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

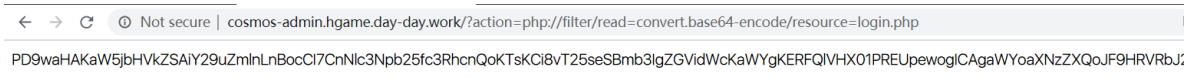
Base Score 200

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~~(? 有了提示我才想到)~~。访问 `/?action=php://filter/read=convert.base64-encode/resource=login.php`

拿到了 `login.php` 的源码



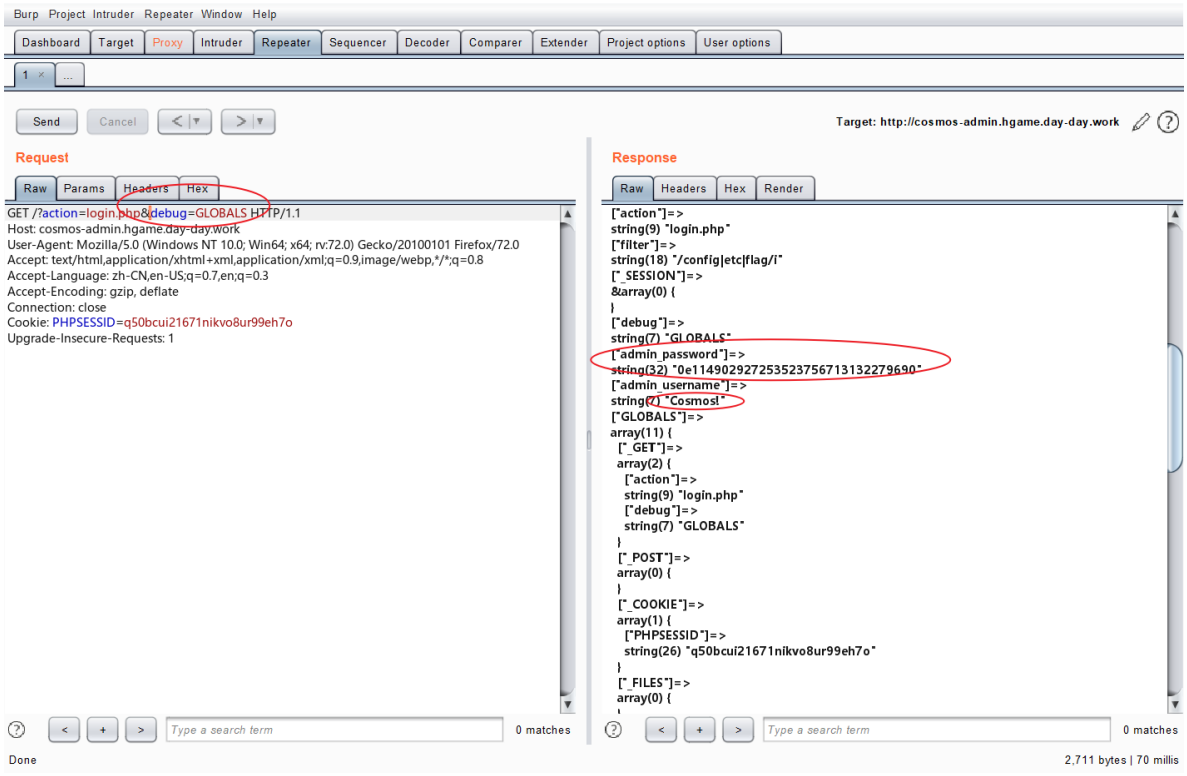
`base64` 解码以后

```php
<?php
include "config.php";
session_start();

//Only for debug
if (DEBUG_MODE){
    if(isset($_GET['debug'])) {
        $debug = $_GET['debug'];
        if (!preg_match("/^[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*$/", $debug)) {
            die("args error!");
        }
        eval("var_dump($$debug);");
    }
}

if(isset($_SESSION['username'])) {
    header("Location: admin.php");
    exit();
}
else {
    if (isset($_POST['username']) && isset($_POST['password'])) {
        if ($admin_password == md5($_POST['password']) && $_POST['username'] === $admin_username){
                $_SESSION['username'] = $_POST['username'];
                header("Location: admin.php");
            exit();
        }
        else {
            echo "用户名或密码错误";
        }
    }
}
?>
```

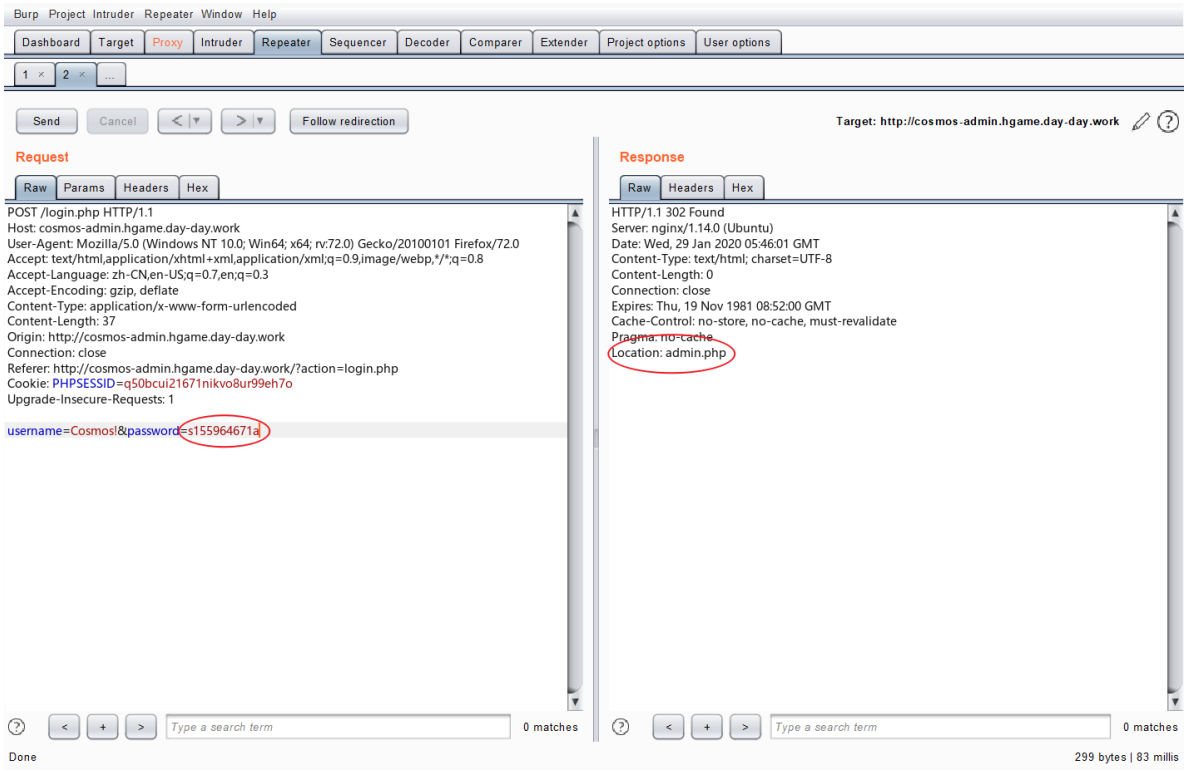
看一下登录需要的条件

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

输入的密码`md5()`加密以后和数据库里的密码要一致，可我不知道数据库里的密码呀？想用`php`伪协议读取`config.php`却看到一行`Hacker fet out!`那么就不能用这种方法读取密码了。在这里注意到有一个`DEBUG`模式，并且第`12`行有一个`eval(var_dump($$debug))`，这里有两个`$`，百度搜一下以后，发现可以在这里传入`/?action=login.php&debug=GLOBALS`，读取`php`全局变量。



`admin_password`为`0e114902927253523756713132279690`，要让`password`的`md5`值与它相等，这里考的就是`php`弱类型比较，当`md5`值为`0e`开头时，就会被当成`科学计数法`，当成数字的比较，于是两边`0==0`，结果就是`true`。百度找一个值`s155964671a`。



于是`username`值为`Cosmos!`，`password`值为`s155964671a`，`POST`登陆上去，可以看到这里成功跳转了。继续读取`admin.php`源码。

```php
<?php
include "config.php";
session_start();
if(!isset($_SESSION['username'])) {
    header('Location: index.php');
    exit();
}

function insert_img() {
    if (isset($_POST['img_url'])) {
        $img_url = @$_POST['img_url'];
        $url_array = parse_url($img_url);
        if (@$url_array['host'] !== "localhost" && $url_array['host'] !== "timgsa.baidu.com") {
            return false;
        }
        $c = curl_init();
        curl_setopt($c, CURLOPT_URL, $img_url);
        curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
        $res = curl_exec($c);
        curl_close($c);
        $avatar = base64_encode($res);

        if(filter_var($img_url, FILTER_VALIDATE_URL)) {
            return $avatar;
        }
    }
    else {
        return base64_encode(file_get_contents("static/logo.png"));
    }
}
?>
```

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`

```
<img height="200" width="500"
src="
AhfQo=">
```

`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...

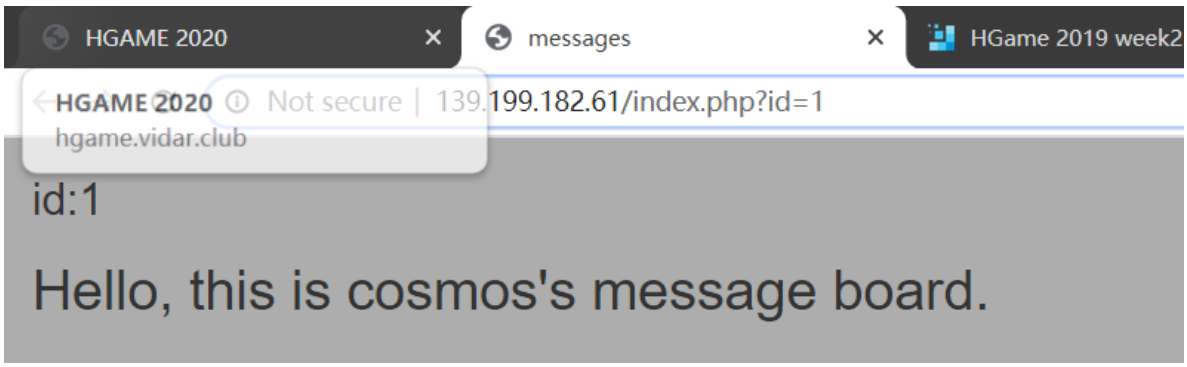
Challenge Address http://139.199.182.61/

