent: Hachi-Roku

Cookie: flavor=Raspberry

Gasoline: 100

X-Forwarded-For: 127.0.0.1

Key Value Description

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

1 大黑侠也想当车手？

Status: 200 OK Time: 728 ms Size: 345 B Save Response

所以我们换了一种方式，添加X-Real-IP请求头，发现拿到了flag

X-Real-IP:127.0.0.1

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

New Import Overview GET https://api.hduhel... New Collection + \*\*\*

No Environment

Save Send Cookies

My Workspace Collections + New Collection GET New Request

Params Authorization Headers (14) Body Pre-request Script Tests Settings

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Referer: qiumingshan.net

User-Agent: Hachi-Roku

Cookie: flavor=Raspberry

Gasoline: 100

X-Forwarded-For: 127.0.0.1

client: 127.0.0.1

X-Real-IP: 127.0.0.1

Key Value Description

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize

hgane[[1\_beought\_A\_S3xy\_sw1nSu1t]]

Status: 200 OK Time: 23 ms Size: 272 B Save Response

在写题解的时候在写XFF要经过三个proxy转发的时候想到出题人禁用XFF这个问题，所以说去尝试了在后面的proxy里面写127.0.0.1，发现出题人的这种禁用只禁用了第一个127.0.0.1这个proxy，那么这种方法也是可以神奇拿到flag的，然后问了问出题人发现这貌似还是预期之外的（

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area is titled 'New Collection / New Request' with a 'GET' method and the URL 'http://shop.summ3r.top'. Under the 'Headers' tab, there are several entries: 'User-Agent' (checked) with value 'Hachi-Roku', 'Cookie' (checked) with value 'flavor=Raspberry', 'Gasoline' (checked) with value '100', 'X-Forwarded-For' (checked) with value '127.0.0.1,127.0.0.1,127.0.0.1', and 'client' (unchecked) with value '127.0.0.1'. Below the headers, the 'Body' tab is selected, showing the response body: 'hgamer{I\_b0ught\_4\_S3xy\_sw1mSuit}'. At the bottom right, there are status indicators: 200 OK, 1581 ms, 349 B, and a 'Save Response' button.

## MISC

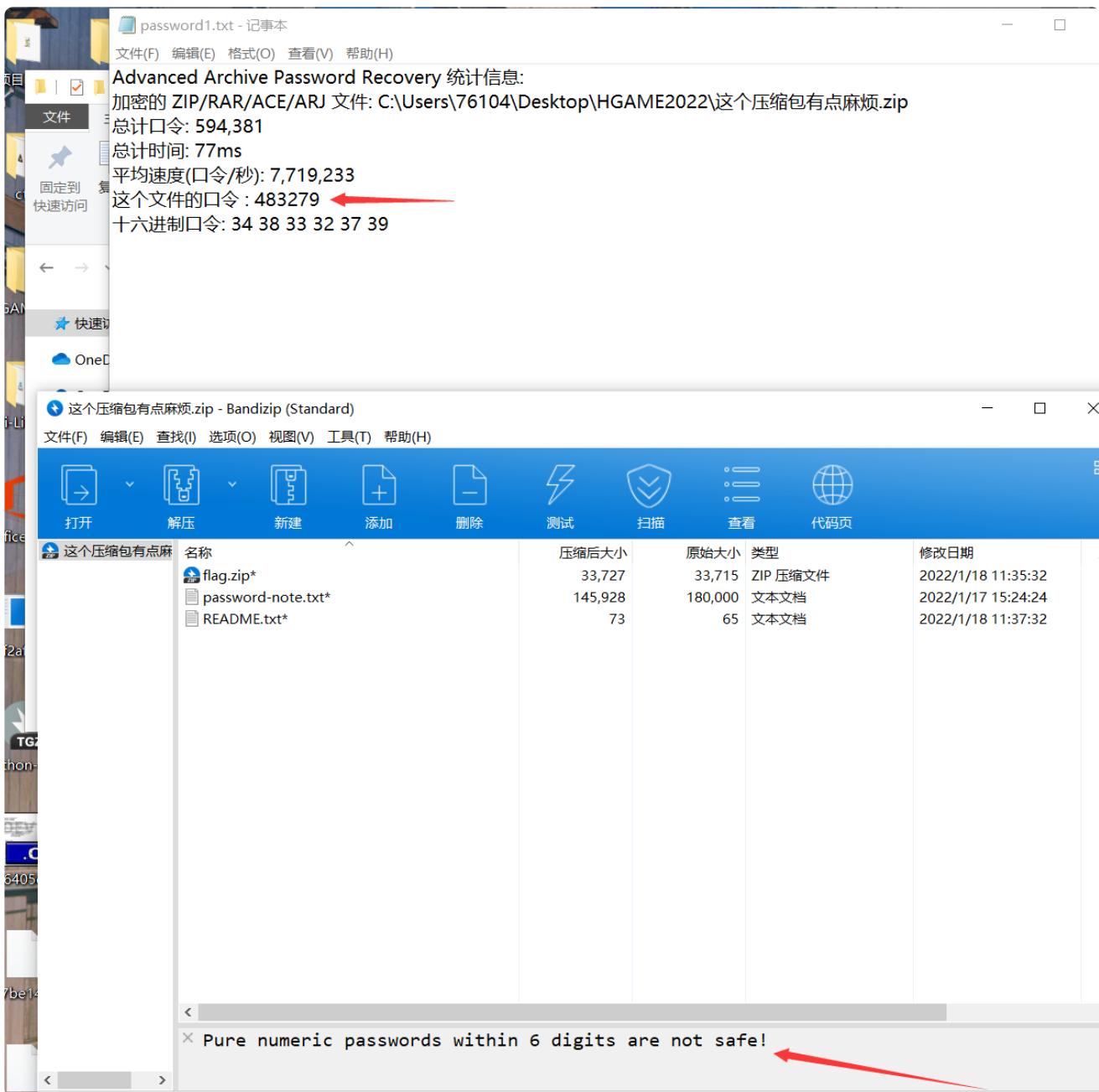
欢迎欢迎！热烈欢迎！

快乐签到( 连签到都没有别人手快

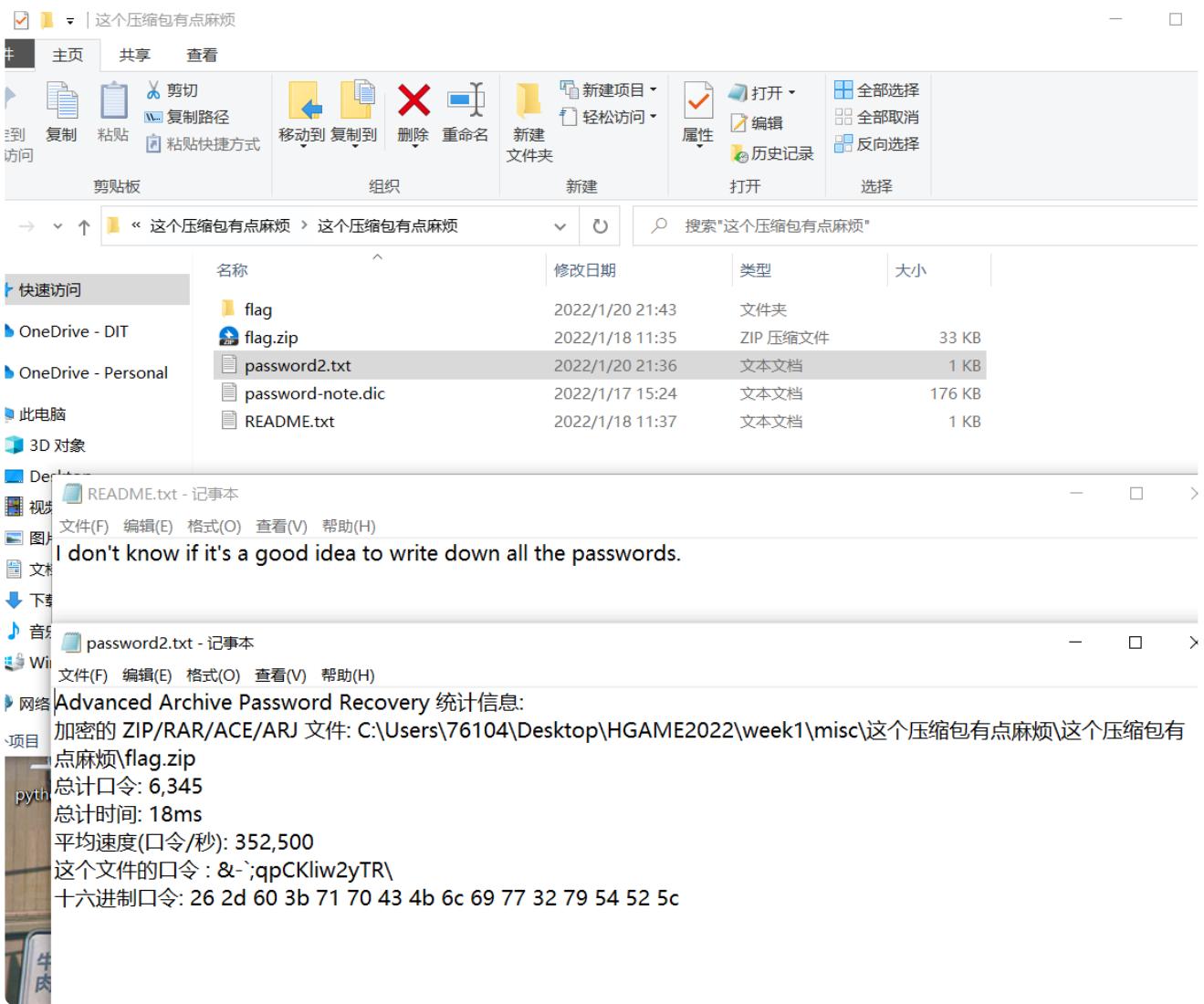
## 这个压缩包有点麻烦

主要是使用ARCHPR这个工具，压缩包总共有6位数字爆破 + 字典 + 明文 + binwalk提取结束位后的隐藏zip + 伪加密这么5层加密，需要注意的是压缩包这个题还更新过附件，我觉得应该是旧压缩包在明文这一层有点问题来着所以一直0人解然后我也是卡在明文这个地方，后面的话就看到更新附件想抢一二三血结果被binwalk这一层卡住了然后好像第二天才搞定hhh

第一层的加密提示用bandzip打开压缩包就能看到，使用ARCHPR成功解密



字典加密的话从README.txt里面可以看出，题目给的这个password-not.dic就是密码字典，使用密码字典也是成功解密



第三层的明文加密，一开始这个给出的README.txt是68B但是压缩包里面的却有73B，不符合明文加密的条件所以也是解不出来，更新附件后就成功解密了，以及README中告诉我们压缩包采用了仅存储的压缩方式，因为明文加密是将README.txt这个压缩包内也有的文件采用相同的方式压缩后，由于有相同的文件，两个压缩包的二进制值会有一部分相等，利用这一点来进行爆破的。

共享 查看

剪切 复制路径 粘贴 快捷方式 移动到 复制到 删除 重命名 新建项目 轻松访问 属性 全部选择 全部取消 反向选择

剪贴板 组织 新建 打开 选择

↑ < 这个压缩包有点麻烦 > flag 搜索"flag"

名称 修改日期 类型 大小

flag.zip 2022/1/18 11:20 ZIP 压缩文件 33 KB  
README.txt 2022/1/18 11:35 文本文档 1 KB

ag.zip - Bandizip (Standard)

编辑(E) 查找(I) 选项(O) 视图(V) 工具(T) 帮助(H)

开 解压 新建 添加 删除 测试 扫描 查看 代码页

ag.zip 名称 压缩后大小 原始大小 类型 修改日期

flag.jpg\* 32,993 32,981 JPG 文件 2022/1/18 11:05:24

README.txt\* 85 73 文本文档 2022/1/18 11:19:34

README.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

If you don't like to spend time compressing files, just stores them.

