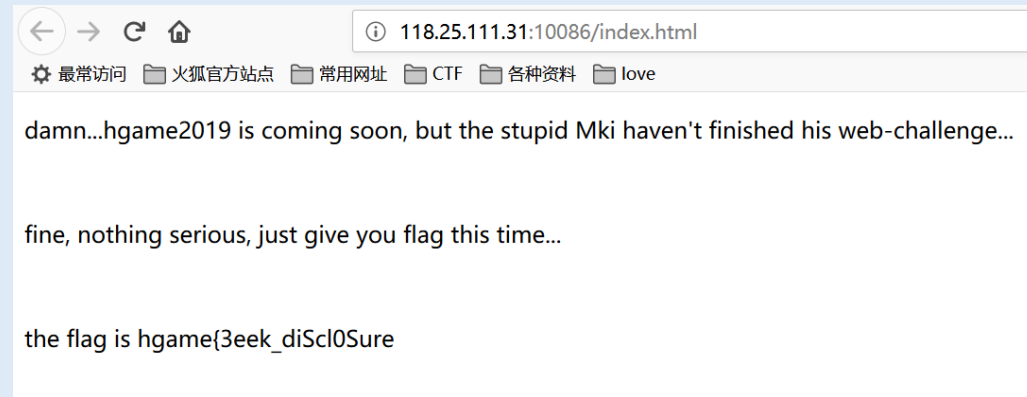


## Web1 谁吃了我的 flag

URL: <http://118.25.111.31:10086/index.html>

一开始真的没头绪……看了源码抓了包都没有什么用。直到后面放出了 hint 才知道做题的思路。



首先 flag 的开头两个单词 seek disclosure 是寻找泄露的意思，然后提示里说作者是用 vim 编辑器写代码的，但是关机前忘记保存。

我上网搜了一下 vim 泄露之类的资料就出来了。说是 vim 在异常退出的时候会备份一个.swp 的文件，只要把 URL 后面的文件名\*.html改成 \*.html.swp 就能下载这个 swp 文件，查看源码就能知道这个文件的信息。

因此我们这样构造：



打开它

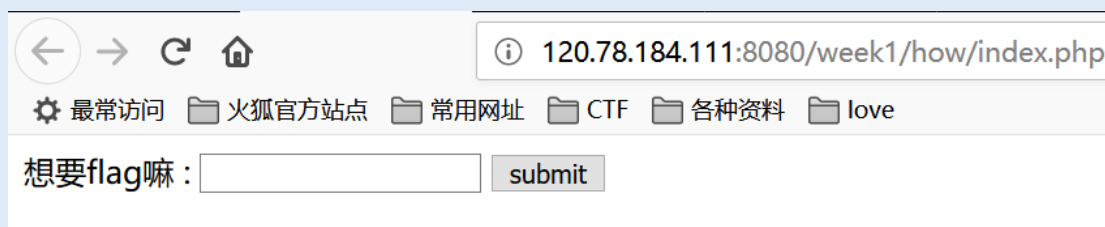
```
index.html.swp - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

</h</html>          </body>          <p>the flag is
hgame{3eek diScI0Sure fRom+wEbsit@}  </br>          <p>fine, nothing
serious, just give you flag this time...</p>  </br>
<p>damn...hgame2019 is coming soon, but the stupid Mki haven't finished his web-
challenge...</p>  <body>  </head>          <title>璋恬念浜溪塚鑽刧lag??</title>  <head>
<html> <!DOCTYPE HTML>
```

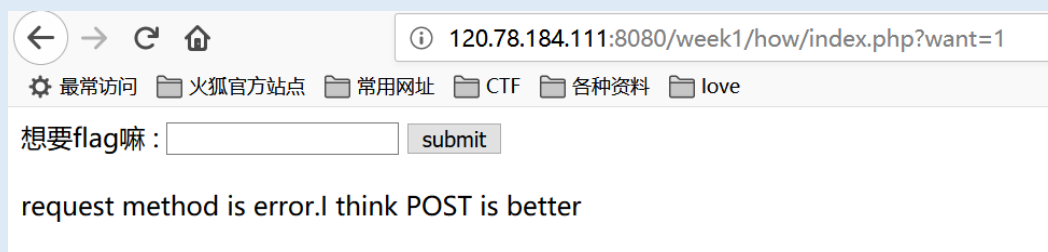
Flag 就在这里

## Web2 换头大作战

URL: <http://120.78.184.111:8080/week1/how/index.php>



直接输入 1, 点 submit



提示说用 post 方法 (刚才用的是 get)

那么用一下火狐插件 hackbar:

想要flag嘛：

https://www.wikiwand.com/en/X-Forwarded-For  
only localhost can get flag

🔍 查看器   性能   控制台   调试器   {} 样式编辑器   内存

Encryption ▾   Encoding ▾   SQL ▾   XSS ▾   Other ▾   Chrome Back

Load URL

Split URL

Execute

☒ Post data   ☐ Referrer   ☐ User Agent   ☐ Cookies

根据提示，在请求里加上 X-Forwarded-For: 127.0.0.1 伪装成 localhost（按 f12，然后选择网络，选择第一个 index.php，点击编辑和重发）：

新请求 发送 取消

POST

请求头:

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2  
Accept-Encoding: gzip, deflate  
Referer: http://120.78.184.111:8080/week1/how/index.php?want=1  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 6  
Connection: keep-alive  
Cookie: admin=0  
Upgrade-Insecure-Requests: 1  
X-Forwarded-For: 127.0.0.1

请求主体:

消息头

Cookie

参数

响应

耗时

堆栈跟踪

▼ 预览

想要flag嘛：

submit

https://www.wikiwand.com/en/User\_agent

please use Waterfox/50.0

▼ 响应载荷 (payload)

1

2

3

4

5

6

7

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>换头大作战</title>

根据提示，改请求头里的 User-Agent（最后一项 firefox 改成 waterfox/50.0）：

新请求

发送

取消

POST

http://120.78.184.111:8080/week1/how/index.php

请求头:

Host: 120.78.184.111:8080

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Waterfox/50.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Referer: http://120.78.184.111:8080/week1/how/index.php

Content-Type: application/x-www-form-urlencoded

Content-Length: 6

Connection: keep-alive

请求主体:

want=1

消息头

Cookie

参数

响应

耗时

堆栈跟踪

▼ 预览

想要flag嘛：

submit

https://www.wikiwand.com/en/HTTP\_referer

the requests should referer from www.bilibili.com

继续，改 Referer：

新请求

发送 取消

POST

http://120.78.184.111:8080/week1/how/index.php

请求头:

Host: 120.78.184.111:8080  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Waterfox/50.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2  
Accept-Encoding: gzip, deflate  
Referer: www.bilibili.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 6  
Connection: keep-alive  
Cookie: admin=0

请求主体:

want=1

想要flag嘛 :

submit

https://www.wikiwand.com/en/HTTP\_cookie  
you are not admin

那么把 cookie 里面的 admin 改成 1 试试

新请求

发送 取消

POST

http://120.78.184.111:8080/week1/how/index.php

请求头:

Referer: www.bilibili.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 6  
Connection: keep-alive  
Cookie: admin=1  
Upgrade-Insecure-Requests: 1  
X-Forwarded-For: 127.0.0.1  
Pragma: no-cache  
Cache-Control: no-cache

请求主体:

want=1

点击发送!