Base Score 150

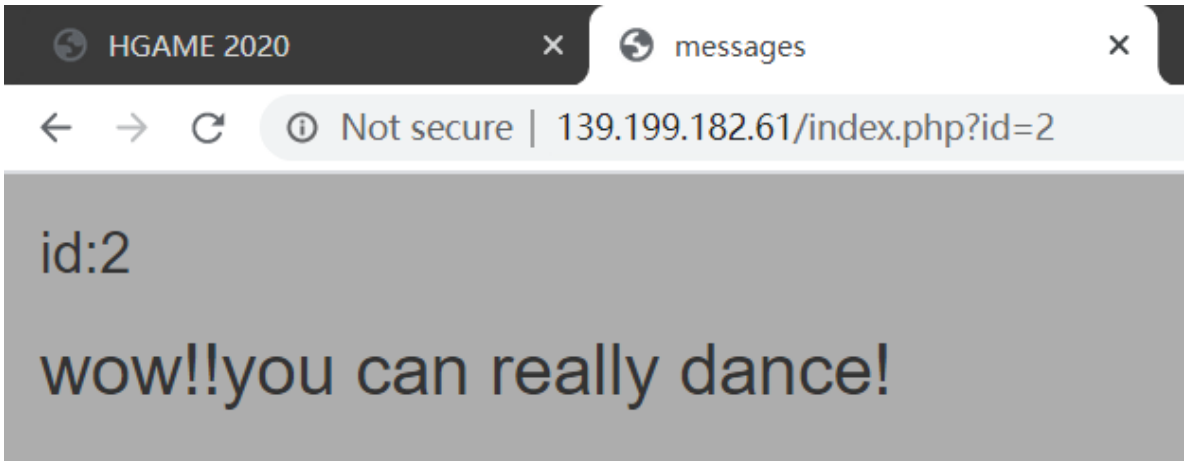
Now Score 150

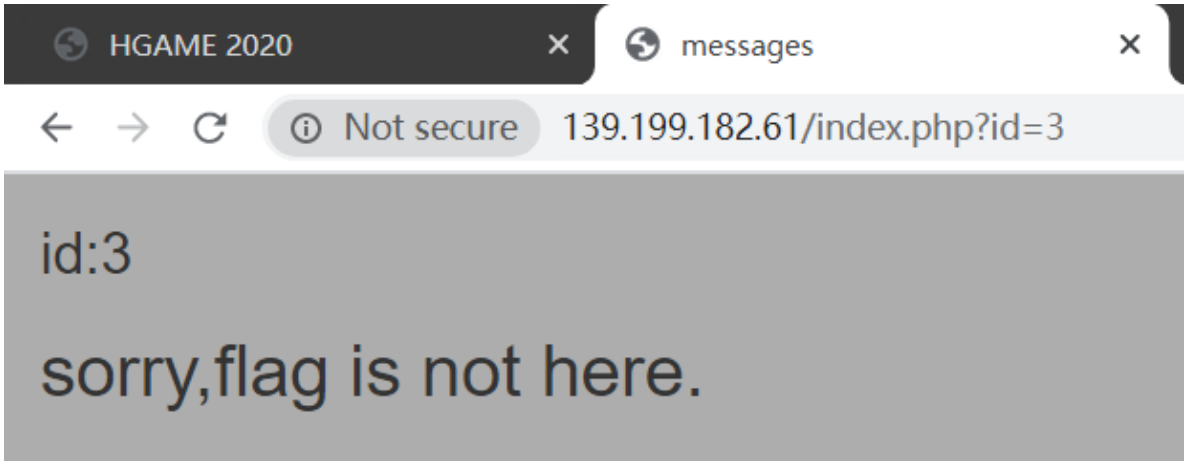
User solved 76

首先访问题目地址

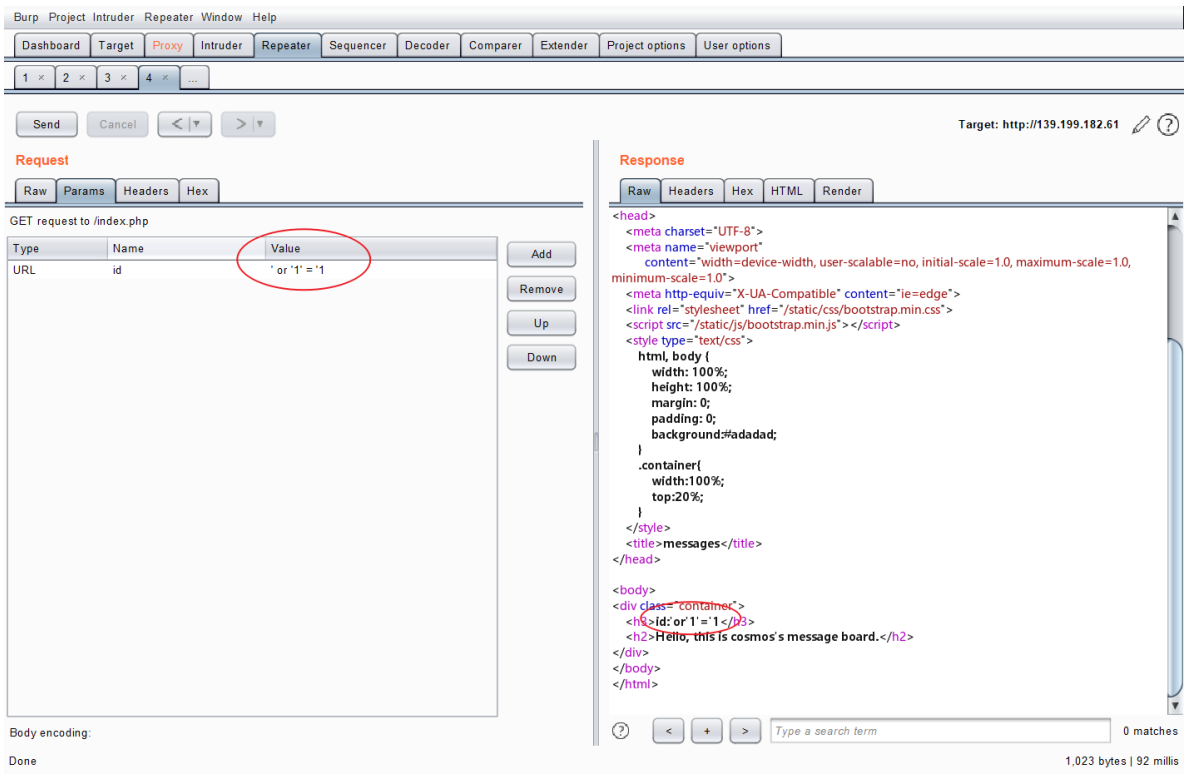


尝试一下不同的`id`

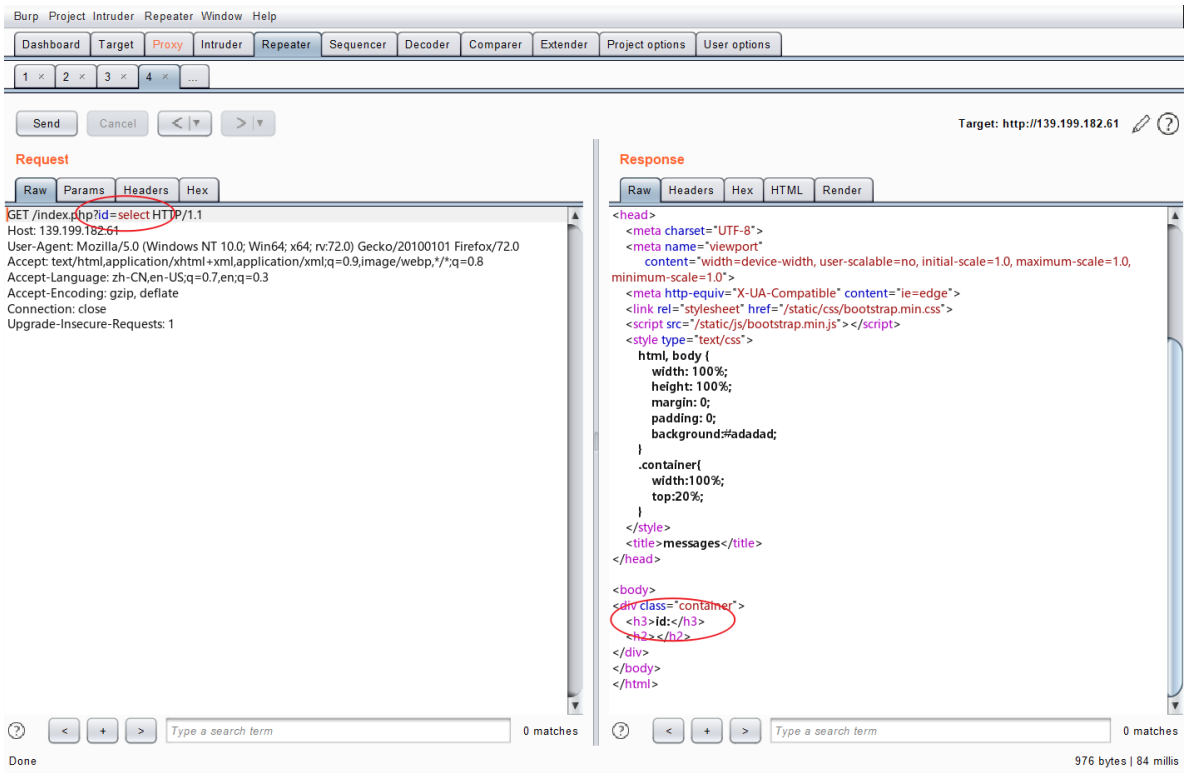


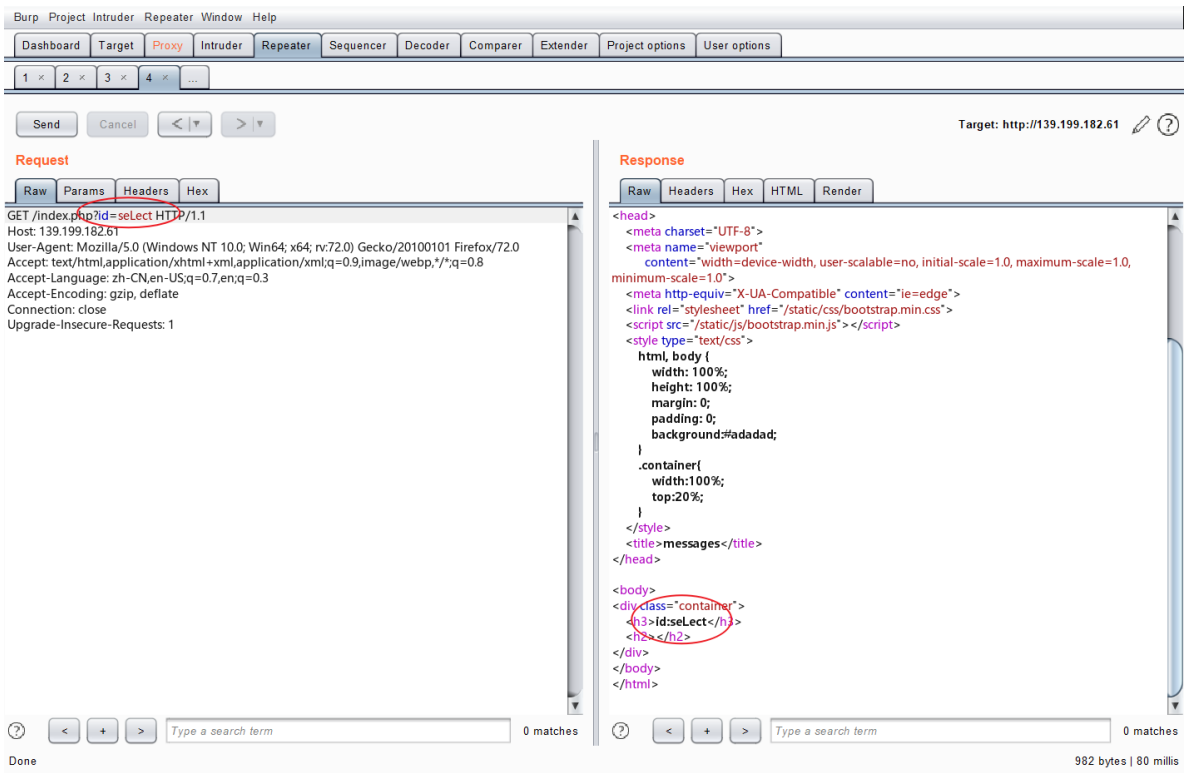


`/?id=4`再往后就没有信息了。可以猜测这是一个`SQL注入`，于是我们测试几个语句，看看它过滤了什么。



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。





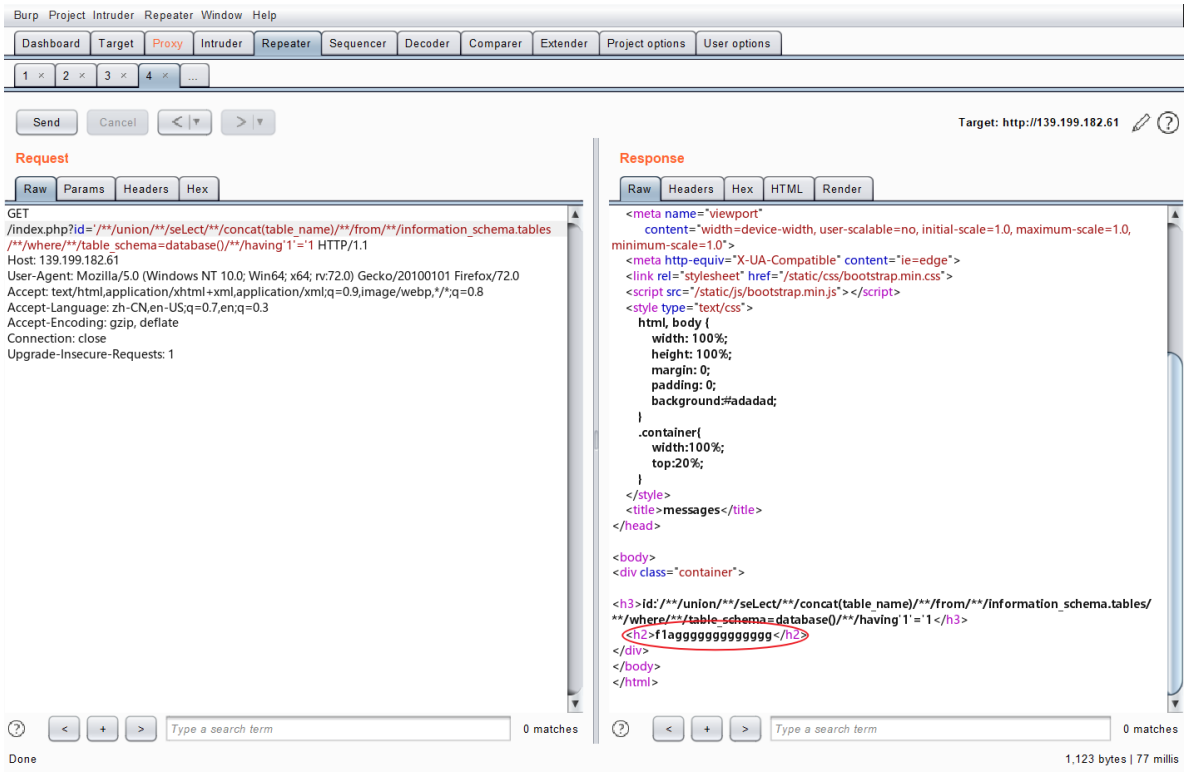
除此以外没什么发现了，于是在此卡住……实在想不出来，于是去问🍗，他让我去搜索`SQl注入+过滤空格`有关的内容，一下子就找到了。~~看来是因为我搜索关键词的能力还不够~~找到网上的这篇文章，依葫芦画瓢，`flag`就出来了。很轻松。