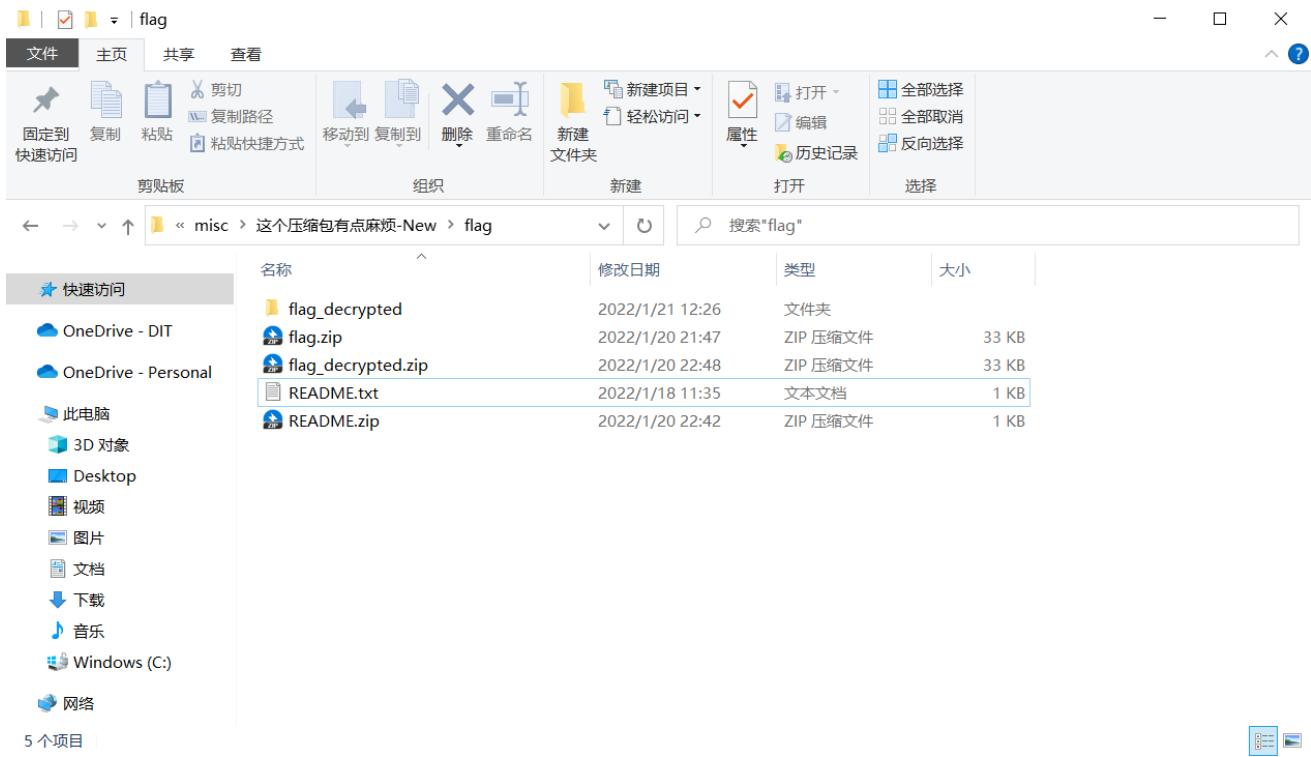
README.txt 属性

常规 安全 详细信息 以前的版本

README.txt

文件类型: 文本文档 (.txt)

打开方式: 记事本 更改(C)...  
位置: C:\Users\76104\Desktop\HNGAME2022\week1\misc  
大小: 68 字节 (68 字节)  
占用空间: 0 字节



解密后我们得到一张flag.jpg，使用winhex查看发现在jpg的结束位后隐藏了信息，我们使用binwalk工具将隐藏的压缩包提取出来，发现仍然是个压缩包，使用winhex打开后，找到504B0102后面的这一串，从50开始数第9，10位是0900，改成0000后保存一下，我们就拿到了flag.jpg



## 位置管理器 (全部)

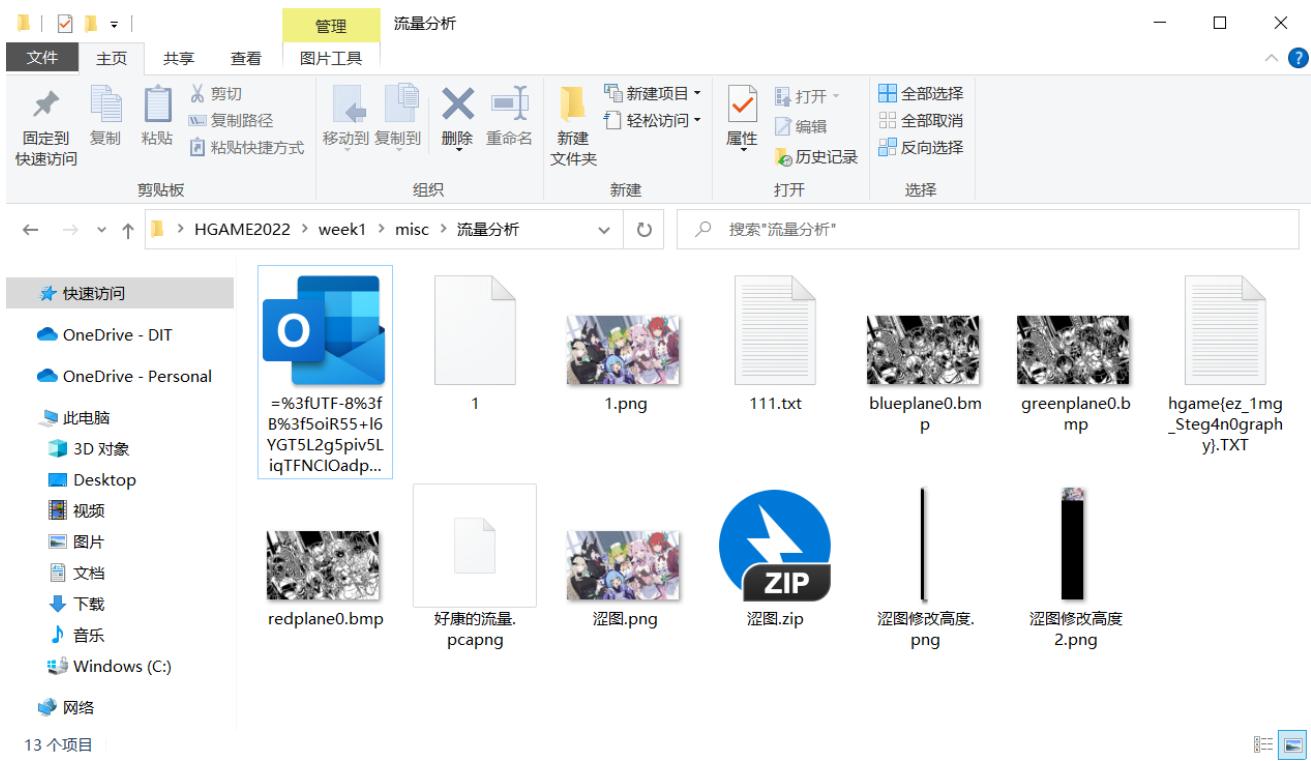
Offset ▲	搜索结果	时间
0 504B		2022/01/21 1...
12448 504B		2022/01/21 1...
12538 504B		2022/01/21 1...

Offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	ANSI ASCII
00012256	1A B8 2D 36 12 02 6A E0 51 40 3E 71 79 E4 6E 3A	, -6 jàQ@>qyän:
00012272	51 FE 24 D4 8D B3 9B 85 BD A2 4B 39 C4 4B E8 E0	Qþ\$Ó „ „ „ K9ÄKèà
00012288	64 45 2E D2 E9 81 42 5F 3B 65 8C C4 E1 C9 7D F7	dE. Òé B_ ;eGÄáÉ}÷
00012304	C4 DB 38 D9 13 22 A8 74 79 F2 20 AB 4E CD 6E 95	ÄÜ8Ù "“tyò «Nín·
00012320	89 EB B9 15 61 53 97 2E D8 82 CC 97 E5 52 CC CE	ñë` aS-. Ø, ï-åRíí
00012336	B6 CD 19 75 6E 2D CB E2 AE 1C 69 4F 7E 32 D4 25	qí un-Éåç iO~2ô%
00012352	AE 55 D8 9C 56 EB 10 F5 CC D4 18 30 83 AA 77 CC	€UØæVë öÌÖ Ofawí
00012368	A0 6C 4A F6 BE C3 E9 A1 09 4B F6 83 69 7A AF A1	1Jö%Ãé; Köfiz-
00012384	83 E5 D7 5D C8 86 19 AF EF 3F 88 98 62 C3 78 AA	få×]Èt “i?~^bÅx^
00012400	03 F3 90 A0 AA 4A 9F 7B 43 5A 62 4C A4 C0 EA E2	ó „ JÝ{CZbLñÀèå
00012416	69 D0 87 3C E8 D2 78 AF 81 A7 BC 24 4E F5 40 DB	iÐ‡<èòx~ \$4\$Nõ@Ù
00012432	9B 74 27 36 47 60 30 65 99 A8 FF A5 96 54 FF 00	>t'6G`0e”“ý¥-Tý
00012448	50 4B 01 02 3F 00 14 00 09 00 08 00 7D 57 32 54	PK ? }W2T
00012464	5C A6 20 F3 7A 30 00 00 C3 33 00 00 08 00 24 00	\; óz0 Å3 \$
00012480	00 00 00 00 00 20 00 00 00 00 00 00 00 66 6C	f1
00012496	61 67 2E 6A 70 67 0A 00 20 00 00 00 00 00 01 00	ag.jpg
00012512	18 00 9F 19 E2 79 17 0C D8 01 79 40 A3 ED 03 0E	Ý áy Ø y@£i
00012528	D8 01 00 55 30 44 1E 0C D8 01 50 4B 05 06 00 00	Ø UOD Ø PK
00012544	00 00 01 00 01 00 5A 00 00 00 A0 30 00 00 00 00	Z 0