消息头

Cookie

参数

响应

耗时

堆栈跟踪

▼ 预览

想要flag嘛 :

submit

hgame{hTTp\_HeaDeR\_iS\_Ez}

得到 flag 啦

## Web3 very easy web

URL: [http://120.78.184.111:8080/week1/very\\_ez/index.php](http://120.78.184.111:8080/week1/very_ez/index.php)

是道代码审计题

```
<?php
error_reporting(0);
include("flag.php");

if(strpos("vidar", $_GET['id']) !== FALSE)
    die("<p>干巴爹</p>");

$_GET['id'] = urldecode($_GET['id']);
if($_GET['id'] === "vidar")
{
    echo $flag;
}
highlight_file(__FILE__);
?>
```

看到 urldecode 函数，觉得应该和 URL 编码有关，上网搜了一下发现有类似的例子，只要将 vidar 进行两次 URL 编码就行了（实际上只要有一个字母二次编码就能绕过 strpos 的判断了，而且字母进行 urldecode 依然是原字母）。于是查了 URL 编码表，字母 v 对应 %76，符号 % 对应 %25，数字 76 就是 76，那么构造 id=%2576idar：

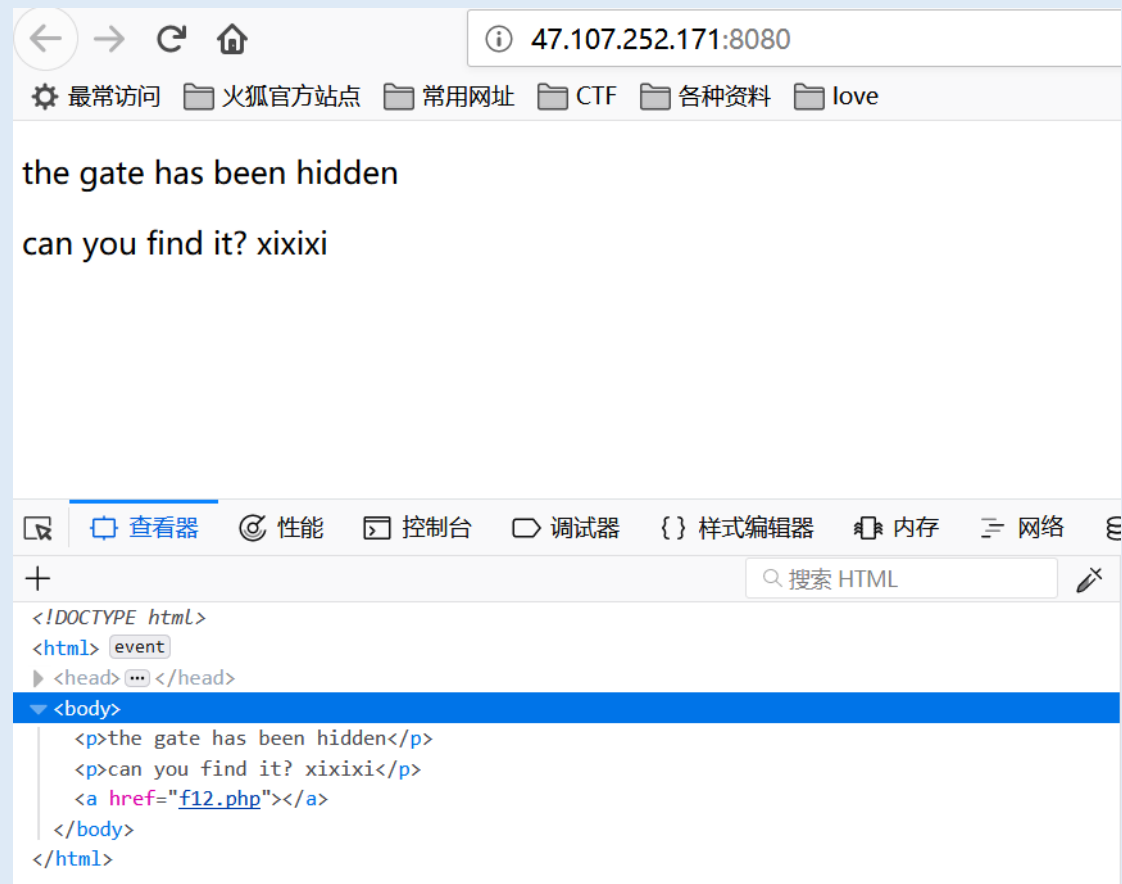


成功得到 flag

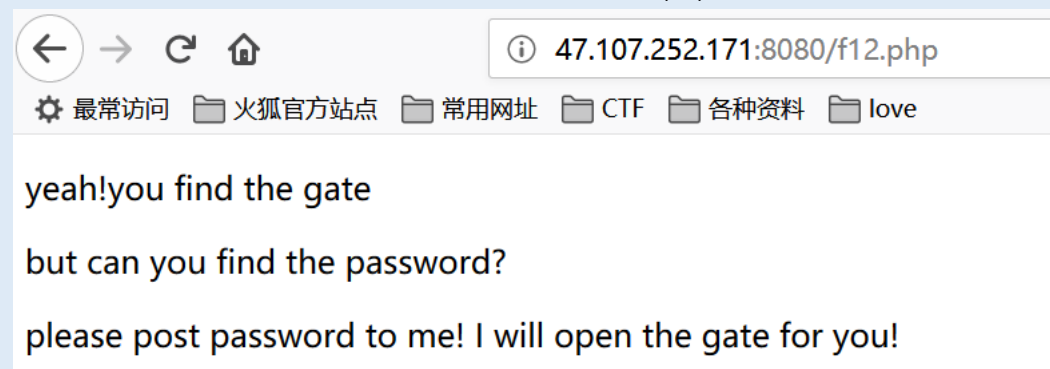
## Web4 can u find me?