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。

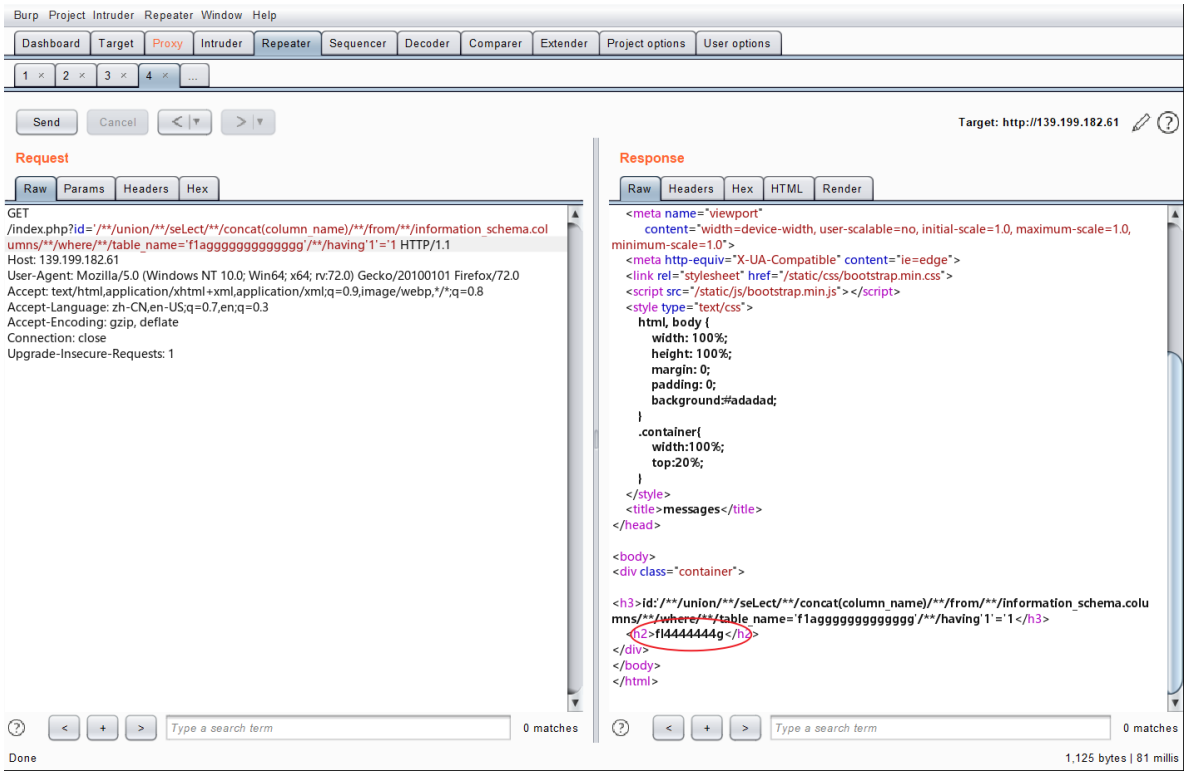
于是开始`注入`

爆表名



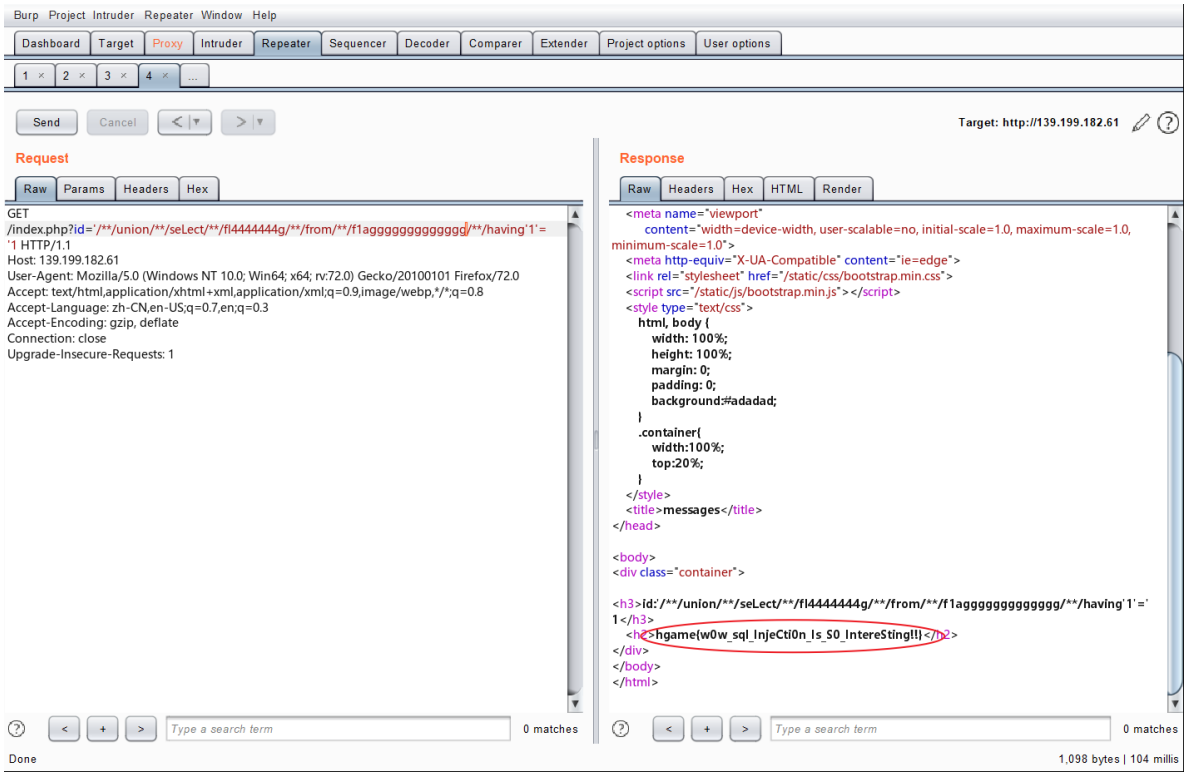
得到一个叫`f1agggggggggggg`的表

爆字段名



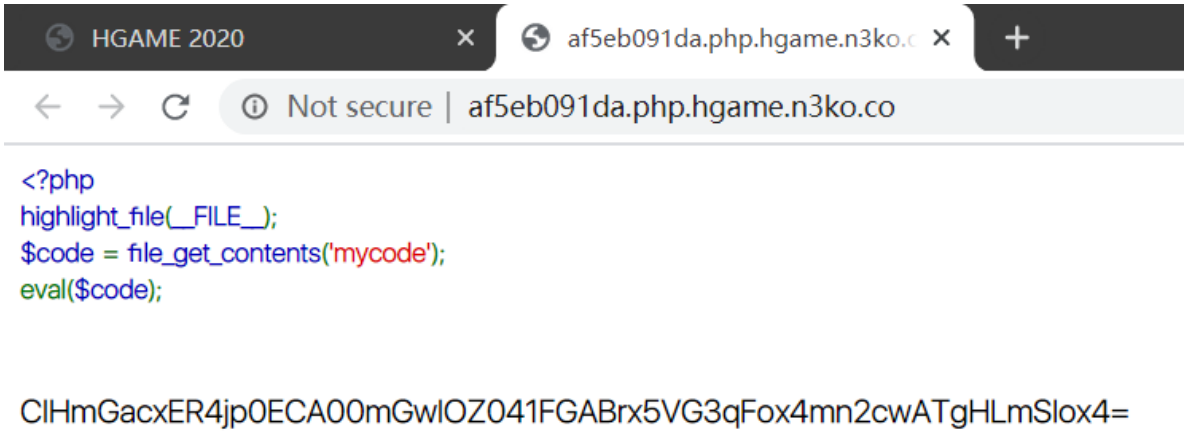
得到一个叫`fl4444444g`的字段

爆出flag

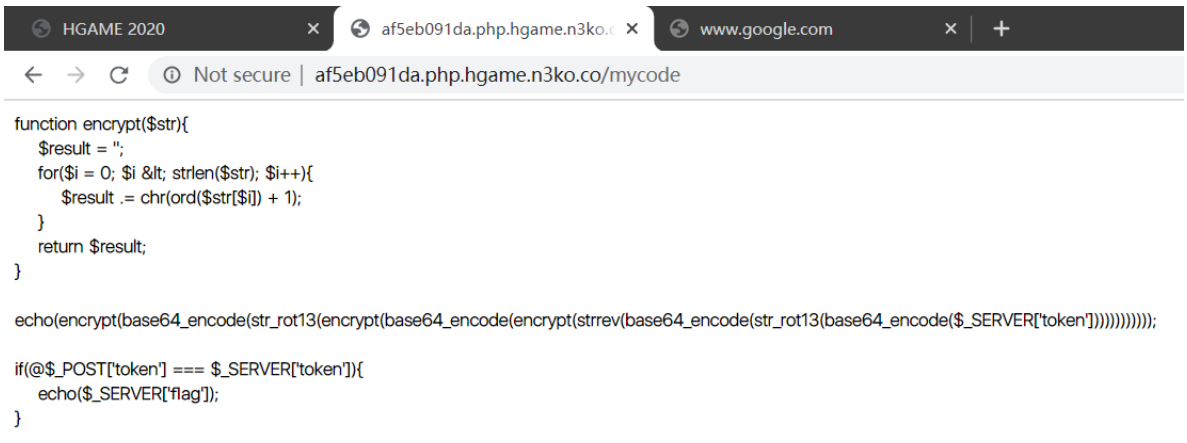


> hgame{w0w_sql_InjeCti0n_Is_S0_IntereSting!!}

Cosmos的新语言



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)
```

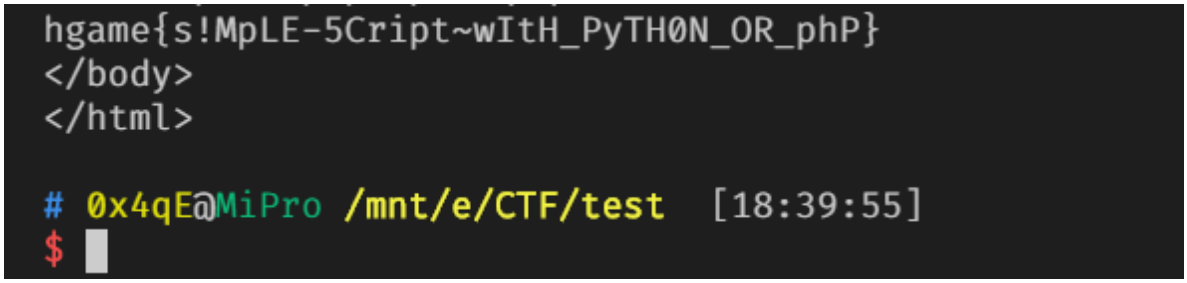
小意外

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，偶然的一次之后就又恢复正常了。



> hgame{s!MpLE-5Cript~wItH_PyTH0N_OR_phP}

Cosmos的聊天室

Cosmos的聊天室[SOLVED]

Description

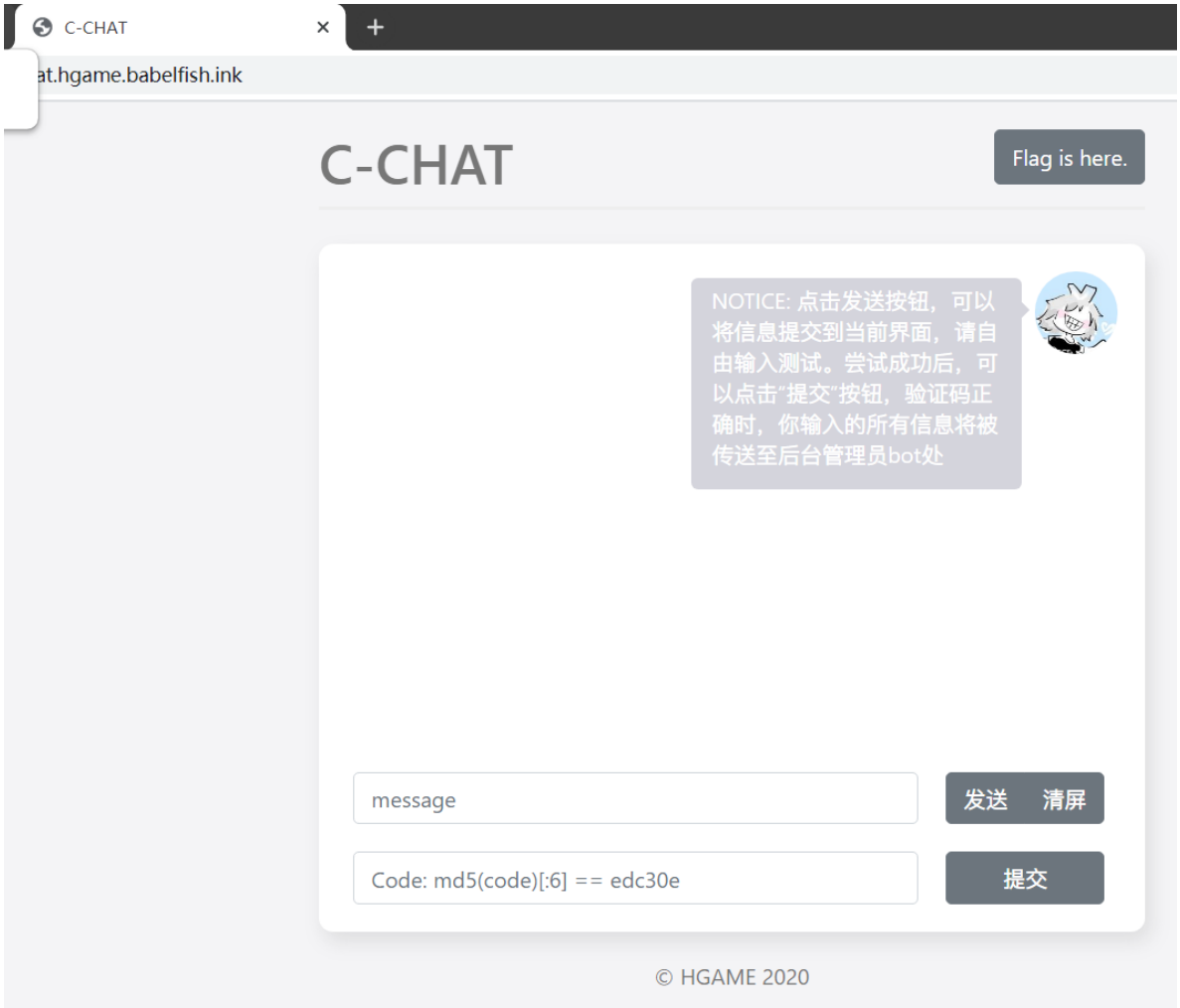
cosmos刚学了一些前端，他写了一个聊天室，并设置了flag的权限，只有管理员才能拿到flag...