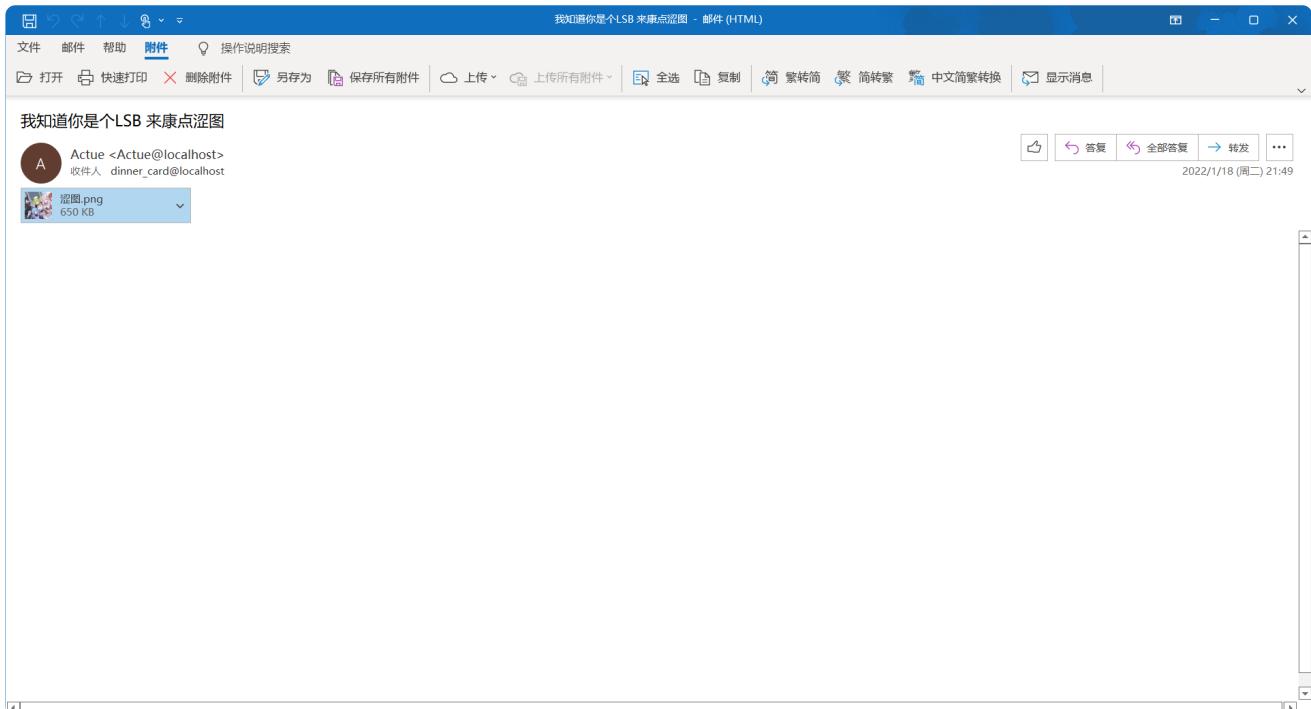
hgame{W0w!\_y0U\_Kn0w\_z1p\_3ncrYpt!}

## 好康的流量

这题考察了Wireshark流量包分析和使用StegSlove分析LSB隐写图片、文本，这题我是真的尝试了很多种后来才发现是LSB隐写文本拿到了另外半个flag，搜索引擎发现的各种方法都试了一遍



首先我们打开Wireshark，左上角导出然后另存为，我们发现这是个.eml，然后发现系统自带的邮件和outlook可以打开但是windows自带的邮件我以前也没添加过账户然后我添加账户还添加不上去，然后换了outlook打开了这个邮件，那么看到两个提示，第一个是LSB，提示使用了LSB隐写，第二个就是这个图片了我们把它保存下来

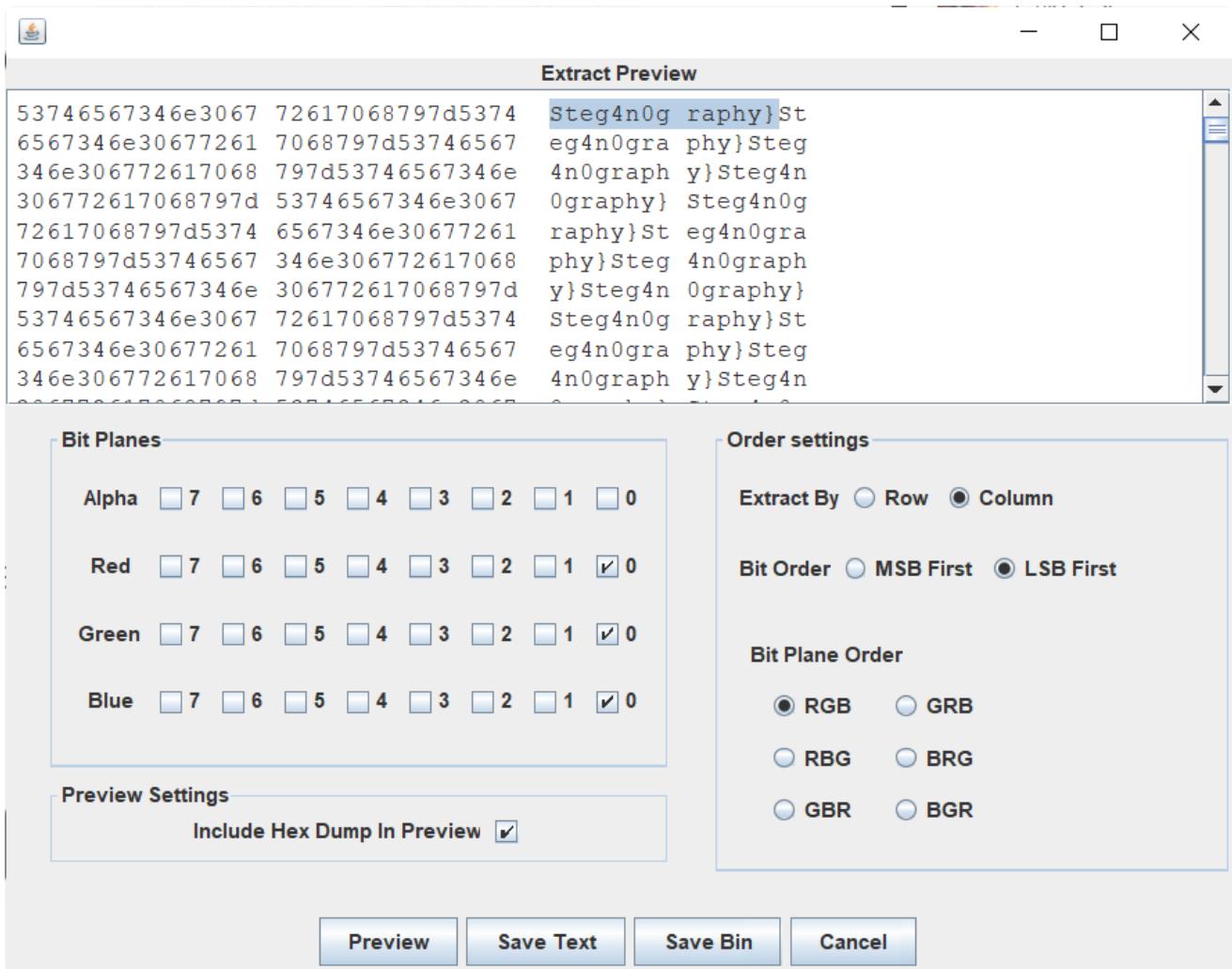


用StegSlove打开后在绿色通道2上我们发现这里隐写了一个条形码，打开支付宝扫码发现这里有半个flag

hgame{ez\_1mg\_



然后另外半个flag需要使用stegslove打开图片后，在Extract By选择按列计算才能够拿到这半个，一开始选的是按行然后通道换来换去也是没找到隐写的信息。