URL: <http://47.107.252.171:8080/>

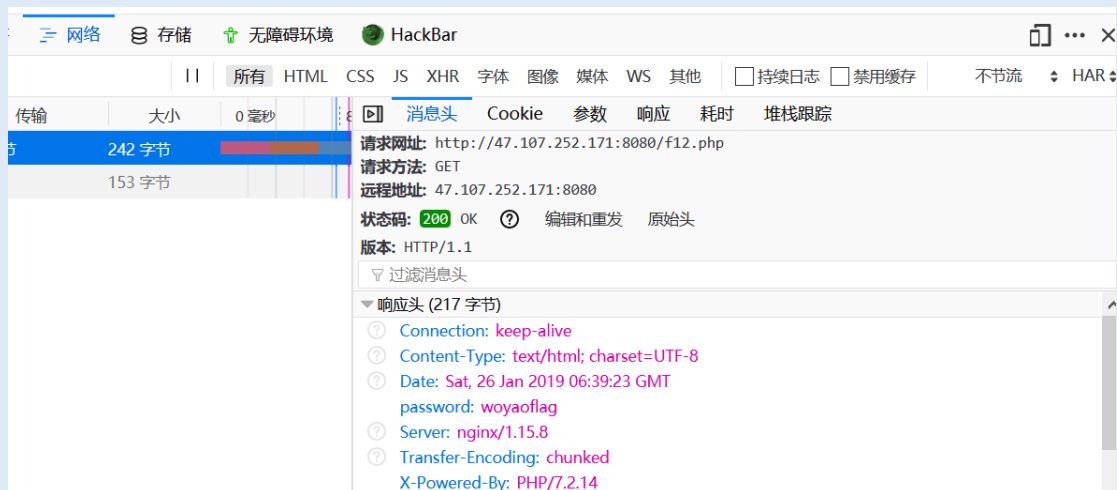
进去先 f12 看源码:



发现有个隐藏链接, 那么我们在 URL 后面加上/f12.php 试试:



看了下源码, 没有隐藏信息, 那再看看网络



找到 password 了！那么用 post 方法传过去试试，结果得到了一个链接：

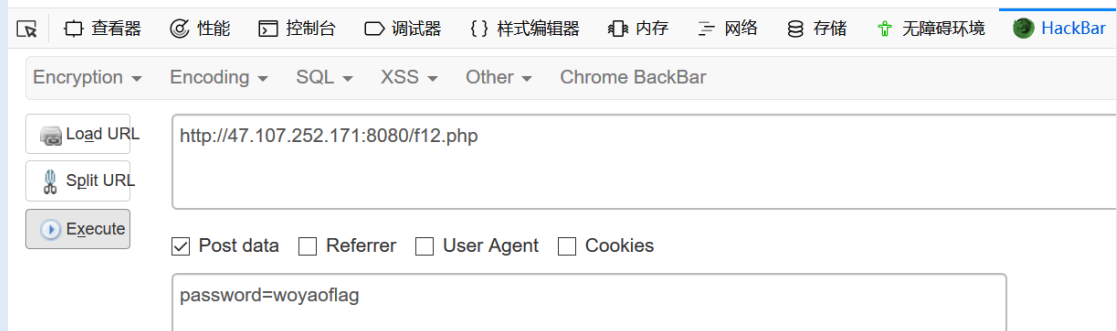
yeah!you find the gate

but can you find the password?

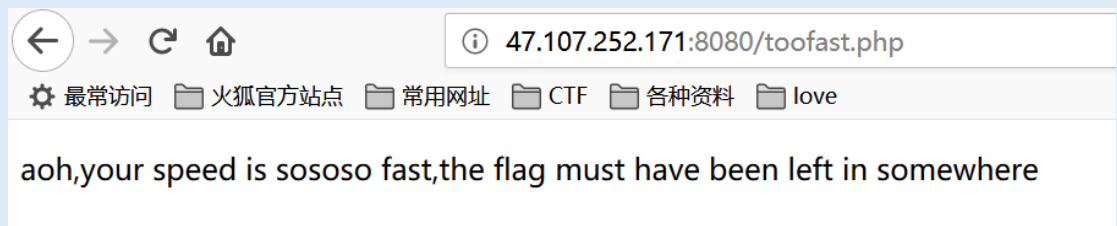
please post password to me! I will open the gate for you!

right!

[click me to get flag](#)

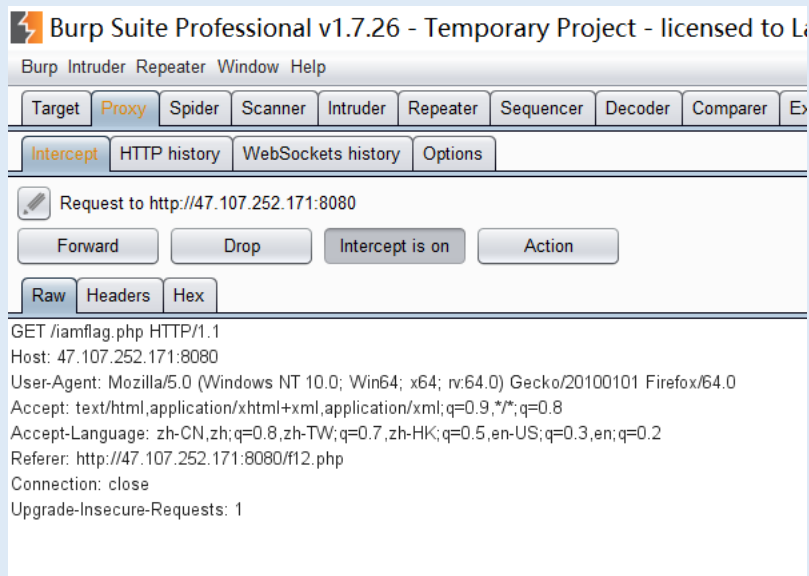


这时候由于之前的做题经验，我有了一些警惕（估计没这么简单）。点进去看了一下，果然不出所料：

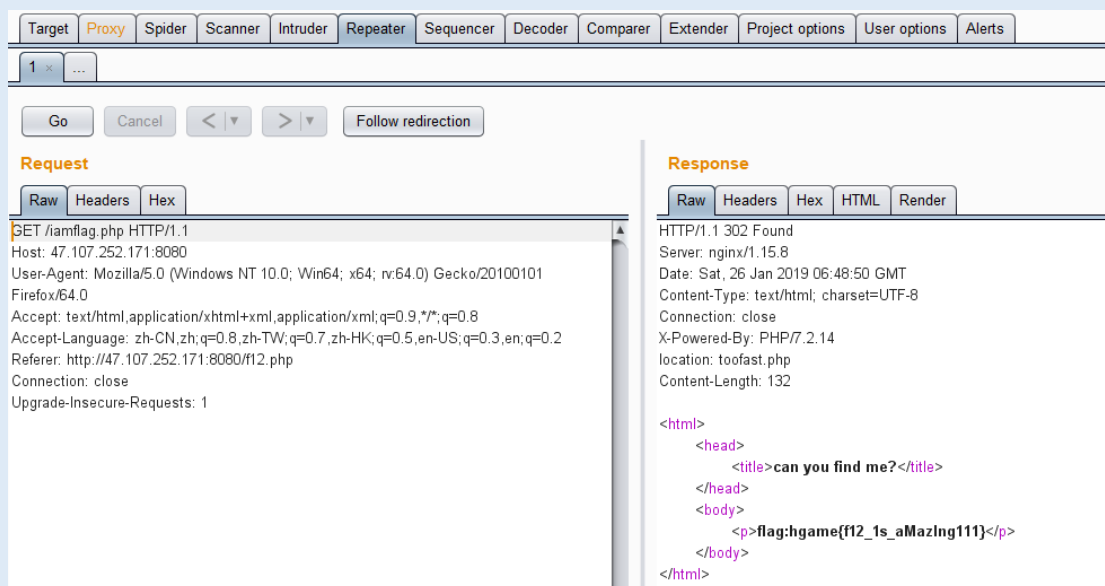


打开浏览器代理选项，然后打开 burpsuite 进行抓包：





右键 send to repeater, go 一下:



Flag 到手啦

## Misc1 Hidden Image in LSB

URL: <http://plir4axuz.bkt.clouddn.com/hgame2019/lbsb.zip>

下载压缩包并打开，一个是 png 图像，一个是.py 文件【用 python 写的代码】。

我先分析了一下 png 图片，查看了下它的属性和源码，没有有用的信息；拿到 010editor 里查看了一下其他信息，顺便改了下图片高度试试，也没有发现隐藏信息。然后打开.py 文件看了下里面的代码，

```
lsb.py
1  # coding=utf-8
2  # author=oyiadin
3
4  # Jan 18, 2019
5  # Welcome to HCTF Game 2019, have fun!
6
7  # I generated the picture attached by running:
8  # python lsb.py --watermark -ir original-power-source.png -iw flag.png -o power-source.png
9
10 # after you have finished your own extract_watermark function
11 # you can extract the flag image by running:
12 # python lsb.py --extract -ir power-source.png -o flag.png
13
14 import argparse
15 import numpy as np
16 from PIL import Image
17
18
19 parser = argparse.ArgumentParser(
20     description='An implement of image steganography')
21
22 add_or_extract_group = parser.add_mutually_exclusive_group(required=True)
23
24 add_or_extract_group.add_argument(
25     '--watermark', default=False, action='store_const', const=True)
26 add_or_extract_group.add_argument(
27     '--extract', default=False, action='store_const', const=True)
28
29 parser.add_argument('-ir', type=str, metavar='IN_FILE', required=True)
30 parser.add_argument('-iw', type=str, metavar='IN_FILE', required=False)
31 parser.add_argument('-o', type=str, metavar='OUT_FILE', required=True)
32
33
34 def apply_watermark(ir: Image, iw: Image, o: str):
35     # ir: image_raw, iw: image_watermark, o: output_image
36     wd, ht = ir.size
37     ir = ir.convert('RGB')
38     iw = iw.convert('RGB')
39
40     ir = (np.array(ir.getdata()) >> 2) << 2
41
42
43
44
45
46
47
48
49
50
51 def extract_watermark(ir, o):
52     raise NotImplementedError
53     # you need to implement this function on your own
54
55
```

这是要我们自己写代码的节奏啊……python 我是会那么一点点但是这个代码我还是不太看得懂，而且还要自己写一个函数，我就先把这道题放着了。

hint 放出来之后：

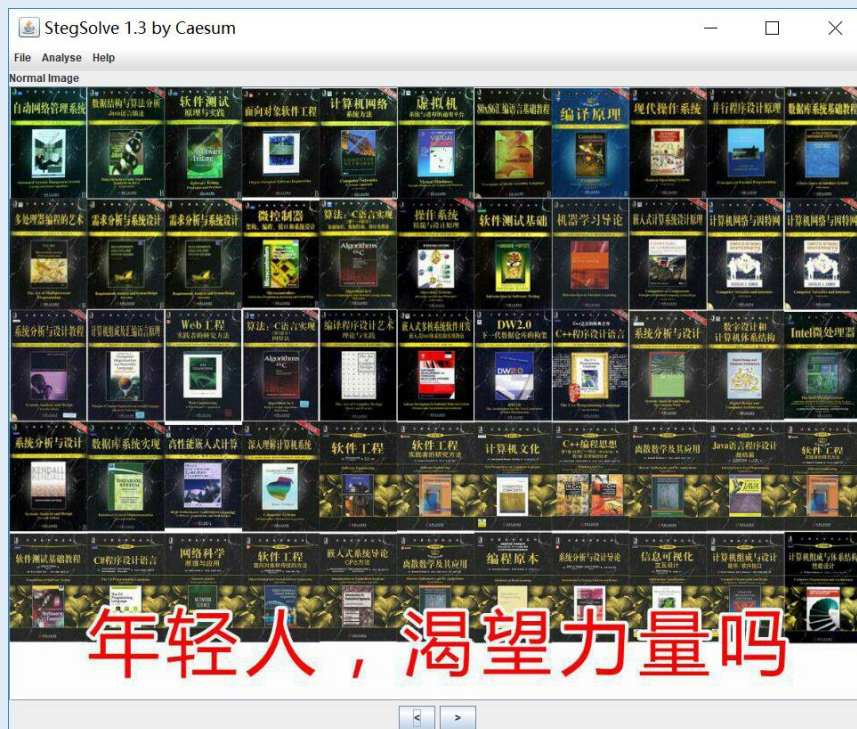
Description

Here are some magic codes which can hide information in an ordinary picture, can you extract the hidden image in the provided picture?