Challenge Address http://c-chat.hgame.babelfish.ink

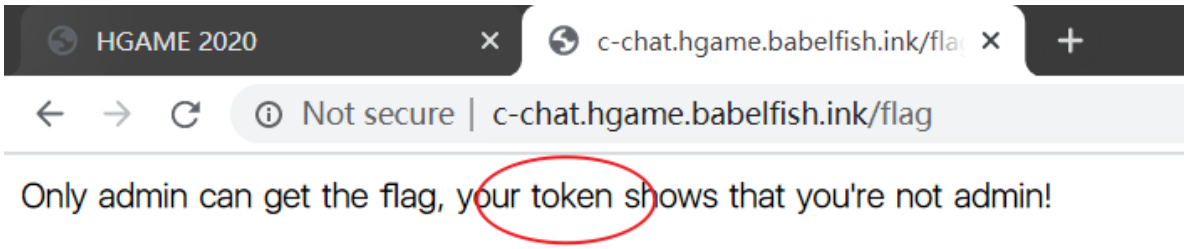
Base Score 150

Now Score 150

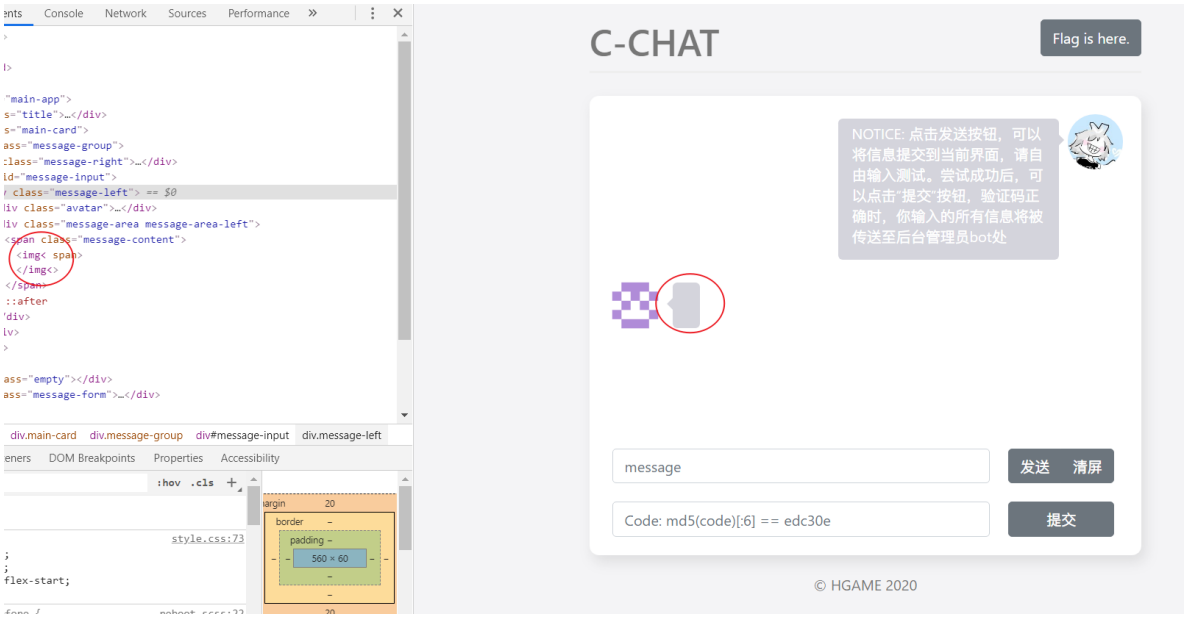
User solved 30



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`



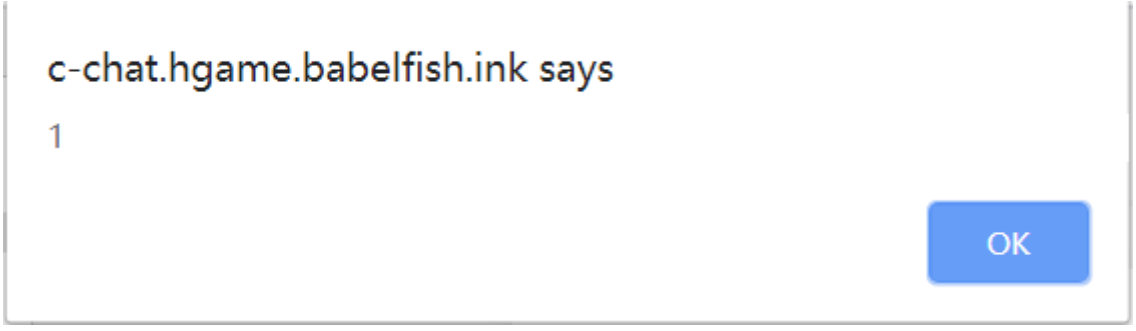
刚开始的时候看到信息是`空的`，我还以为`<`和以后的字符串全部被过滤了，因此尝试了好久，其实，如果打开`F12`看一看就完全不会卡住了，可以看到，虽然我只输入了一个`未闭合`的尖括号，浏览器自动帮我闭合，这就是利用了浏览器的`容错性`，而这点，还是🔗提醒我的。

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。

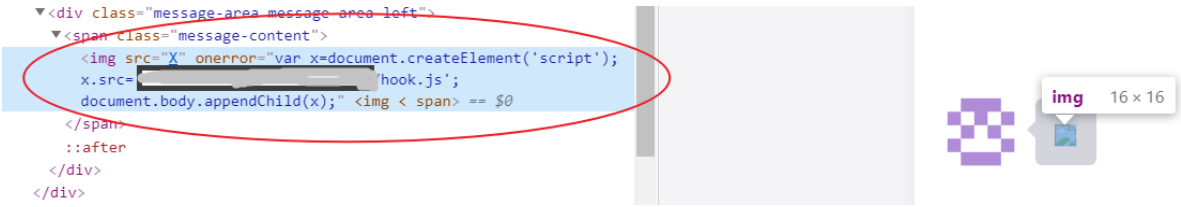
```
<img onerror=&#97;&#108;&#101;&#114;&#116;&#40;1&#41; <img src=x
```

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！

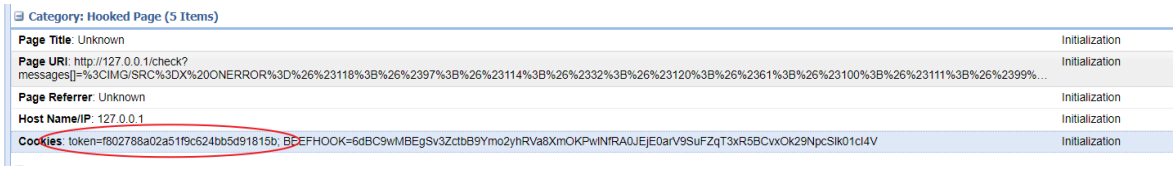