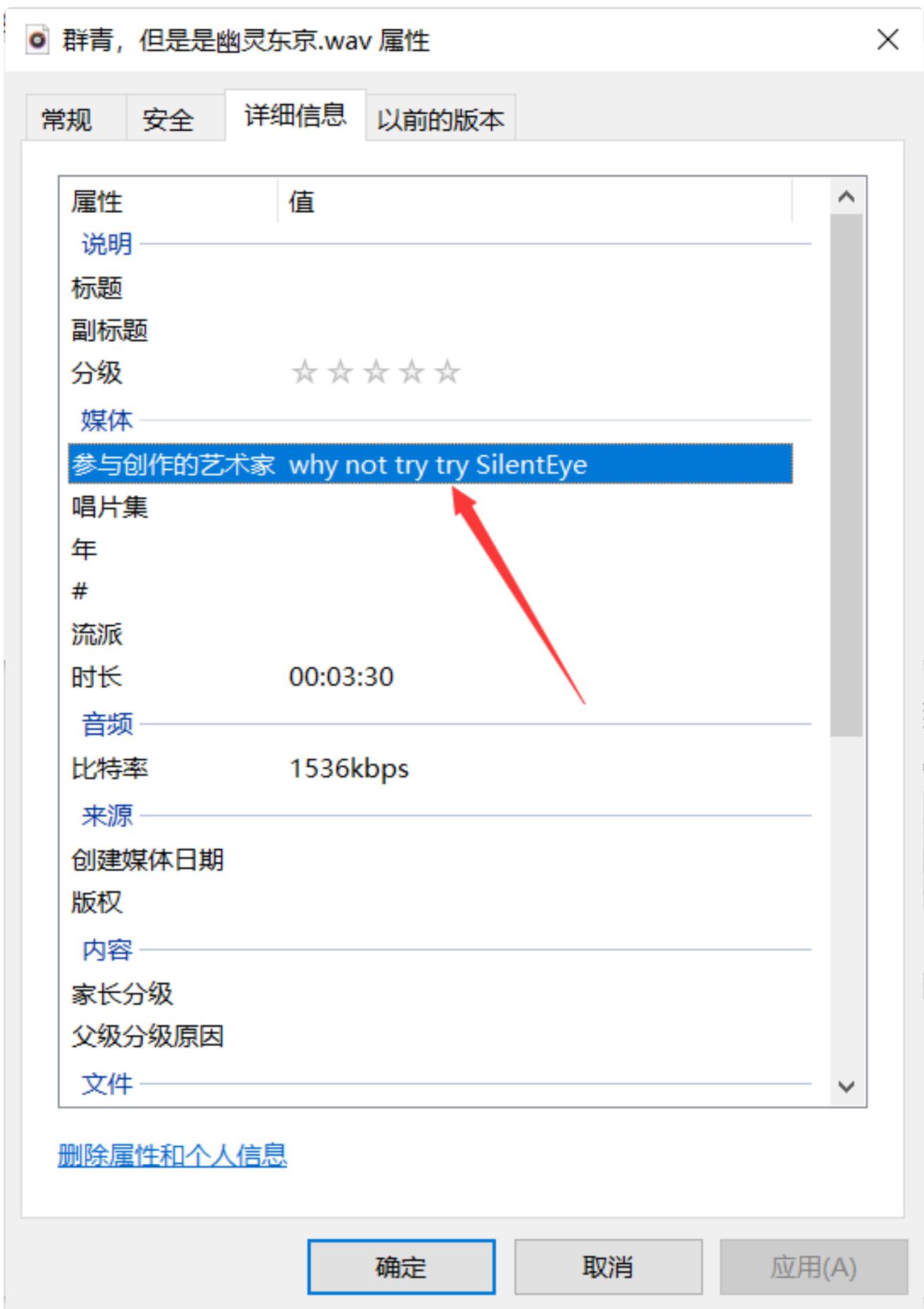


拼凑起来我们就拿到了flag啦

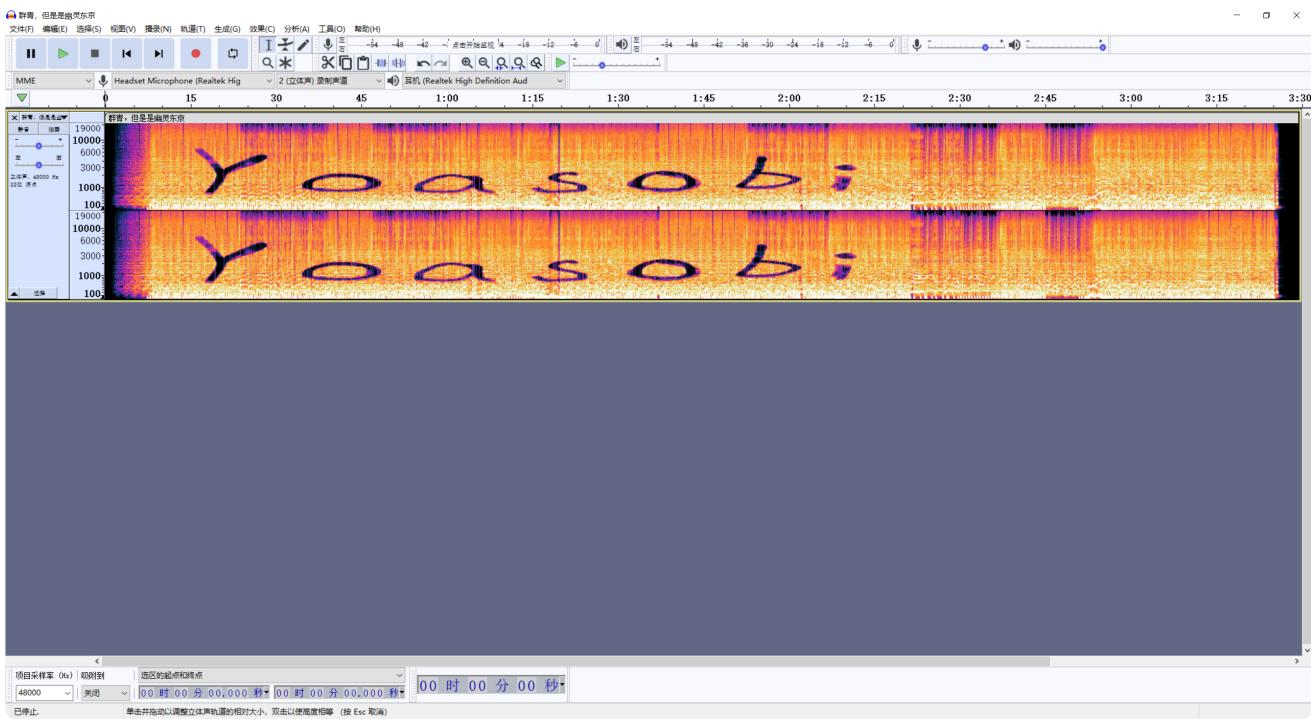
```
hgame{ez_1mg_Steg4n0graphy}
```

## 群青(其实是幽灵东京)

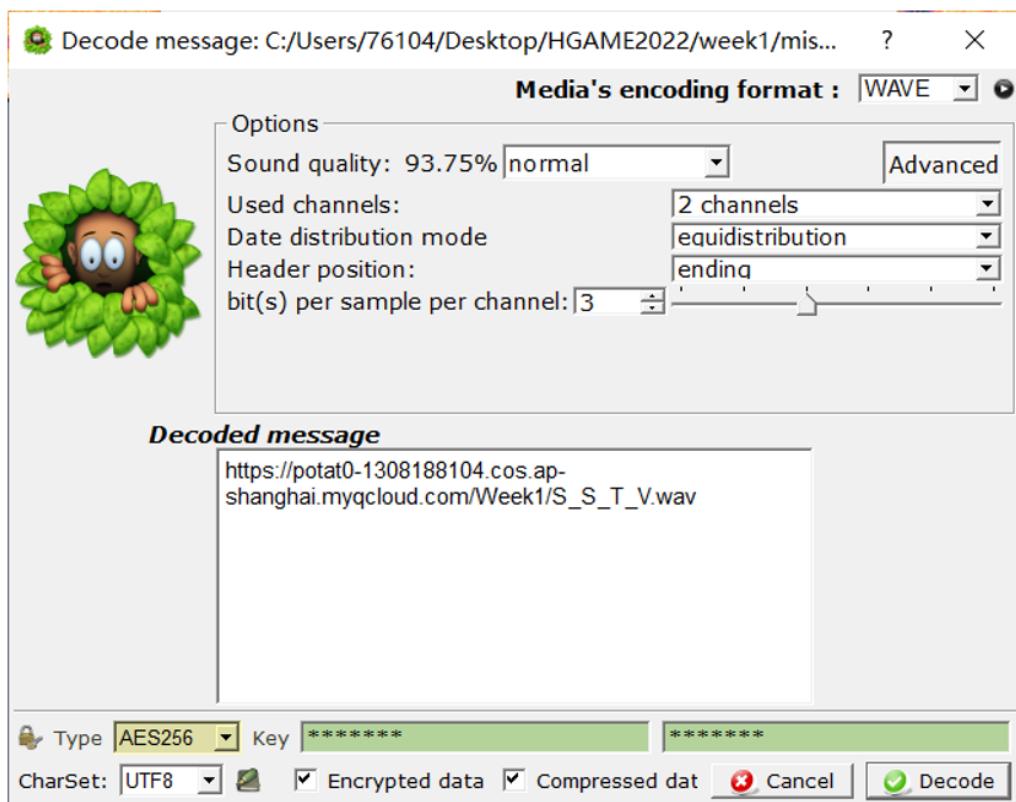
这是考察音频隐写的一个题。



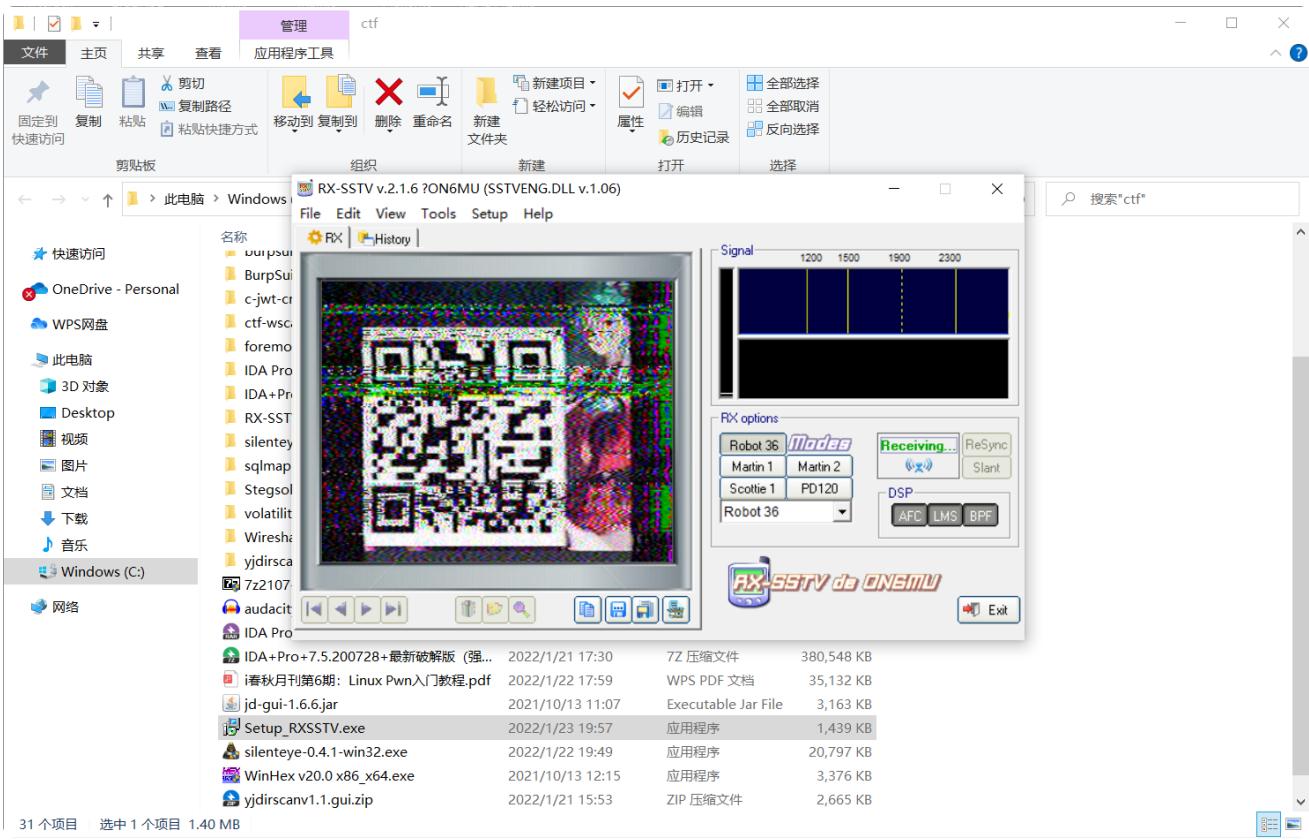
搜索后发现SilentEye是个隐写的工具，下载后尝试解密但是啥也没得到，于是用audacity打开，查看频谱图发现隐写了文本Yoasobi