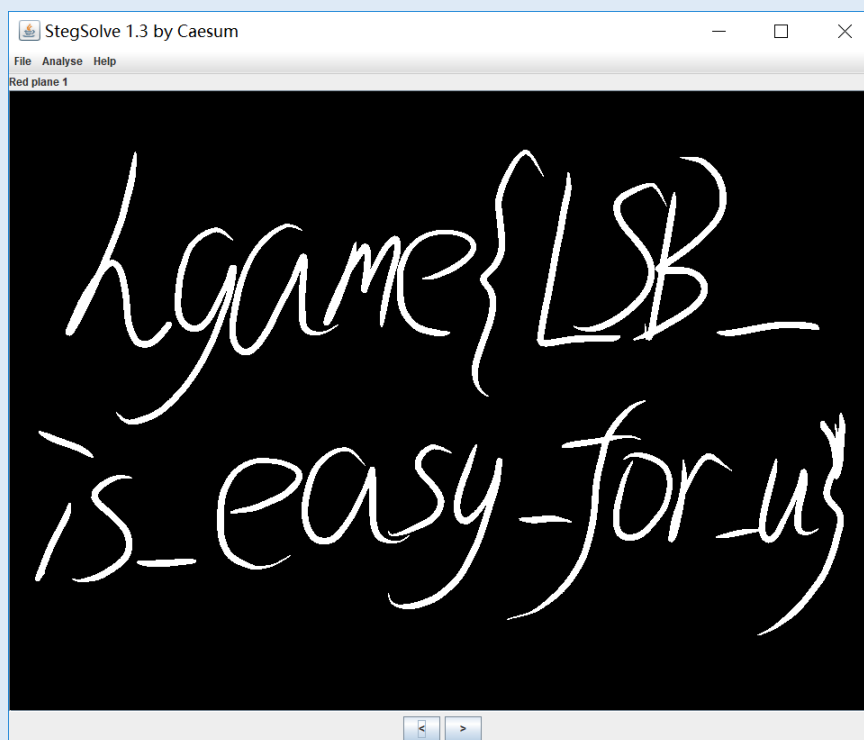
其实本来想让大家写写代码，后来干脆就送分了

有个神器叫 stegsolve，利用它可以直接提取本题 flag

打开 stegsolve，打开图片，按底下的>或者<按键



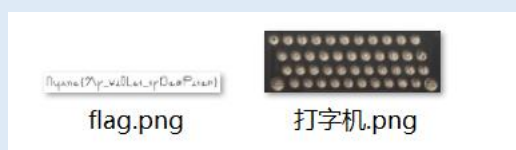
在这一层找到了 flag:



Misc2 打字机

URL: <http://plps4kyke.bkt.clouddn.com/%E6%89%93%E5%AD%97%E6%9C%BA.zip>

解压出来得到两张图片，一张叫 flag，一张叫打字机：



hgame{My\_violet\_tyPewRiter}



Flag 这张图很明显就是 flag 的格式，hgame{\*\*\_\*\*\*\*\*\_\*\*\*\*\*}。那么就知道了花括号{}前面的几个符号对应的字母了；然后根据打字机这张图片对应普通电脑键盘的字母，就可以知道 flag 里的其他几个符号对应的字母（和数字）。

目前能比较确定的是 hgame{M\*\_\*\*0Le\*\_\*\*PewR\*\*er}

因为一般的 flag 都是由有意义的单词构成的，那么我就盲猜花括号里的第一个单词是 my，得到：hgame{My\_\*\*0Le\*\_\*\*yPewR\*\*er}

这时候基本能看出原来的单词构造了，因为题目是打字机，估计最后一个单词就是 typewriter 了，得到 hgame{My\_\*i0Let\_tyPewRiter}。

最后一个字母，为了组成一个完整单词，基本确定为 v，得到 hgame{My\_violet\_tyPewRiter}。

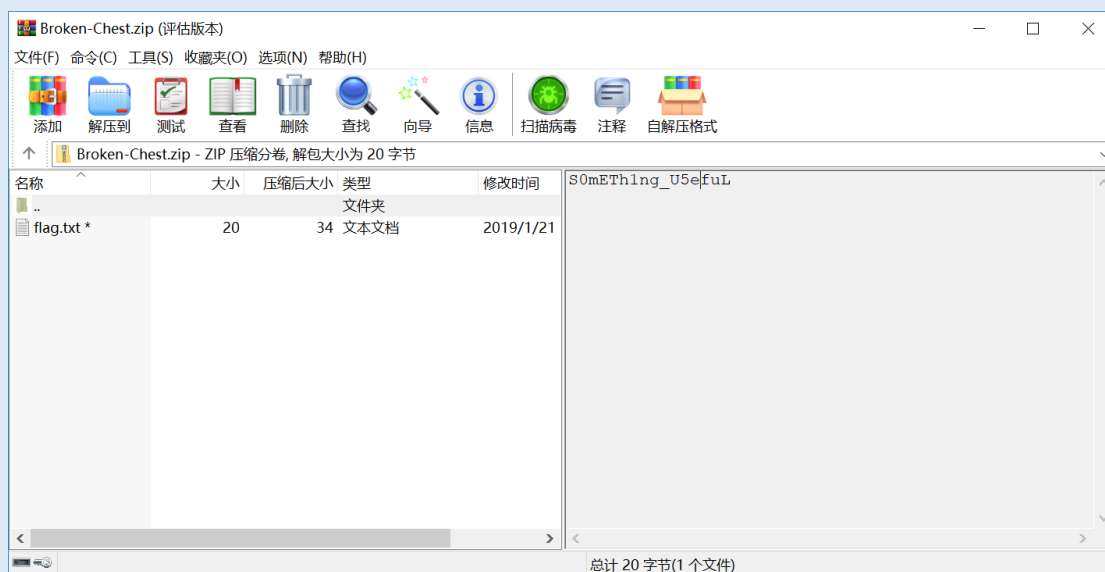
最后提交 flag，发现错误，我又想了想是不是某些字母的大小写有问题，反复试了几次后最后发现 l 是小写。