接着就可以利用`xss平台`盗取管理员的`cookie`了，在这里我搭平台至少用了一天时间，首先是找在线的`xss平台`，比如`xss.pt`，可是等了半小时迟迟收不到`cookie`，气死我了，换了两个平台都不行，只好亲自动手，丰衣足食，最后找到了`docker`里的`beef`，成功搭起了我的`xss平台`。

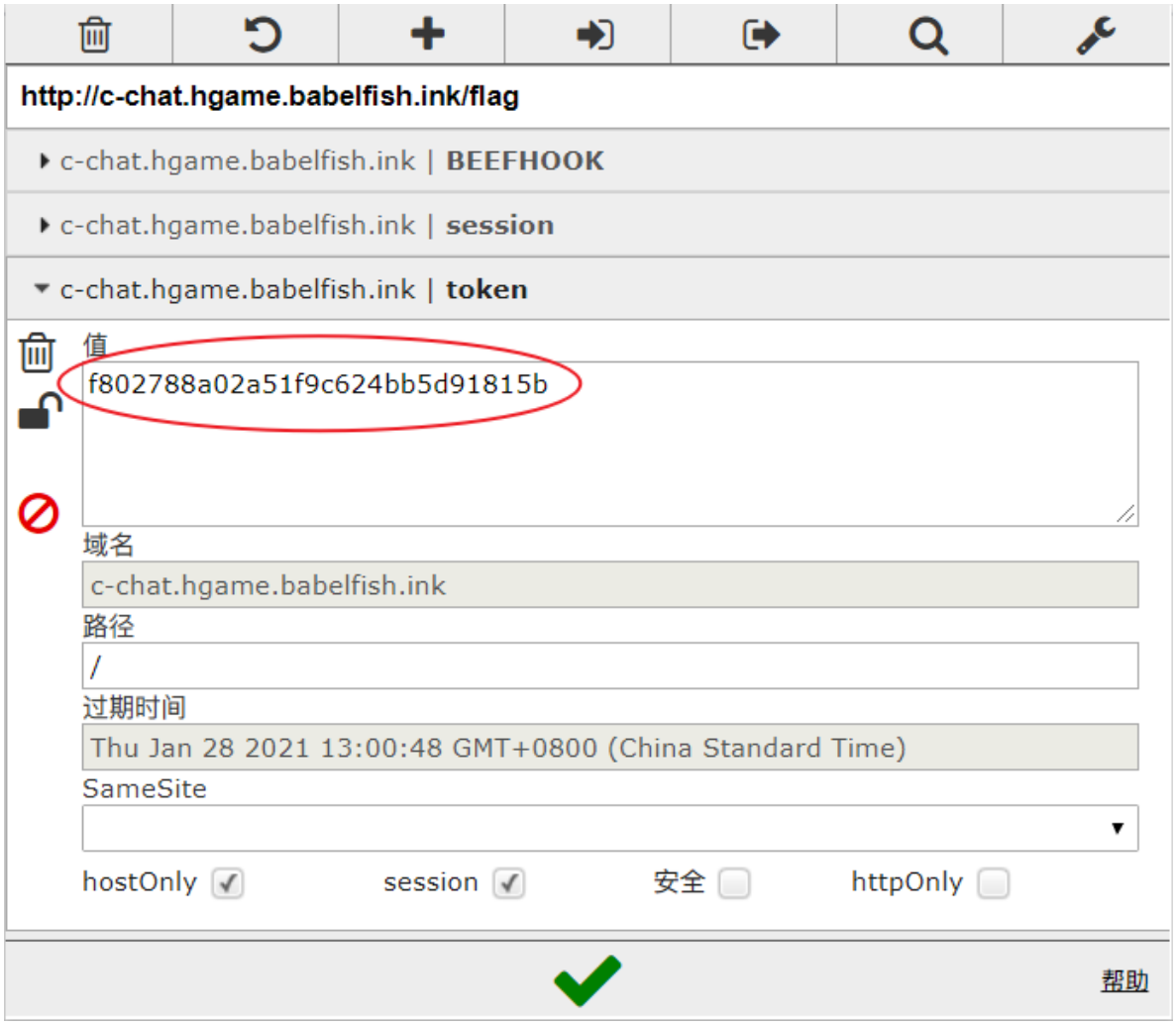
然后就可以构造`xss`语句，思路就是当图片加载失败的时候，让浏览器访问`xss平台`上的钓鱼脚本`hook.js`，然后`xss平台`立刻就会相应，返回管理员的相关信息，其中就包含了我想要的`cookie`。于是开始构造！



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`。



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`。



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。

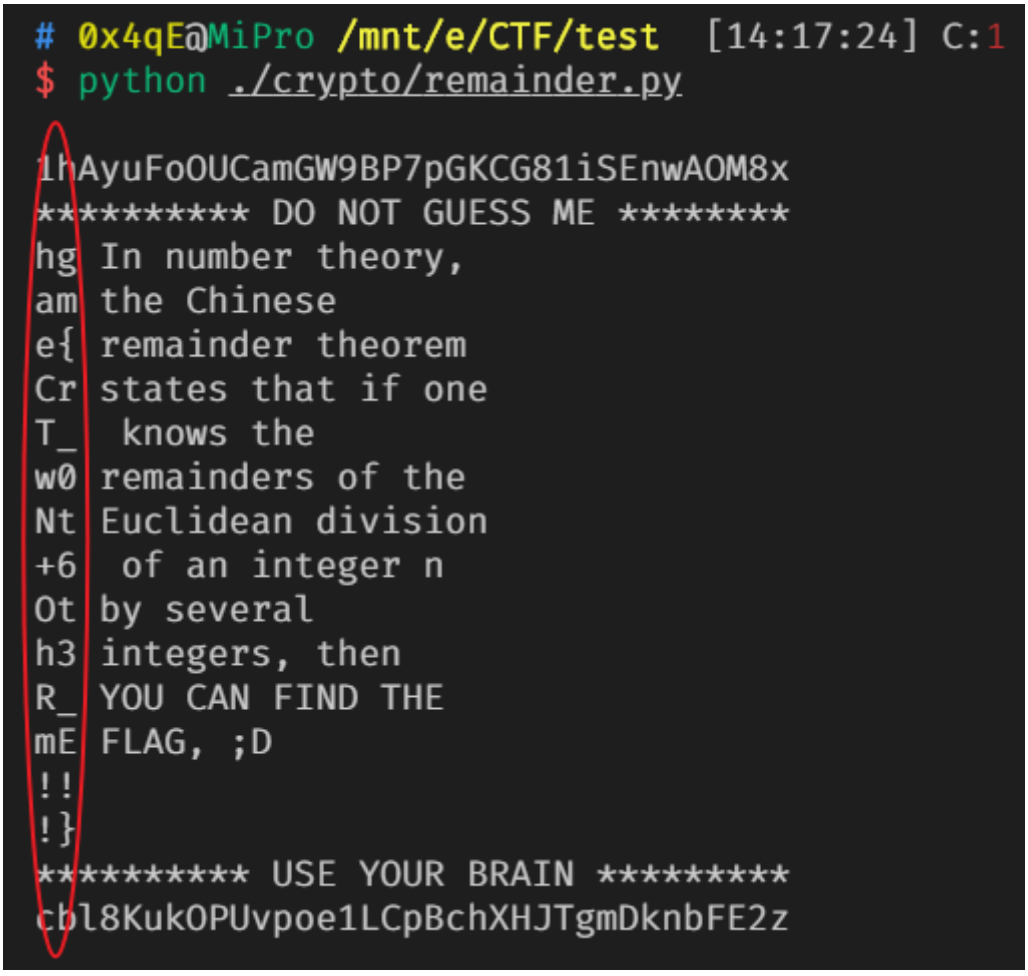
```
# 0x4qE@MiPro /mnt/e/CTF/test  [17:20:59] C:1
$ python3 ./crypto/vertification_code.py
[+] Opening connection to 47.98.192.231 on port 25678: Done
qcvw
[*] Switching to interactive mode