在SilentEye里将Yoasobi填为密码，发现可以成功解密



下载新的音频后，搜索SSTV发现慢扫描电视（Slow-scan television 简称SSTV）是业余无线电爱好者的一种主要图片传输方法那么可以判断这里隐藏了图片，我们使用RX-SSTV这类的解密SSTV的软件，发现了隐写的二维码，扫码就能够得到flag啦

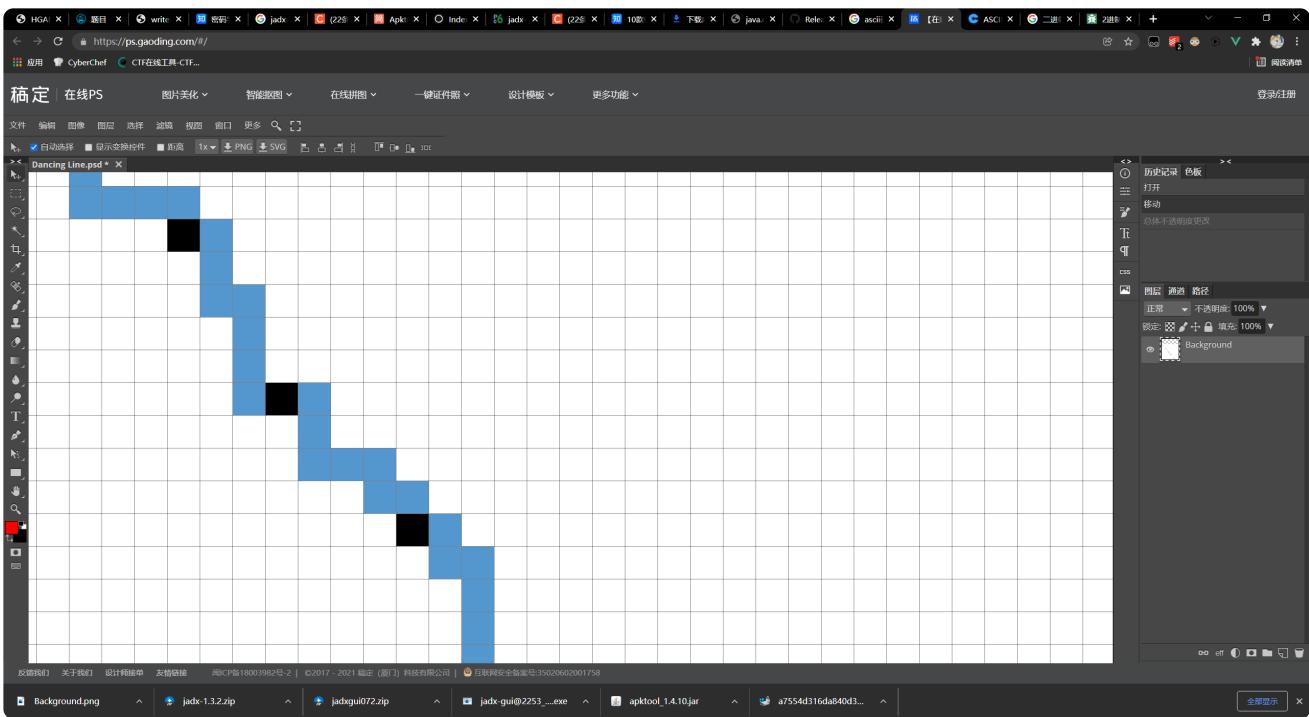


## CRYPTO

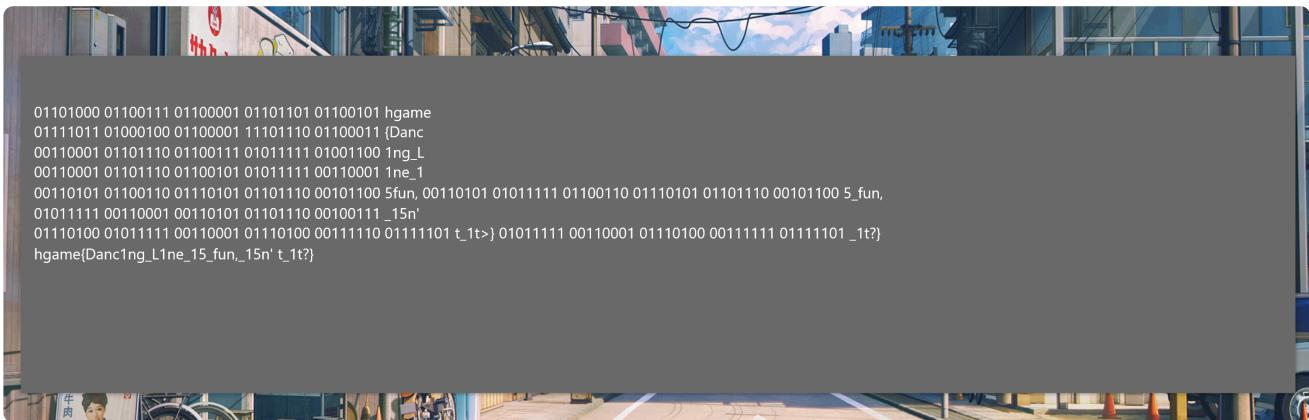
### Dancing Line

题目和描述给出的信息不多，打开发现就是以Dancing Line这款游戏为背景的一个图片，用一条线的横和竖表示了0，1，那么黑点有什么用呢，我数了数发现有37个好像（，然后一个flag的长度也差不多，那我们就可以猜测这个解出来就是flag，每两个黑点间的一段就是表示flag的一个字符，但是如何清楚的看清图片走向呢，我找了个在线ps网站，打开图片发现这个图片恰好是每步一个像素点

<https://ps.gaoding.com/#/>



那这样的话我们只需要数一数或者想办法用python脚本做一些图像识别的工作再转换成flag就可以了，由于这里我想了想点不算多，20分钟可以搞定，于是我才用了手动数的方法，下面是手动数记录的过程hhh，我数了前5个然后转换了一下0和1的对应关系，刚好能对应上hgame，横是0竖是1，然后就每次数5个，把这个flag数出来了



写题解的时候又思考了一下，正确的解法应该是用Pillow这个Python的图像处理库，来做这个信息处理的工作，可惜写题解写到这里的时候已经18.26了马上8点要week2了，所以没来得及把这个题写个脚本处理一下试试。

## Easy RSA

这是一道比较简单的RSA算法的题目

首先查资料了解什么是rsa：

1. 随意选择两个大的质数p和q, p不等于q, 计算N = pq.
2. 根据欧拉函数, 求得r=φ(N)=φ(p)φ(q)=(p-1)(q-1)。

3. 选择一个小于r的整数e, 是e与r互质。并求得e关于r的模反元素，命名为d。 (求d令 $ed \equiv 1 \pmod{r}$ )。 (模反元素存在，当且仅当e与r互质)