所以正确的 flag 应该是 hgame{My\_violet\_tyPewRiter} 【我没记错的话】

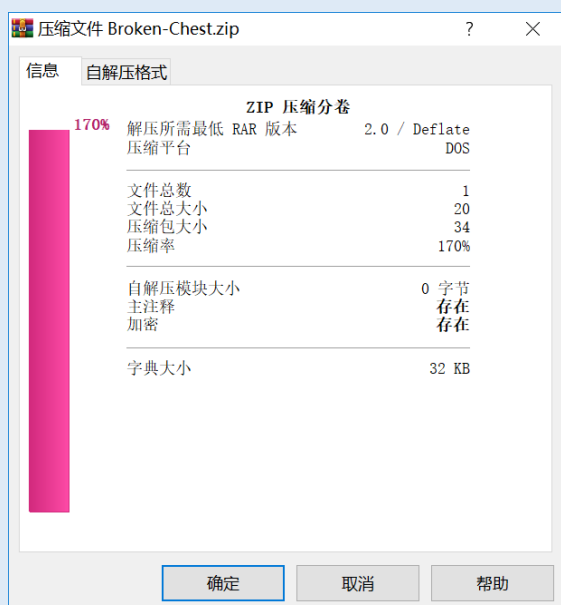
## Misc3 Broken Chest

URL: <http://plqfgjy5a.bkt.clouddn.com/Broken-Chest.zip>

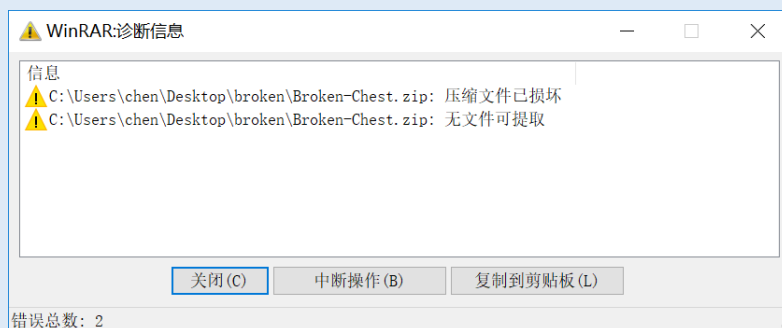
打开下载的压缩包后



看到一个注释（something useful? ），可能是密码，查看了一下压缩文件的信息，发现果然是加密的。



尝试解压文件没有提示输入密码，却提示了压缩包文件损坏。

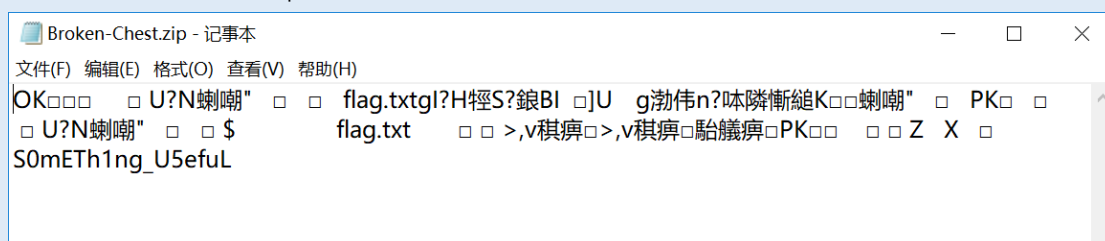


想到题目的描述里说的疯狂钻石（jojo 里一个有修复能力的角色），于是使用 WinRAR 自带的压缩包修复工具（左上角 工具->修复压缩文件）试着修复了一下：



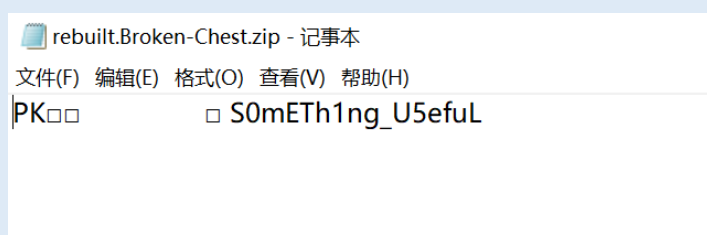
结果修复后的 rebuilt 压缩包里的文件 flag.txt 直接没了，想必不能这么干。

用记事本打开原来的 zip 文件，

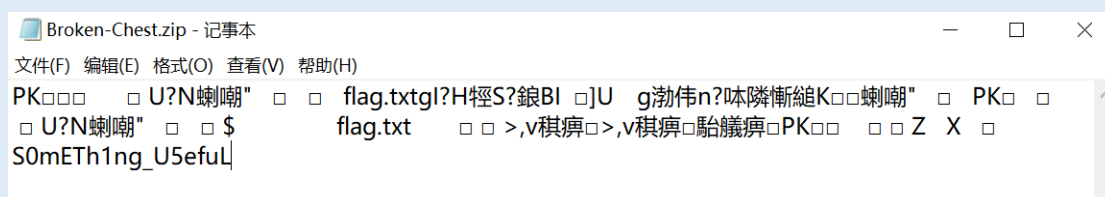


好像没什么有用的信息。

比较一下被刚才修复过的 rebuilt 文件，看看两者有什么不同：



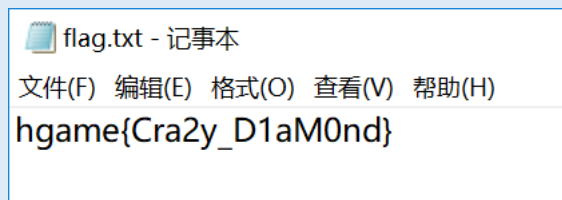
看到开头字母突然就想到了，系统判断一个文件是什么类型一般看文件头标志，比如 png 文件的文件头标志一般长这个样 `PNG`，也就是 16 进制的 89 50 4E 47。看修复过的 zip 文件头，估计 zip 文件头标志就是 PK，原文件之所以损坏是因为系统无法识别它的文件头标志。于是把原文件头的 OK 改成 PK：



保存，重新打开它并解压，输入密码：



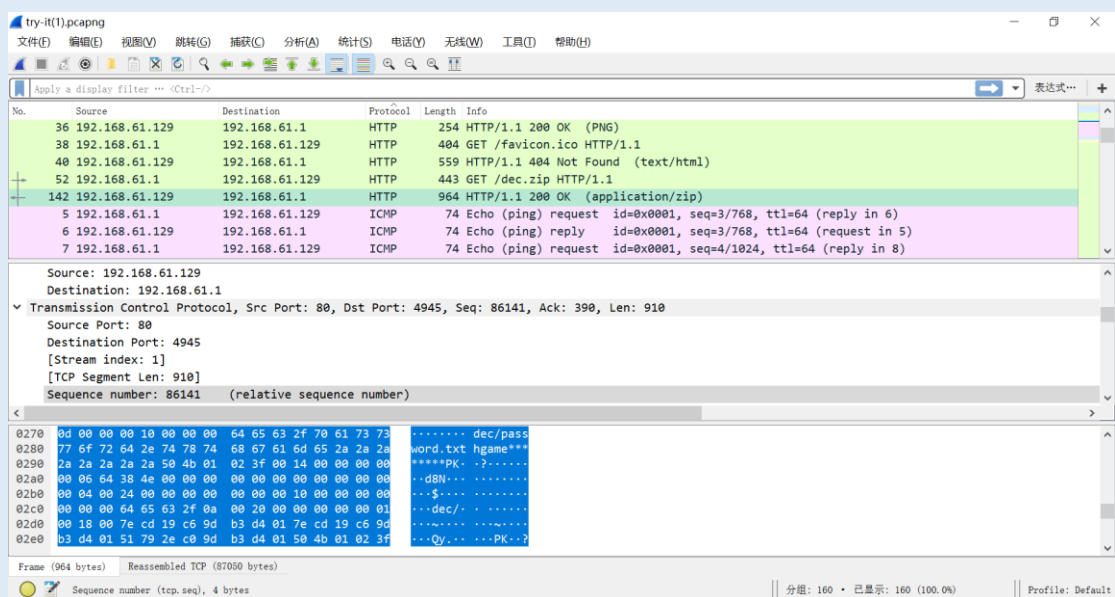
打开解压后的 flag.txt，成功得到 flag



## Misc4 Try

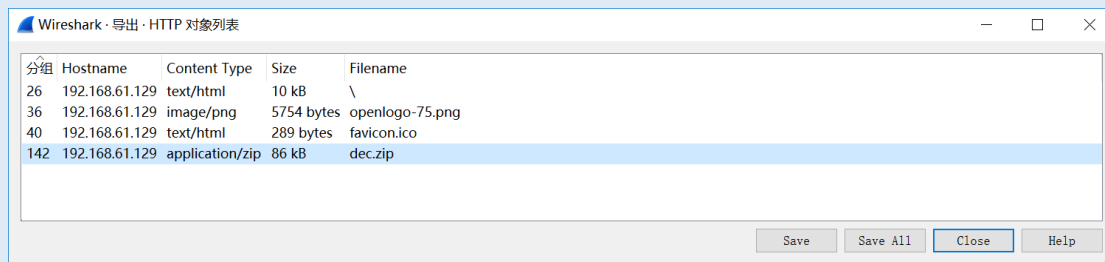
URL: <http://plqfgjy5a.bkt.clouddn.com/try-it.pcapng>