> Ok, you find me.
Here is the flag: hgame{It3Rt0O|S+I5_u$3fu1~Fo2_6rUtE-f0Rc3}
Bye~
[*] Got EOF while reading in interactive
$
```

Remainder

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 =
9459829630571337665254041163194943430139623511167337273827675465418826701080552254206800445313767859889133540817027760138194458427933936205657926230842754467168861492383979452267137855927678473475872721307040383863228628047345008676228670686392296872320283039826622053388512917550214253360055929238800591456 1
# q =
1500882164174049638936792428889929987932579033439947926979391217380294777904545833496600101388493792476973514786401036309378542808470513073408894727406158296404360452232777491992630316999043165374635001806841520490997788796152678742544032835808854339130676283497122770901196468323977265095016407164510827505883
# r =
1458977360966890961517047403276651763086250974841167137800503111987756074658620664068308517102618689138358663351071462429793599649451252144208211466709197411182544020969441394839887454504809897065241916693712082102729075639365169904732466153750226307082134867258098193600334704682931009266167297422777297057 27

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

#
7843078601165052122456192481484361429480697498859959105891552039751852629642279108969210748853415758985661122997806865997097637497165890998729975971953351935823218072148071963560251552594267898889672712888480363825722784817629817289615546381326420698250579761306721518284955935633601563454318180629635555254 3
#
4957635642347422218820518730688416762074647967759012121379109390897729580347620351000106018095919097276817541142411523867555147201992480220531431019627681572335103200586388519695931348304970651875582413052411224818844160945410884130575771617919149619341762325633301313732947264125576866033934018462843559419 3
#
4813107796264949783318929263786144276756214744704013441107888448551384055318818595438333023619025338893778553065827976862021306224405315161496289362894634359564251387076687781053448053673720030269953939681054542002105422520468342852282035035647088357446384914642215024430414761819561379639901049212538332292 2
```

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 mod 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。



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：

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

from Crypto.Util.number import *
import gmpy2

n = 3
```

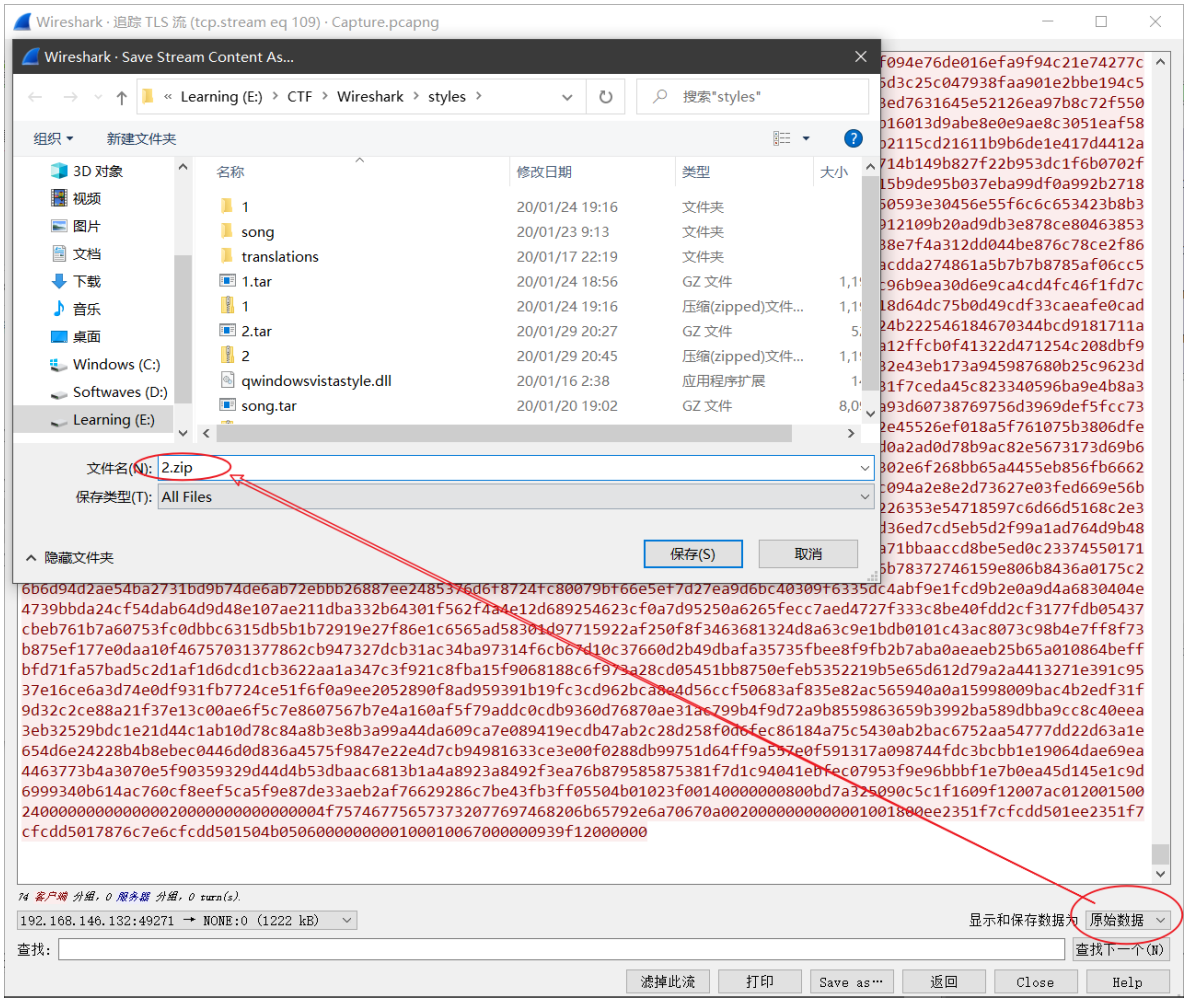
```
a =
[784307860116505212245619248148436142948069749885995910589155203975185262964227910896921074885341575898566112299780686599709763749716589099872997597195335193582321807214807196356025155259426789888967271288848036382572278481762981728961554638132642069825057976130672151828495593563360156345431818062963555525543,

495763564234742221882051873068841676207464796775901212137910939089772958034762035100010601809591909172768175411424115238675551472019924802205314310196276815723351032005863885196959313483049706518755824130524112248188441609454108841305757716179191496193417623256333013137329472641255768660339340184628435594195,

481310779626494978331892926378614427675621474470401344110788844855138405531881859543833302361902533889377855306582797686202130622440531516149628936289463435956425138707668778105344805367372003026995393968105454200210542252046834285228203503564708835744638491464221502443041476181956137963990104921253833229225, ]     # 余数
m =
[945982963057133766525404116319494343013962351116733727382767546541882670108055225420680044531376785988913354081702776013819445842793393620565792623084275446716886149238397945226713785592767847347587272130704038386322862804734500867622867068639229687232028303982662205338851291755021425336005592923880059145615,

150088216417404963893679242888992998793257903343994792697939121738029477790454833496600101388493792476973514786401036309378542808470513073408894727406158296404360452232777491992630316999043165374635001806841520490997788796152678742544032835808854339130676283497122770901196468323977265095016407164510827505883,

145897736096689096151704740327665176308625097484116713780050311198775607465862066406830851710261868913835866335107146242979359964945125214420821146670919741118254402096944139483988745450480989706524191669371208210272907563936516990473246615375022630708213486725809819360033470468293100926616729742277729705727, ]     # 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。

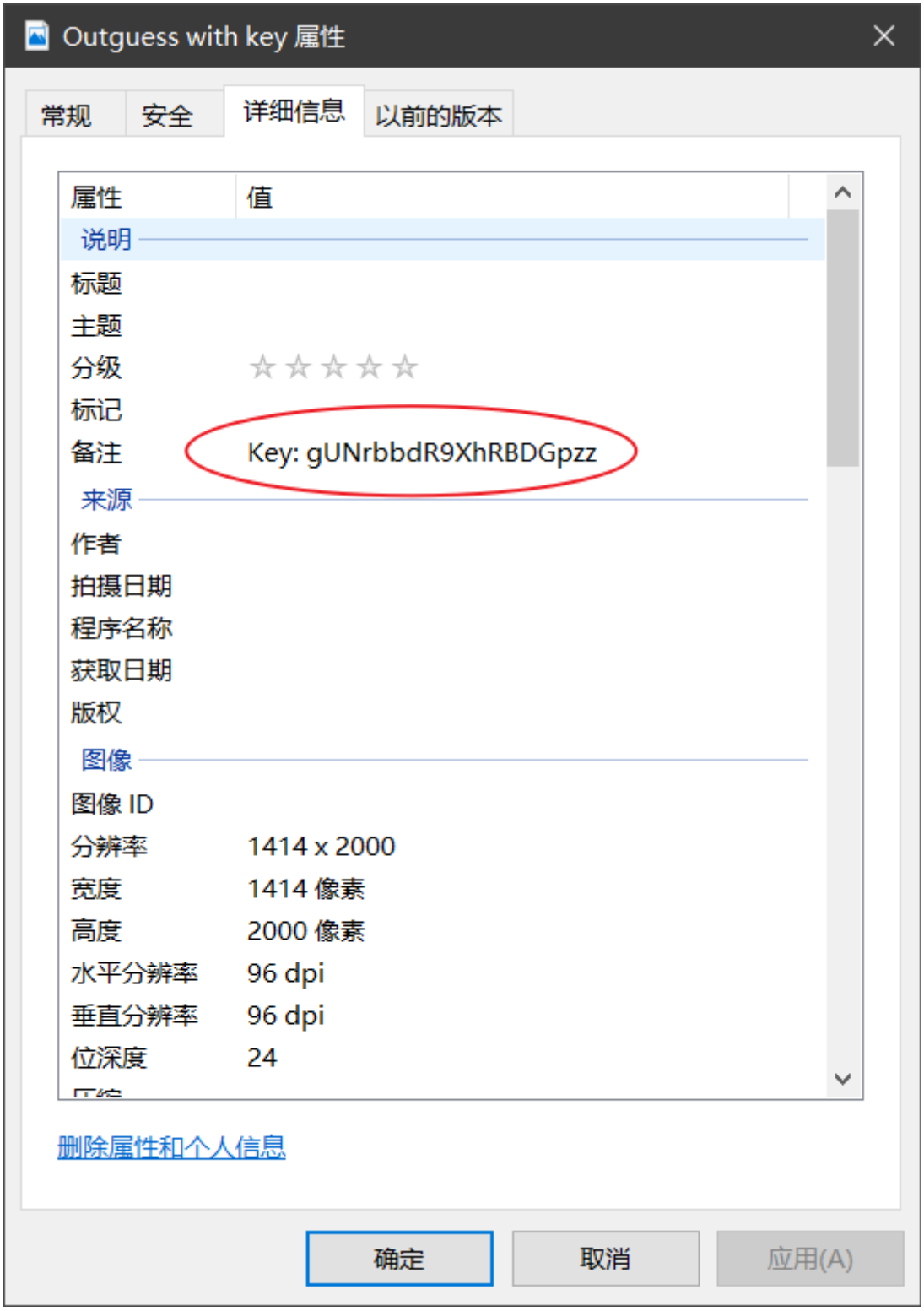
Challenge Address http://oss-east.zhouweitong.site/hgame2020/week2/MeisLunch_6gbsgtLMHGFmS82L5ilRN7JjOzY9Pw3h.zip