4. 将p和q的记录销毁。

其中(N, e)是公钥，(N, d)是私钥。

然后我们看到题目的加密算法：将flag每个字符通过rsa加密并给出了pqe和c，那么我们要做到就很简单了，求出d然后解出m。

```
import gmpy2
def make_key(p, q, e):
    n = p * q
    phi = (p-1) * (q-1)
    d = gmpy2.invert(e, phi)
    return d
m = [(12433, 149, 197, 104), (8147, 131, 167, 6633), (10687, 211, 197, 35594),
      (19681, 131, 211, 15710), (33577, 251, 211, 38798), (30241, 157, 251, 35973),
      (293, 211, 157, 31548), (26459, 179, 149, 4778), (27479, 149, 223, 32728),
      (9029, 223, 137, 20696), (4649, 149, 151, 13418), (11783, 223, 251, 14239),
      (13537, 179, 137, 11702), (3835, 167, 139, 20051), (30983, 149, 227, 23928),
      (17581, 157, 131, 5855), (35381, 223, 179, 37774), (2357, 151, 223, 1849),
      (22649, 211, 229, 7348), (1151, 179, 223, 17982), (8431, 251, 163, 30226),
      (38501, 193, 211, 30559), (14549, 211, 151, 21143), (24781, 239, 241, 45604),
      (8051, 179, 131, 7994), (863, 181, 131, 11493), (1117, 239, 157, 12579), (7561,
      149, 199, 8960), (19813, 239, 229, 53463), (4943, 131, 157, 14606), (29077, 191,
      181, 33446), (18583, 211, 163, 31800), (30643, 173, 191, 27293), (11617, 223,
      251, 13448), (19051, 191, 151, 21676), (18367, 179, 157, 14139), (18861, 149,
      191, 5139), (9581, 211, 193, 25595)]
print(len(m))
for i in range(38):
    e, p, q, c = m[i]
    d = make_key(p, q, e)
    n = p * q
    plaintext = (gmpy2.powmod(c, d, n))
    print(chr(plaintext), end='')
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "web" and contains files like "autoClick", "main.py", and "decode.py".
- Code Editor:** The "decode.py" file is open, displaying the provided Python script.
- Run Tab:** The "decode" configuration is selected, and the command "C:/Users/76104/Desktop/HGAME2022/week1/web/virtualenv/Scripts/python.exe C:/Users/76104/Desktop/HGAME2022/week1/crypto/rsa/decode.py" is shown.
- Output Tab:** The output shows the command being run and the resulting output: "38", "hgame looks like you've mastered RSA!", and "进程已经结束,退出代码0".

## Matryoshka

题目就是已知了凯撒加密的key和维吉尼亚加密的key，意思就是密文flag被层层加密后变成了盲文，现在需要解密回去，解密顺序是这样的

Braille 盲文 -> Reverse 逆序密码 -> Morse 摩斯 -> hex 转16进制 -> Vigenère 维吉尼亞 -> base64 -> 棚栏:22 -> Caesar 凯撒:21 -> Flag

The screenshot shows the CyberChef interface with the following setup:

- Operations:** rever (highlighted in blue)
- Recipe:** From Braille (highlighted in green)
- Input:** A large block of Braille text (length: 1099 lines: 1).
- Output:** The resulting Morse code (length: 1099 lines: 1).
- Buttons:** STEP, BAKE!, Auto Bake.

cyberchef不知道为什么解不出这层莫斯，然后换了个在线解密网站解出来了，如果不加这层逆序的话莫斯之后后面是解着解着就发现有问题的，也是根据hint 2个置换2个代换4个编码，已知有维吉尼亞和凯撒两个代换密码了，所以说的话如果莫斯直接解不出那么就看看能不能加一层代换密码，加了逆序后再莫斯就能够顺利解出来了

解出莫斯密码后因为字符中最大都没有超过F，猜测是16进制编码去cyberchef里面转HEX

The screenshot shows the CyberChef interface with the following configuration:

- Operations:** base64
- Recipe:** From Hex → Vigenère Decode
- Input:** Hexadecimal string: 46,66,42,75,56,45,46,6E,6D,4C,73,36,44,33,73,69,59,74,4C,36,58,32,70,34,69,4E,30,63,64,53,6C,79,6B,6D,39,72,51,4E,39,6F,4D,53,31,6A,6B,73,39,72,4B,32,5D,136,6B,4C,38,68,6F,72,30,3D
- Vigenère Decode:** Key: hgame
- Output:** Decrypted string: YzBibXZnaH16X3swUmF6X2d4eG0wdGhrem9fMG9iMG1fdm9rY2N6dF8hcn0=

再解一层维吉尼亚后 这个显然是base64的形式所以说再解一层base64

然后栅栏的话我在cyberchef好像找不到，可能是我搜的不吧于另外找了个网站尝试了

栅栏密码是一种简单的移动字符位置的加密方法，规则简单，容易破解。栅栏密码的加密方式：把文本按照一定的字数分成多个组，取每组第一个字连起来得到密文1，再取每组第二个字连起来得到密文2.....最后把密文1、密文2.....连成整段密文。例如：

明文：栅栏密码加密规则示例

每组字数：5

按照字数先把明文分成：

栅栏密码加

密规则示例

先取每组第一个字：栅密

-----

一个一个试过去，因为已经知道最后一层是凯撒然后栅栏密码的key也不会超过这个原文长度，那么key从1开始解码之后去凯撒21再解一层看看是不是flag的形式就好，最后也是试出key=22时候能解密出flag

From Braille x | HGAME 20... x | 凯撒密码加密 x | 楼梯密码在线 x | 摩斯密码在线 x | 摩尔斯电码 x | (22条消息) x | +

← → C 不安全 | http://www.atoolbox.net/Tool.php?Id=778

应用 CyberChef CTF在线工具-CTF...

一个工具箱 - 好用的在线工具都在这里! 登陆 搜索工具... Go

**推荐热门**

- 工具榜单
- 专享工具
- 视频音频
- 图形图像
- 日期时间
- 文字编辑
- 加密解密**
- 编程开发
- 计算换算
- 金融理财
- 生活日常
- 扫码访问
- 问题反馈

## 凯撒密码加密/解密

明文: hgame{Welc0me\_t0\_the\_w0rld\_0f\_crypt0graphy!}

偏移量: 21

加密 解密

密文: cbvhz{Rzgx0hz\_o0\_ocz\_r0mgy\_0a\_xmtko0bmvkct!}

一个小工具箱，拥有便捷实用小工具的微信小程序，无须安装，用完即走！工具小程序，一个就够了！

微信搜索：一个小工具箱

后来发现原来cyberchef是解的出morsecode的 是我分隔符没有选对 选forward slash就可以了

From Braille x | HGAME 20... x | 楼梯密码在线 x | 楼梯密码在线 x | 摩斯密码在线 x | 摩尔斯电码 x | (22条消息) x | +

← → C https://icyberchef.com/#recipe=From\_Braille()Reverse('Character')From\_Morse\_Code('Forward%20slash','L...' | ☆ | 阅读清单

Download CyberChef [Download](#)

Last build: 2 years ago Options [About / Support](#)

Operations	Recipe	Input
morse	From Braille	length: 1099 lines: 1
From Morse Code	Reverse	
To Morse Code	By Character	
Favourites	From Morse Code	
Data format	Letter delimiter Forward slash	
Encryption / Encoding	Word delimiter Line feed	
Public Key		
Arithmetic / Logic		
Networking		
Language		
Utils		
Date / Time		
Extractors		
Compression		

Output

```
46,66,42,75,66,45,46,6E,6D,4C,73,36,44,33,73,69,59,74,4C,36,58,32,70,34,69,4E,30,63,64,53,6C,79,6B,6D,39,72,51,4E,39,6F,4D,53,31,6A,6B,73,39,72,4B,32,52,36,6B,4C,38,68,6F,72,30,3D
```

STEP BAKE! Auto Bake

## English Novel

题目首先给出了410个分块加密的txt，其中每个txt是410字节，是能够一一对应的，那么整体的思路就是先根据加密算法写一个python的脚本找到加密的key，然后用key和加密后的flag.enc找到加密前的flag.enc

一开始没有注意到原文和密文并不是根据序号直接对应的所以说尝试解了几个后发现每个解出来的key都不一样

```
21: # 19141819173127702511272425162512505171617232041801919181513231110141417412142051451723219121021162819198021313419932182161217139181182132517952561018191820171267451815014112022562131322012711552417121  
22: # 925282523251215205131812596191917212025951420152229114121651601317515112022141332291919212014124212380021762525151204141521151921322138822296915122080126141418145247217825010519325147  
23: # 22319542523021872122251237619106164684170253962514921014281788916182010614206241913102512098071111514161804215256612251716369619181018491800111610106925241951807652561924171132421141211221157231604112208
```

首先是寻找对应part部分 findPart.py

```
import os

filePath=r"" # 文件夹路径
fileList=os.listdir(filePath)

for file in fileList:
    f=open(os.path.join(filePath,file))
    while True:
        line = f.readline()
        if not line:
            break
        line = line.strip('\n')
        key = 'ktl'
        if key in line:
            print(file)# txt文件内容
            print(line)
```

根据findPart.py 检索关键字

由于flag的开头一定是hgame 那么的话有了flag.enc我们就能知道key[0]-key[4]的值

比如part0 最前面5个字符是 read 加密后就会是 wmme 然后我们用wmme去加密后的part里面检索，就能够找到part0对应了part175

解密部分 encrypt.py

```
encryptData = "" # 未加密数据
originalData = "" # 加密后数据
key = [0 for i in range(410)]
encryptFlag = 'klsyf{W0_j0v_ca0z_\'Ks0ao-bln1qstxp_juqfqy\'?}'
def encrypt(originalData, encryptData,):
    for i in range(len(originalData)):
        key[i] = ord(encryptData[i]) - ord(originalData[i])
def enflag(key, encryptFlag):
    result = ""
```

```

for i in range(len(encryptFlag)):
    if encryptFlag[i].isupper():
        result += chr((ord(encryptFlag[i]) - ord('A') - key[i]) % 26 + ord('A'))
    elif encryptFlag[i].islower():
        result += chr((ord(encryptFlag[i]) - ord('a') - key[i]) % 26 + ord('a'))
    else:
        result += encryptFlag[i]
return result
encrypt(originalData, encryptData)
print(enflag(key, encryptFlag))

```

解密之后由于我们key只能在大小写字符处得到，而如果是空格引号等我们的key是默认为0的，所以说只用一组数据我们只能解一部分除非你找的这一组数据前面相当于flag长度的这么一串都是大小写字符但是这个肯定不太可能的，所以说我再随便找了2组数据。

part21对应326, part38对应par280:

根据3个对应关系找到的flag 然后稍微拼凑一下就得到flag啦

```

# flag.enc klsyf{W0_j0v_ca0z_Ks0ao-bln1qstxp_juqfqy'?
# part 0 -> part 175 kgame{W0_y0u_kn0w_Kn0wn-bla1nttxt_attack'?
# part 21 -> part 326 hgame{W0_y0v_kn0w_Kn0wn-pla1ntext_autaqk'?
# part 38 -> part 280 kgame{D0_j0u_ka0w_Kn0wo-pln1nttxt_attacy'?
# hgame{D0_y0u_kn0w_Kn0wn-pla1ntext_attack'?

```

```

# flag.enc klsyf{W0_j0v_ca0z_Ks0ao-bln1qstxp_juqfqy'?
# part 0 -> part 175 kgame{W0_y0u_kn0w_Kn0wn-bla1nttxt_attack'?
# part 21 -> part 326 hgame{W0_y0v_kn0w_Kn0wn-pla1ntext_autaqk'?
# part 38 -> part 280 kgame{D0_j0u_ka0w_Kn0wo-pln1nttxt_attacy'?
# hgame{D0_y0u_kn0w_Kn0wn-pla1ntext_attack'?