下载下来是一个.pcapng 文件，于是打开 wireshark 分析数据包：

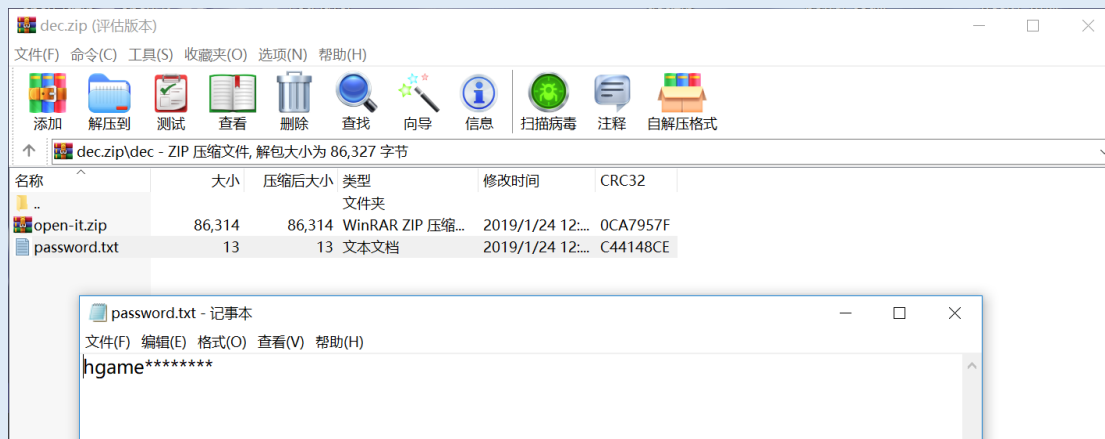


一个个看过去终于找到一个关键文件，这里似乎是下载了一个 dec.zip 文件。于是用 wireshark 提取出来【左上角，文件->导出对象->HTTP】





点 save 保存，然后打开，发现里面有两个文件



其中 open-it.zip 里有一个.jpg 文件，解压需要密码。那么密码应该就是 password.txt 里的内容了。我一开始直接把 hgame\*\*\*\*\*当成密码输入进去，结果密码错误，估计后面的\*\*\*\*\*是一串数字，还要我们自己解开密码。查看了各个文件源码并没有什么有用的信息，于是我就试试暴力破解 zip 密码。

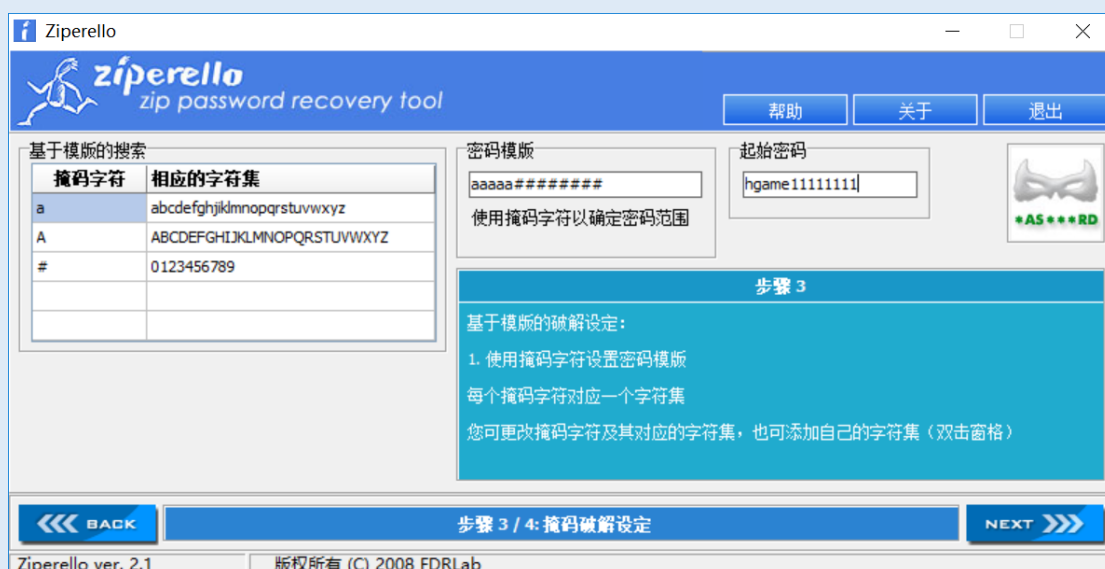
打开 zipperello，打开 open-it.zip：







这里我选了基于模板的破解，因为 password.txt 里给了密码的模板



一开始看到剩余六千多天的时候有点慌，但是因为给了起始密码 hgame11111111 所以

按顺序找起来还是比较快的。  
接下来输入密码解压 open-it.zip，里面是一张 jpg 图片。



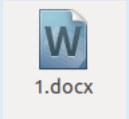
按普通套路看了图片属性、源码、拖进 stegsolve 分析之后，发现这不仅是一个 jpg 文件，好像还有 zip 的文件包含在里面。拖进 linux 虚拟机里，用 binwalk 分析了一下，的确，这个文件实际上是由一个 jpg 和一个 zip 文件组合成的：

```
root@chen537-virtual-machine:~/桌面# binwalk 1.jpg
DECIMAL      HEXADECEMAL  DESCRIPTION
-----
0             0x0          JPEG image data, JFIF standard 1.01
79837        0x137DD     Zip archive data, at least v2.0 to extract, compressed size: 9447, uncompressed size: 12178, name: 1.docx
89408        0x15D40     End of Zip archive, footer length: 22
root@chen537-virtual-machine:~/桌面#
```

用 foremost 工具分离：

```
root@chen537-virtual-machine:~/桌面# foremost 1.jpg
Processing: 1.jpg
| foundat=1.docx*ZW\K[00000000]000000005Hpw[00000000]
*|
```