Base Score 200

Now Score 200

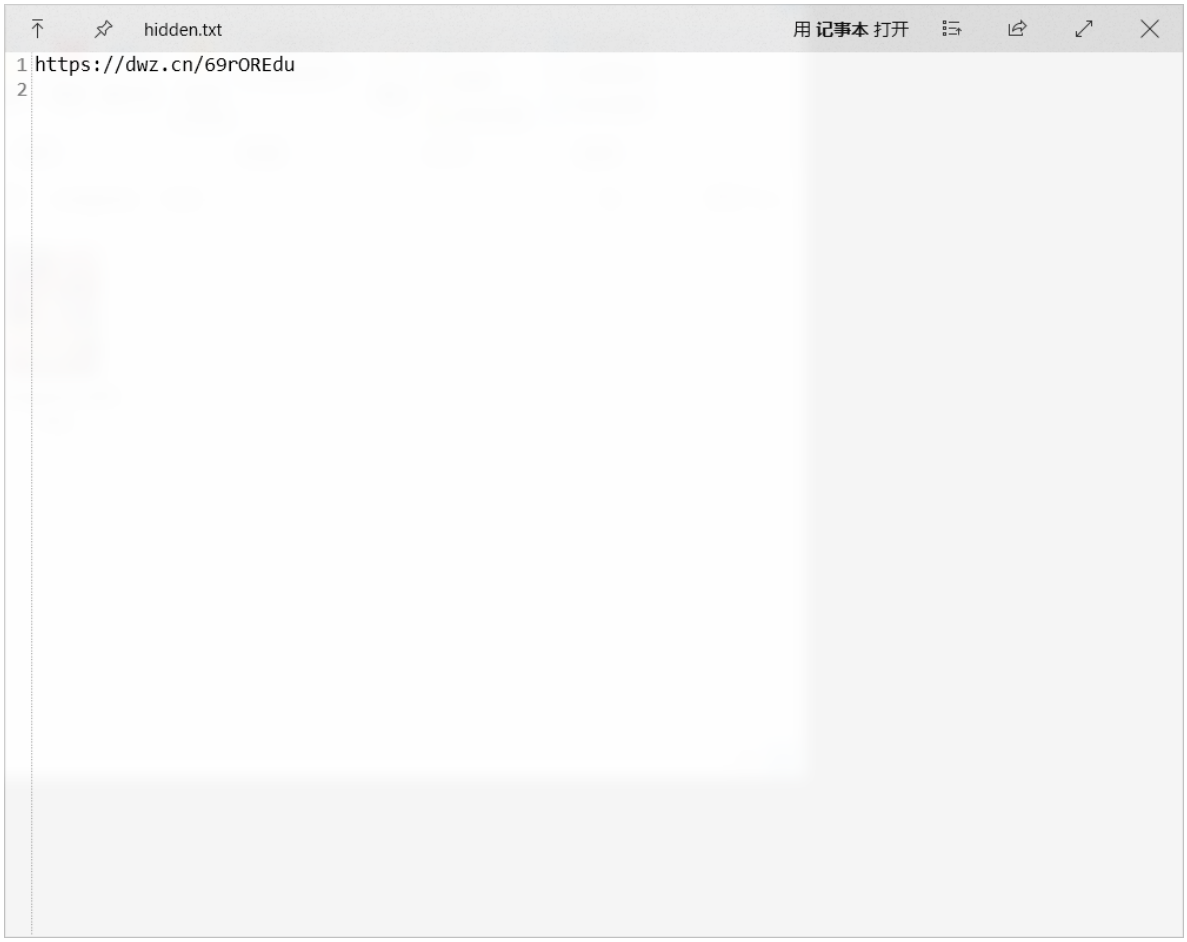
User solved 46

于是我们打开图片 属性。

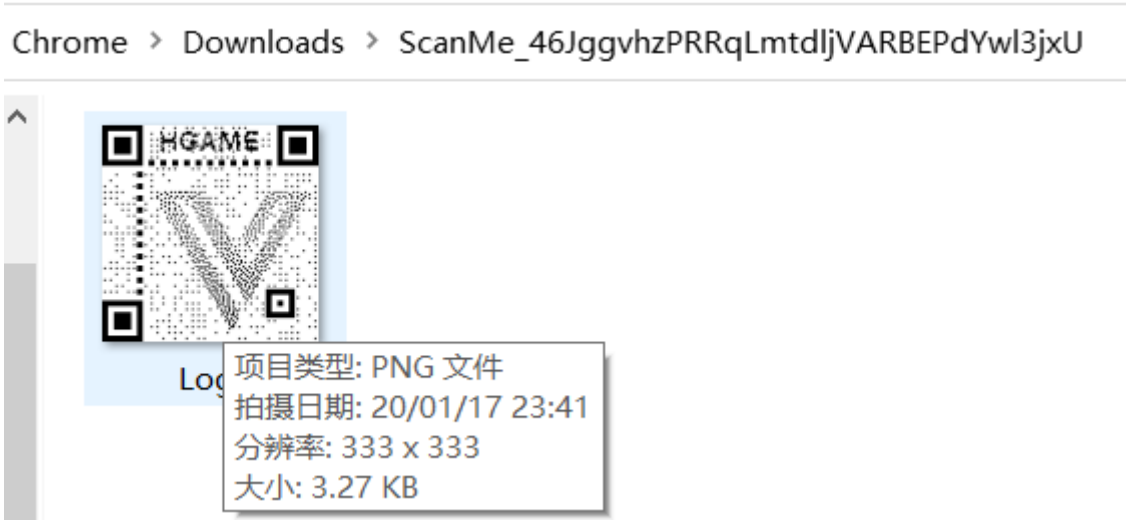



```
# 0x4qE@MiPro /mnt/e/ctf/outguess/test master x [20:59:34]
$ outguess -k 'gUNrbbdR9XhRBDGpzz' -r 'outguess with key.jpg' hidden.txt
Reading outguess with key.jpg....
Extracting usable bits:   1161827 bits
Steg retrieve: seed: 3, len: 24
```

得到一个`hidden.jpg`



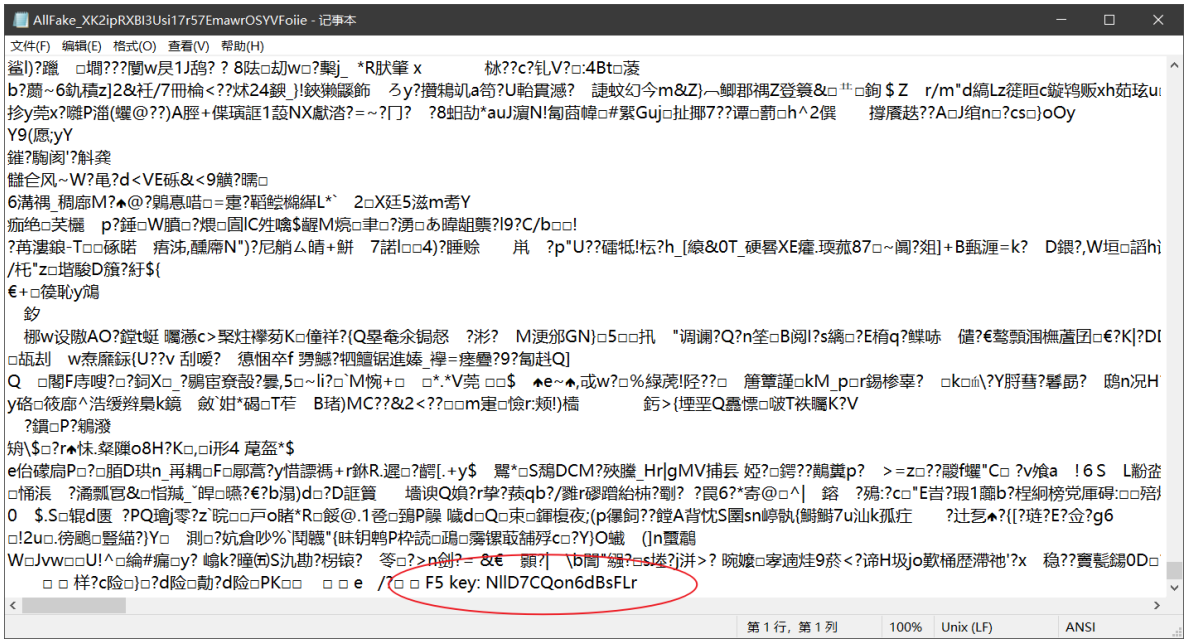
访问之后下载来一个压缩包，解压后打开，是一个二维码，扫一下就可以得到`flag`了。



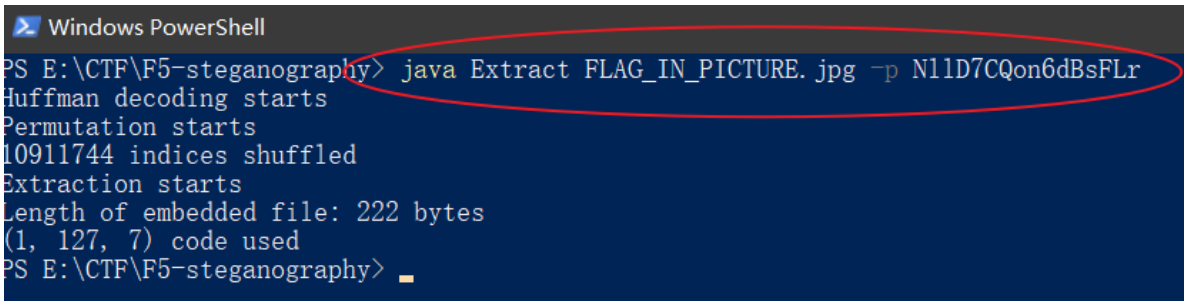
所见即为假

下载，解压，打开后是一张图片

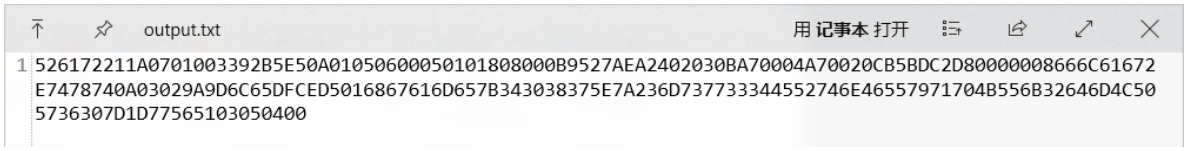
首先试试`binwalk`，`Stegslove`看看有没有`LSB`，都不行，在即将放弃的试着用`记事本`打开压缩包，终于找到了突破口！



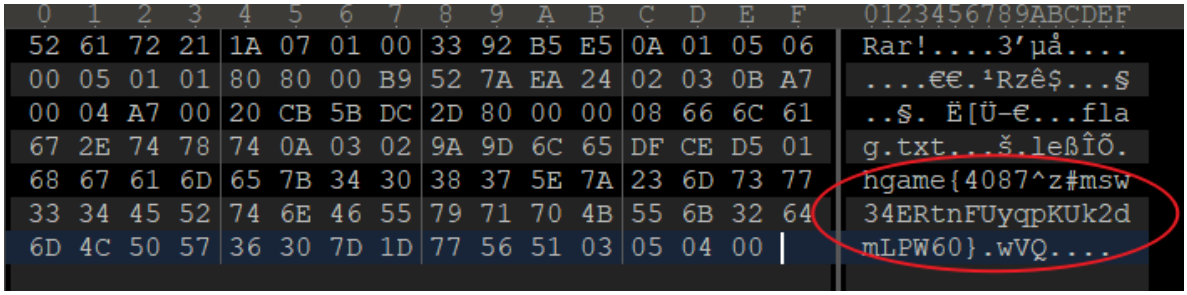
于是搜索`F5+隐写`，找到了工具`F5-steganography`。



得到一个`output.txt`，打开看看



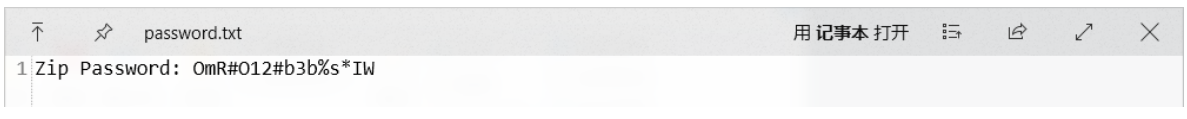
观察发现可能是`16进制`文件，于是在`010 Editor`里输入这一串`16进制`数字。