# 191410191131277025127242516291250517161723204180191015132311101414174121420514517232151210211620151980213134198321821612171391811021325179526101819170201712674510150141120256213132201271155241
# 72520252235111520515811247713812598019197121205951428152221141216251601317511282221413322919192128014212130022176251515122414152115192215221388229691512280312641418145247217825010519325
# 22119542520218721223123761916146841702539025149210142817889161820106142862419131025120980711151416188421525661225171656961918101849180811161018697252419528619724171132421141211221157231604112
# flag.enc klsyf{W0_j0v_ca0z_Ks0ao-bln1qstxp_juqfqy'?
# part 0 -> part kgame{W0_y0u_kn0w_Kn0wn-bla1nttxt_attack'?
# hgame{W0_y0v_kn0w_Kn0wn-pla1ntext_autaqk'?
# hgame{D0_y0u_kn0w_Kn0wn-bla1nttxt_attack'?

```

IOT

## 饭卡的uno

这个题的话我是不太懂iot的，然后看到下载下来是个HEX文件，hex是16进制嘛所以说先解码成字符看一看

### HEX-字符互转

本工具主要目的是实现hex与字符之间的转换。目前支持utf-8/unicode及gbk(兼容gb2312)编码。“字符编码”为“自动”时，将自动识别hex内容并使用正确的编码处理及优化。如果不能识别或是转hex那么将使用默认utf8编码处理。“字符编码”：“hex”用于格式化源hex数据（专业治强迫症），此时“转hex”和“转字符”结果是一样的。

The screenshot shows a tool for hex-to-ascii conversion. The input hex dump includes several recognizable strings like 'ame{First', 'Step\_0A...F\_IO', and 'T)...~...\$.'. The output area shows the corresponding ASCII characters. A red arrow highlights the character 'F' in the input hex dump, which corresponds to the '{' character in the output. A green box at the bottom right indicates '未能识别的数据' and '当前编码: [Hex + Ascii]'.

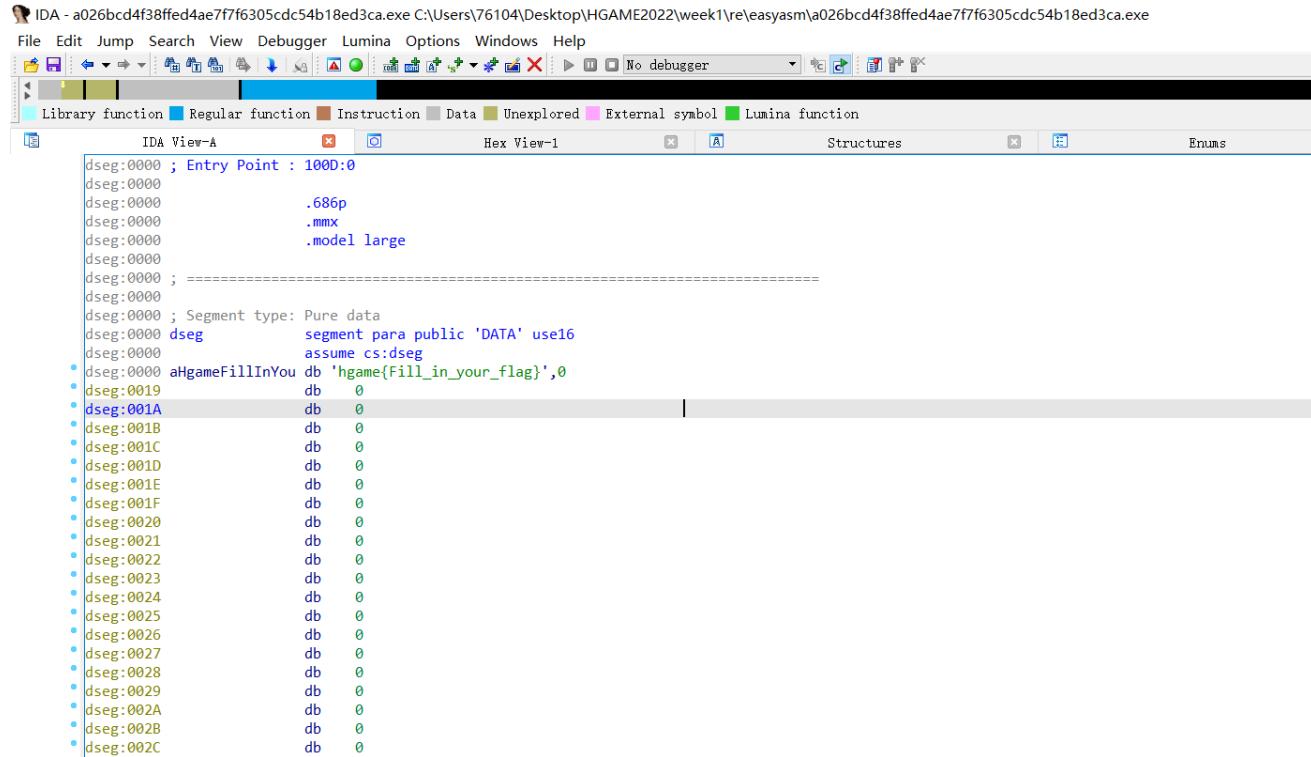
然后我们好像就得到了flag，虽然说格式有点问题，但是flag一般是把字符改成数字一些，还是可以猜出原意，把一些乱七八糟的去掉后也是成功拿到flag，然后后来因为web misc crypto都ak了去看能不能做几个我一点也不会的pwn和re，接触到了ida，使用ida反汇编后，我们就直接发现flag其实被记录在这里。

```
• seg000:000005B0 db 0FFh
• seg000:000005B1 align 2
• seg000:000005B2 dw 5Fh
• seg000:000005B4 db 8Ch
• seg000:000005B5 align 2
• seg000:000005B6 dw 14Ch
• seg000:000005B8 db 0BDh
• seg000:000005B9 align 2
• seg000:000005BA dw 98h
• seg000:000005BC db 0AFh
• seg000:000005BD align 2
• seg000:000005BE db 68h ; h
• seg000:000005BF aGameF1rst5tep0 db 'game{F1rst_5tep_0F_IOT}',0
• seg000:000005D7 db 0Dh
• seg000:000005D8 db 0Ah
• seg000:000005D9 align 2
• seg000:000005DA dw ?
• seg000:000005DC dd 1E09h dup(?)
• seg000:00007E00 dd 0B7842411h, 0FF81BE14h, 0E085D0F0h, 819380h, 9380E082h
seg000:00007E00 dd 0E18800C0h, 0C19380h, 9380E086h, 0E18000C2h, 0C49380h
• seg000:00007E00 dd 0D0C9E08Eh, 0E0869A25h, 0EF3CE320h, 9330E091h, 93200085h
seg000:00007E00 dd 0BB960084h, 0CFE9BB0h, 95A89A1Dh, 0F7A95081h, 24DD24CCh
seg000:00007E00 dd 94832488h, 2EABE0B5h, 2E9AE1A1h, 2EBFE0F3h, 3481D0A2h
seg000:00007E00 dd 0D09FF461h, 0D0AF2F08h, 0F0113802h, 0F4113801h, 0C001E084h
seg000:00007E00 dd 0D08DF083h, 3482C089h, 0F184F411h, 3485C003h, 0F085F419h
```

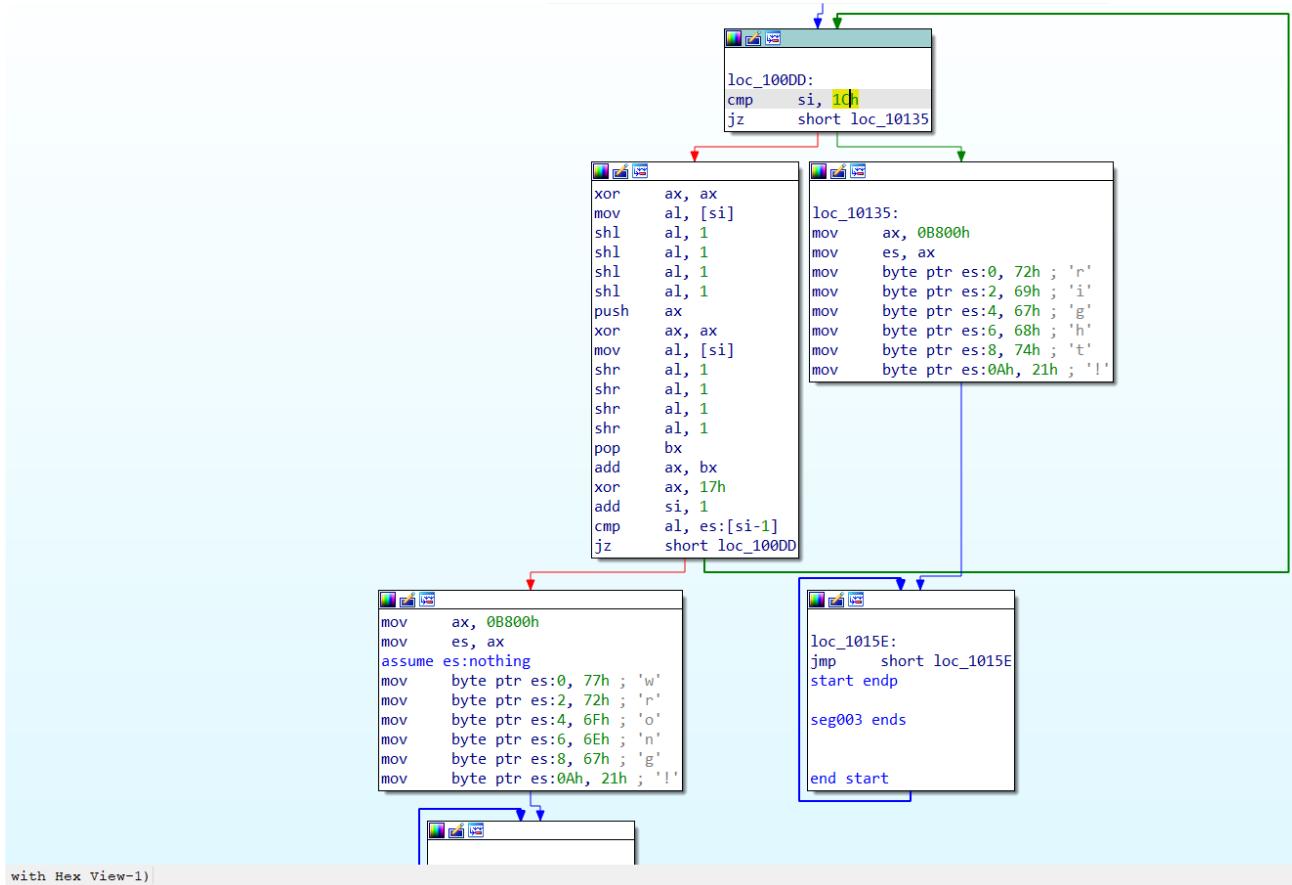
## easyasm

可执行文件是打不开的，是个DOS程序，然后题目也说了asm，asm指Assembly，所以说这题是需要看汇编代码

我们将可执行文件拖入IDA查看，首先是能够看到flag的一个形式