打开 output->zip，里面是一个 docx 文件。



打开来，什么都没有，一片空白，还真的是“无字天书”。肯定还有什么信息漏掉了，重新 binwalk 分析一下，里面有好多内容，而且这又是一个 zip 文件

```
root@chen537-virtual-machine:~/桌面/output/zip/00000155/output/docx# binwalk 00000000.docx
DECIMAL      HEXADECEMAL  DESCRIPTION
-----
0             0x0          Zip archive data, at least v2.0 to extract, compressed size: 346, uncompressed size: 1312, name: [Content_Types].xml
915           0x393        Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
1715          0x6B3        Zip archive data, at least v2.0 to extract, compressed size: 244, uncompressed size: 817, name: word/_rels/document.xml
2281          0x8E9        Zip archive data, at least v2.0 to extract, compressed size: 830, uncompressed size: 3055, name: word/document.xml
3158          0xC56        Zip archive data, at least v2.0 to extract, compressed size: 1761, uncompressed size: 8398, name: word/theme/theme1.xml
4970          0x136A       Zip archive data, at least v2.0 to extract, compressed size: 1206, uncompressed size: 3231, name: word/settings.xml
6223          0x184F       Zip archive data, at least v2.0 to extract, compressed size: 501, uncompressed size: 1444, name: word/fontTable.xml
6772          0x1A74       Zip archive data, at least v2.0 to extract, compressed size: 295, uncompressed size: 655, name: word/webSettings.xml
7117          0x18CD       Zip archive data, at least v2.0 to extract, compressed size: 370, uncompressed size: 711, name: docProps/app.xml
7797          0x1E75       Zip archive data, at least v2.0 to extract, compressed size: 381, uncompressed size: 773, name: docProps/core.xml
8489          0x2129       Zip archive data, at least v2.0 to extract, compressed size: 2917, uncompressed size: 29131, name: word/styles.xml
12156         0x2F7C       End of Zip archive, footer length: 22
```

但是用 foremost 分离之后得到的还是原来的 docx 文件。  
我在这卡了好一会，后来把这个 docx 文件重新拖回 Windows 系统里面分析，把后缀名改成 zip 之后解压，得到了以下文件：

| 名称                  | 修改日期            | 类型     | 大小   |
|---------------------|-----------------|--------|------|
| _rels               | 2019/1/27 18:16 | 文件夹    |      |
| docProps            | 2019/1/27 18:16 | 文件夹    |      |
| word                | 2019/1/27 18:16 | 文件夹    |      |
| [Content_Types].xml |                 | XML 文档 | 2 KB |

一个个打开来看，最后在 word->document.xml 里面找到了 flag

```
C:\Users\chen\Desktop\open-it\rebuilt.1\1\word\document.xml
C:\Users\chen\Desktop\op...
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

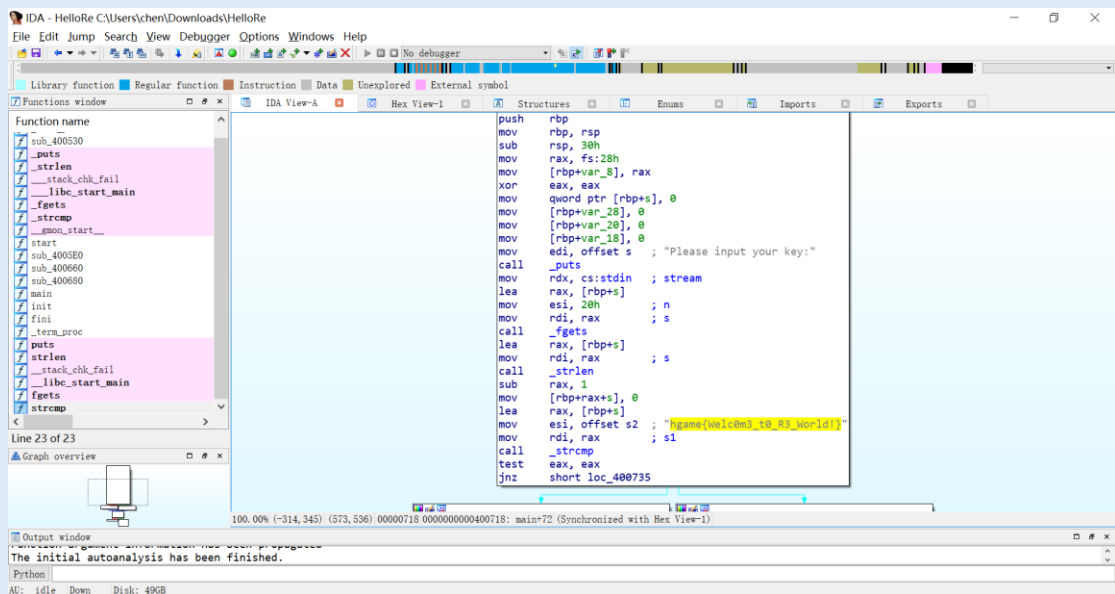
- <w:rPr>
  <w:vanish/>
</w:rPr>
</w:pPr>
<w:bookmarkStart w:name="_GoBack" w:id="0"/>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:vanish/>
    </w:rPr>
    <w:t>hgame</w:t>
  </w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:rFonts w:hint="eastAsia"/>
    <w:vanish/>
    </w:rPr>
    <w:t>{</w:t>
  </w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:vanish/>
    </w:rPr>
    <w:t>59d28413e36019861498e823f3f41406</w:t>
  </w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:rFonts w:hint="eastAsia"/>
    <w:vanish/>
    </w:rPr>
    <w:t>}</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0"/>
</w:p>
- <w:sectPr w:rsidRPr="004066DE" w:rsidR="005C0554">
  <w:pgSz w:w="11906" w:h="16838"/>
  <w:pgMar w:gutter="0" w:footer="992" w:header="851" w:left="1800" w:bottom="1440" w:right="1800" w:top="1440"/>
  <w:header="992" w:header="851" w:left="1800" w:bottom="1440" w:right="1800" w:top="1440"/>
</w:sectPr>
```

把这几部分拼起来就是 flag 了

## Re2 HelloRe

URL: <http://plps4kyke.bkt.clouddn.com/HelloRe>

下载的文件用 IDA 打开



直接获取 flag

## Re5 Pro 的 Python 教室(一)

URL: <http://plqbnxx54.bkt.clouddn.com/first.py>

【虽然我不是学逆向的，但是看到做出来的人挺多的而且我会一点点 python 所以就看了一下……】

代码审计时间到：

```
import base64
import hashlib

enc1 = 'hgame{'
enc2 = 'SGVyZV8xc18zYXN5Xw=='
enc3 = 'Pyth0n}'

print 'Welcome to Processor\'s Python Classroom!\n'
print 'Here is Problem One.'
print 'There\'re three parts of the flag.'

print '-----',

print 'Plz input the first part:'
first = raw_input()
if first == enc1:
    pass
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Plz input the secend part:'
secend = raw_input()
secend = base64.b64encode(secend)
if secend == enc2:
    pass
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Plz input the third part:'
third = raw_input()
third = base64.b32decode(third)
if third == enc3:
    pass
```

```
else:
    print 'Sorry , You\'re so vegatable!'
    exit()

print 'Oh, You got it !'
```

看了一遍代码发现就算把三个部分都输入对了，也没有任何显示 flag 之类的交互代码，我正纳闷它的工作原理的时候，就随便试了一下，把 enc1、enc2 经过 base64 解码后和 enc3 这三部分拼到一起【hgame{Here\_1s\_3asy\_Pyth0n}，因为长得比较像 flag】直接当成 flag 提交上去，结果发现 flag 正确了……（说实话我不太理解这道题的意思？）