```
dseg:0000 ; Entry Point : 100D:0
dseg:0000
dseg:0000 .686p
dseg:0000 .mmx
dseg:0000 .model large
dseg:0000
dseg:0000 ; =====
dseg:0000
dseg:0000 ; Segment type: Pure data
dseg:0000 dseg segment para public 'DATA' use16
dseg:0000 assume cs:dseg
dseg:0000 aHgameFillInYou db 'hgame{Fill_in_your_flag}',0
dseg:0019 db 0
dseg:001A db 0
dseg:001B db 0
dseg:001C db 0
dseg:001D db 0
dseg:001E db 0
dseg:001F db 0
dseg:0020 db 0
dseg:0021 db 0
dseg:0022 db 0
dseg:0023 db 0
dseg:0024 db 0
dseg:0025 db 0
dseg:0026 db 0
dseg:0027 db 0
dseg:0028 db 0
dseg:0029 db 0
dseg:002A db 0
dseg:002B db 0
dseg:002C db 0
```



with Hex View-1)

程序首先会将si和1Ch做比较，如果相等就会走右边提示right，如果不相等就会走左边，当左边的循环走完后如果没有进入right那就会进入wrong，这里可以得出这个for循环需要让si从00H一直++到1Ch总共走28次，下面是对于程序块汇编代码的翻译

•   seg001:0000	db 91h
•   seg001:0001	db 61h ; a
•   seg001:0002	db 1
•   seg001:0003	db 0C1h
•   seg001:0004	db 41h ; A
•   seg001:0005	db 0A0h
•   seg001:0006	db 60h ; ^
•   seg001:0007	db 41h ; A
•   seg001:0008	db 0D1h
•   seg001:0009	db 21h ; !
•   seg001:000A	db 14h
•   seg001:000B	db 0C1h
•   seg001:000C	db 41h ; A
•   seg001:000D	db 0E2h
•   seg001:000E	db 50h ; P
•   seg001:000F	db 0E1h
•   seg001:0010	db 0E2h
•   seg001:0011	db 54h ; T
•   seg001:0012	db 20h
•   seg001:0013	db 0C1h
•   seg001:0014	db 0E2h
•   seg001:0015	db 60h ; ^
•   seg001:0016	db 14h
•   seg001:0017	db 30h ; 0
•   seg001:0018	db 0D1h
•   seg001:0019	db 51h ; Q
•   seg001:001A	db 0C0h
•   seg001:001B	db 17h

xor ax, ax ax是累加器寄存器，ax和ax一定相等 结果是将ax清零

mov al, [si] si是源变址寄存器，用于存放源操作数的偏移地址，将si地址上的值给al al是低8位寄存器 1表示low 是ax寄存器的低8位

shl al, 1

shl al, 1

shl al, 1

shl al, 1 左移4位

push ax push ax是把ax里的值压入堆栈。即当前esp - 4出的值变为ax的值，ax本身的值不变。

xor ax, ax 重复前面的操作

mov al, [si]

shr al, 1

shr al, 1

shr al, 1

shr al, 1 右移4位

pop bx pop bx是把当前esp的值赋给bx，并且esp + 4 (bx的值改变，esp在pop之前指向的地方的值不变，即堆栈里的哪个值不会自动清零) 此时bx = 08H

add ax, bx 相当于把这个2位16进制数左右两位呼唤

xor ax, 17h 和17H异或

add si, 1 si++

cmp al, es:[si-1] 将al和以es为段首地址以si为偏移地址的值进行比较

jz short loc\_100DD

所以我们需要先和17H异或，然后把异或的值再左右位互换，转成ascii码表示的字符，就能够得到flag了啦

一开始还没想明白，忘了逆向是flag校验的一个过程，需要倒着算回去，正着算我寻思咋算也不太对，后来想明白后算了算前俩，检验发现前两个字符是 h,g，那么也说明我们这个思路是正确的

```
easyasm.py
1 def func(inputs, num):
2     flag = ''
3     for i in range(1, num):
4         flag += chr((inputs[i-1] << 4 + inputs[i-1] >> 4) ^ 0x17)
5         # chr(inputs[i-1] ^ i)
6     return flag
7
8
9 def main():
10    inputs = [0x91, 0x61, 0x01, 0xC1, 0x41, 0xA0, 0x60, 0x41, 0x01, 0x21, 0x14, 0xC1, 0x41, 0xE2, 0x50, 0xE1, 0xE2, 0x54, 0x20, 0xC1, 0xE2, 0x60, 0x14, 0x30, 0x01, 0x51, 0xC0, 0x17]
11    flag = func(inputs, 28)
12    print flag
13
14
15
16 if __name__ == '__main__':
17     main()
```

运行: easyasm  
C:\Python27\python.exe C:/Users/76104/Desktop/HGAME2022/week1/re/easyasm/easyasm.py  
Traceback (most recent call last):  
File "C:/Users/76104/Desktop/HGAME2022/week1/re/easyasm/easyasm.py", line 16, in <module>  
 main()  
File "C:/Users/76104/Desktop/HGAME2022/week1/re/easyasm/easyasm.py", line 11, in main  
 flag = func(inputs, 28)  
File "C:/Users/76104/Desktop/HGAME2022/week1/re/easyasm/easyasm.py", line 4, in func  
 flag += chr((inputs[i-1] << 4 + inputs[i-1] >> 4) ^ 0x17)  
OverflowError: Python int too large to convert to C long

然后我尝试直接写个python脚本解决，发现位运算上我这个脚本就很奇怪，溢出的部分不会自动忽略到，然后因为想着快点拿到flag所以说先和17H异或，再把异或的结果人工把前后两位换了一下，最后用程序转ascii输出了，算是半自动吧（

16进制到ASCII字符串在线转换工具

相关工具

- 16进制到10进制转换
- 10进制到16进制转换
- 8进制到10进制转换
- 10进制到8进制转换
- 2进制到10进制转换
- 10进制到2进制转换

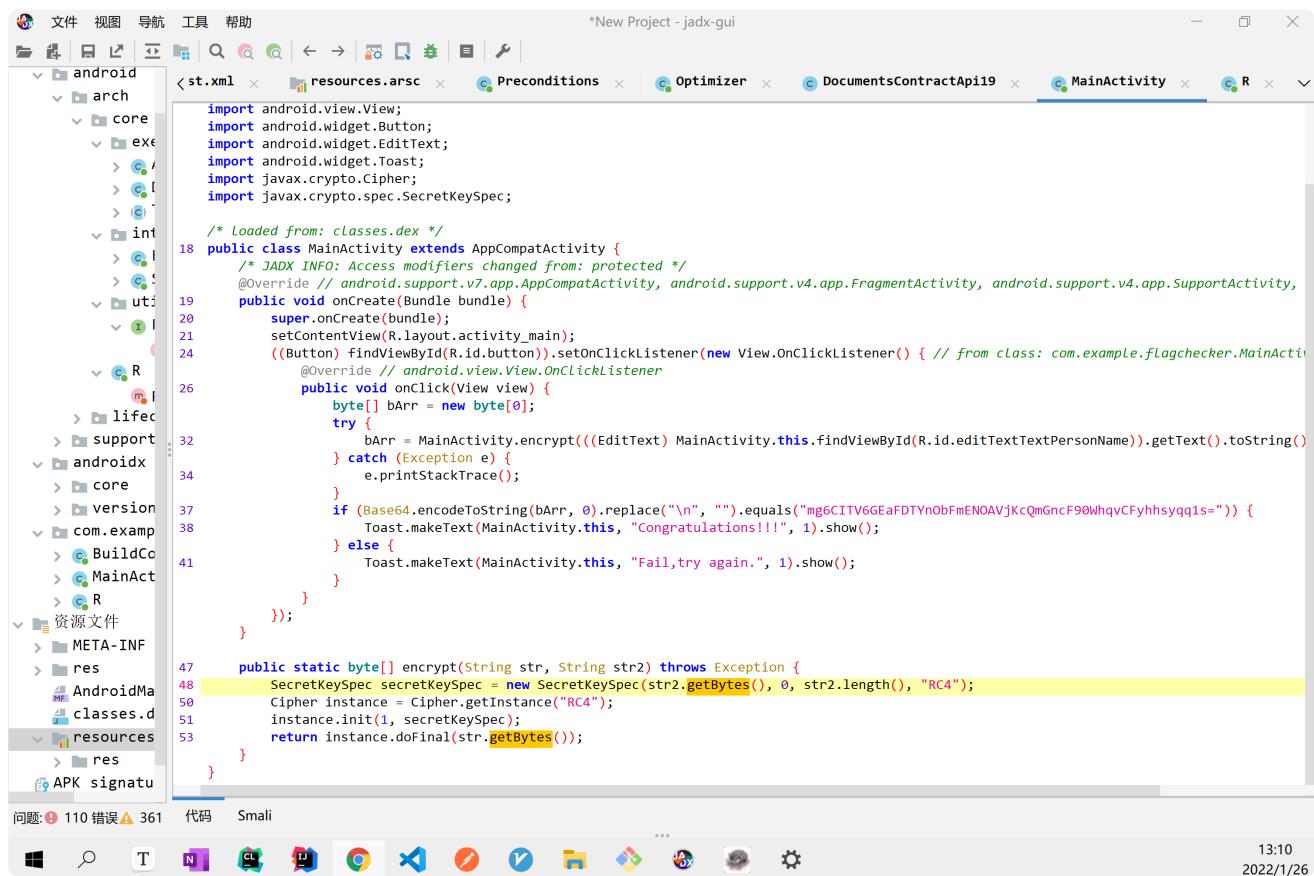
清空 交换位置 示例 转换 保存结果 复制结果

1 h•g•a•m•e•{•w•e•l•c•0•m•e•\_•t•o•\_•4•s•m•\_•w•0•r•l•d•}

## flagchecker

这是个apk逆向的题，我用了jadx-gui然后反编译了这个apk，并且找到了校验flag的类MainActivity。

从代码看flag应该是先以carol为key用RC4加密然后再base64加密后能够等同于这一串字符串，通过校验。



The screenshot shows the Jadx-GUI interface with the MainActivity.java file open. The code implements a button click listener that encrypts the text from an EditText field using RC4 and then base64 encodes it. It compares the result against a hardcoded string and shows a toast message if they match.

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

/* Loaded from: classes.dex */
public class MainActivity extends AppCompatActivity {
    /* JADe INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.SupportActivity
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        ((Button) findViewById(R.id.button)).setOnClickListener(new View.OnClickListener() { // from class: com.example.flagchecker.MainActivity
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                byte[] bArr = new byte[0];
                try {
                    bArr = MainActivity.encrypt(((EditText) MainActivity.this.findViewById(R.id.editTextTextPersonName)).getText().toString());
                } catch (Exception e) {
                    e.printStackTrace();
                }
                if (Base64.encodeToString(bArr, 0).replace("\n", "").equals("mg6CITV6GEaFTDYNobFmENOAVjKcQmGncF90WhqvCFyhhsyqq1s=")) {
                    Toast.makeText(MainActivity.this, "Congratulations!!!", 1).show();
                } else {
                    Toast.makeText(MainActivity.this, "Fail, try again.", 1).show();
                }
            }
        });
    }

    public static byte[] encrypt(String str, String str2) throws Exception {
        SecretKeySpec secretKeySpec = new SecretKeySpec(str2.getBytes(), 0, str2.length(), "RC4");
        Cipher instance = Cipher.getInstance("RC4");
        instance.init(1, secretKeySpec);
        return instance.doFinal(str.getBytes());
    }
}
```

那么我们只需要把这个里面明文表示的字符串先base64解码，再RC4解码就可以得到flag了，这里使用了cyberchef这个在线解密网站。

The screenshot shows the CyberChef interface with a 'From Base64' recipe selected. The input is a long string of characters: 'mg6C1TV6GEafDTYnObFmENoAVjKcQmGncF90uhqvCfyhhsyqq1s='.

The output section shows the decoded result: 'hgame{welCOME\_To\_tHE\_WORLd\_oF\_AnDr0}'.

The interface includes tabs for 'Operations' (RC, CRC, RC4, etc.), 'Input' (Alphabet, Passphrase), and 'Output' (Input format: Latin1, Output format: Latin1). A 'BAKE!' button is at the bottom.

## PWN

### test nc

只需要用用nc命令连接靶机，然后靶机直接给了我们shell，那这样的话cat flag我们就拿到flag啦

然后由于pwn一直被扫端口现在pwn的靶机连接前都需要进行身份验证了，我也是没有这个爆破这个sha256前4位的脚本没法复现只能文字描述一下啦。