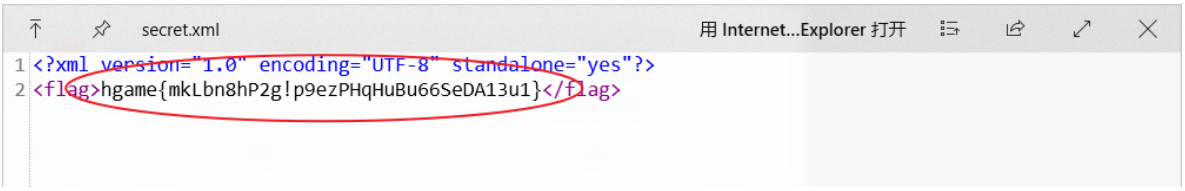
成功！

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`！



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/

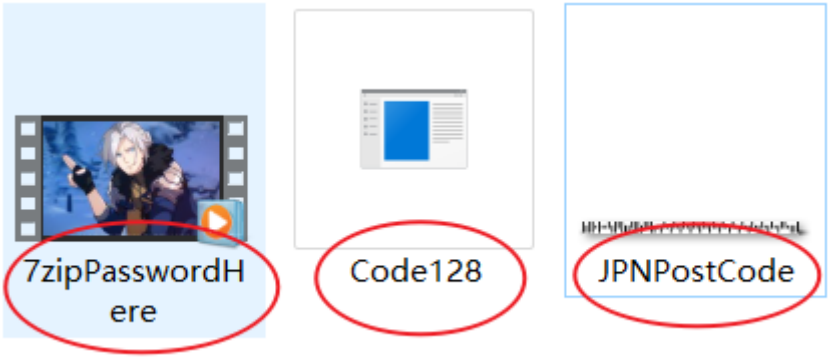
【参考资料2】：https://sourceforge.net/projects/virtualdubffmpeginputplugin/

Challenge Address http://oss-east.zhouweitong.site/hgame2020/week2/PlayWithCode_lVqVlOB8Sph7Zfelk8pFzHb7ay58nYPe.zip

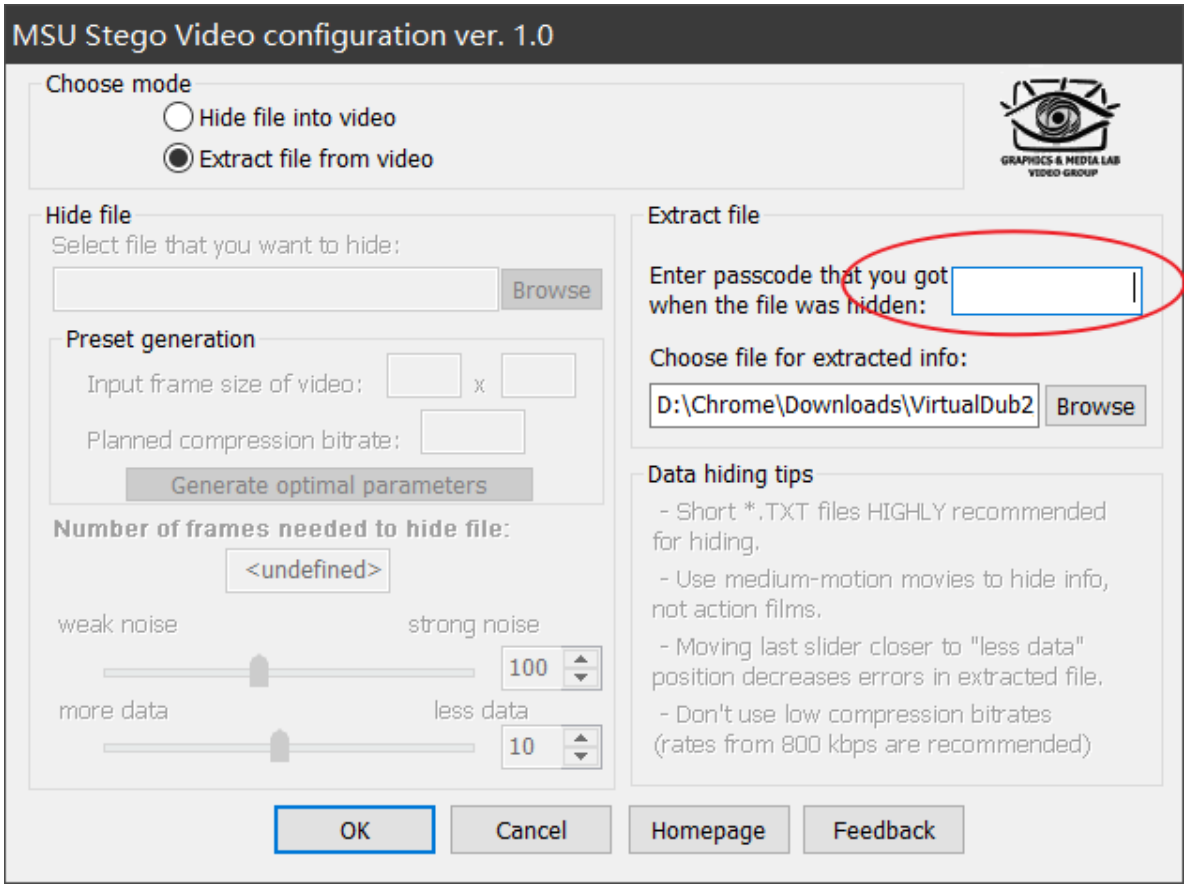
Base Score 250

Now Score 250

User solved 23



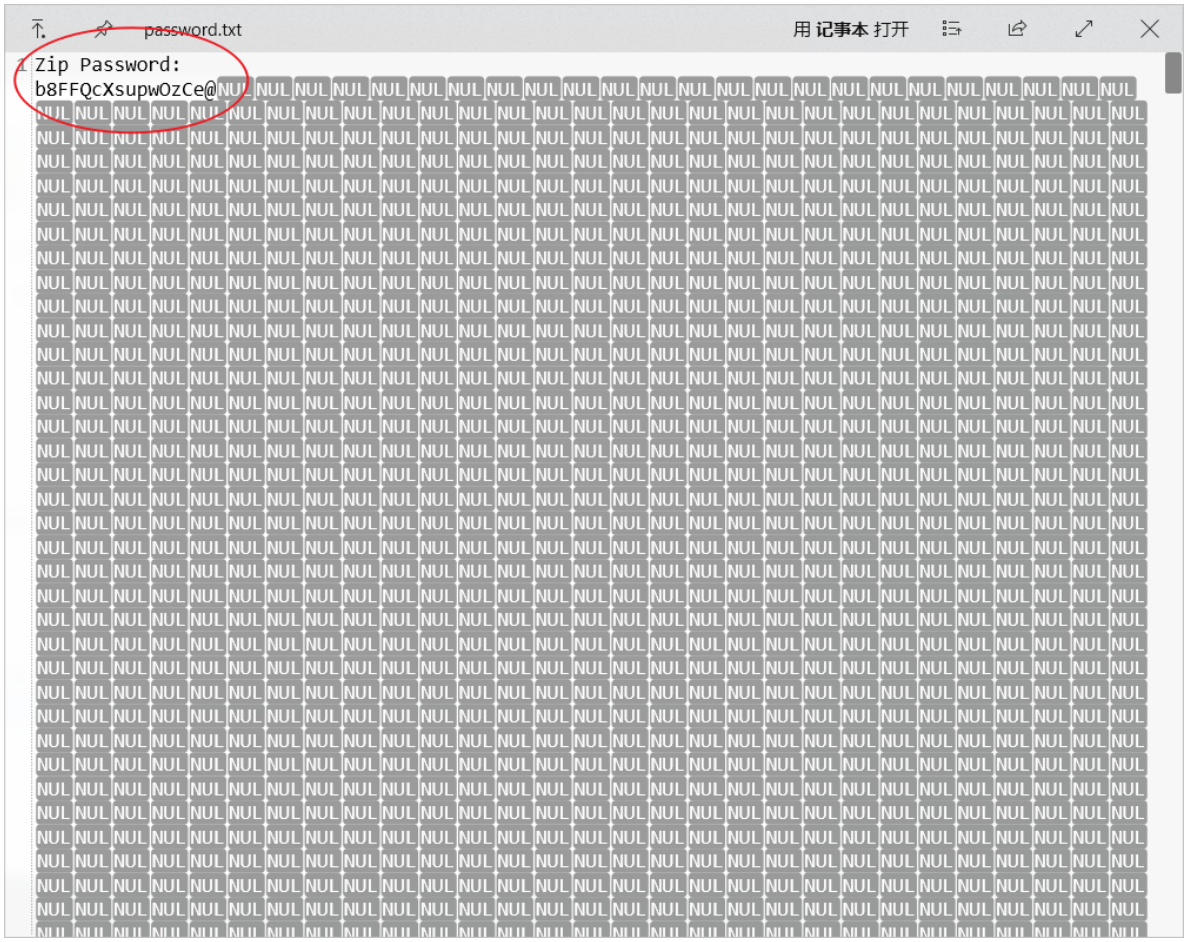
可以看到都需要解码，我首先看到的是`7zipPasswordHere`，于是想办法看看`.mp4`里藏着什么信息。经过学长的提示，下载了`MSU Stego Video`这个过滤器，在提取信息的时候发现需要输入密码。



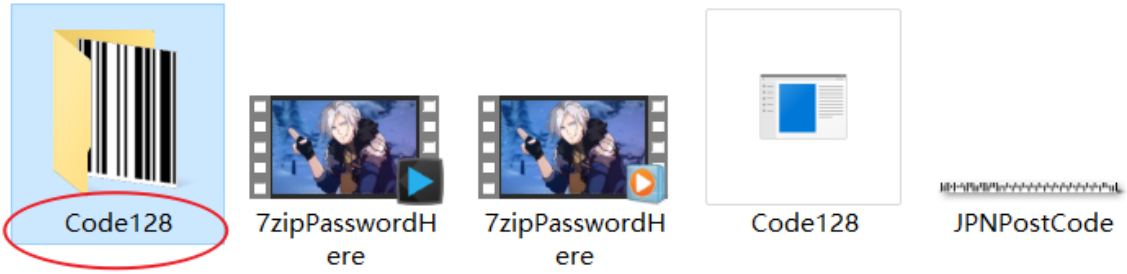
那可能就需要去解那个`JPNPostCode`了，`Google`一下猜测这应该是一个`日本`的`邮政条形码`。在网上找解码器是行不通的，我换了好几个在线网站都不能解出来，最终找到了在线条码生成器，手动去凑`条码`的图形，终于拿到了密码。

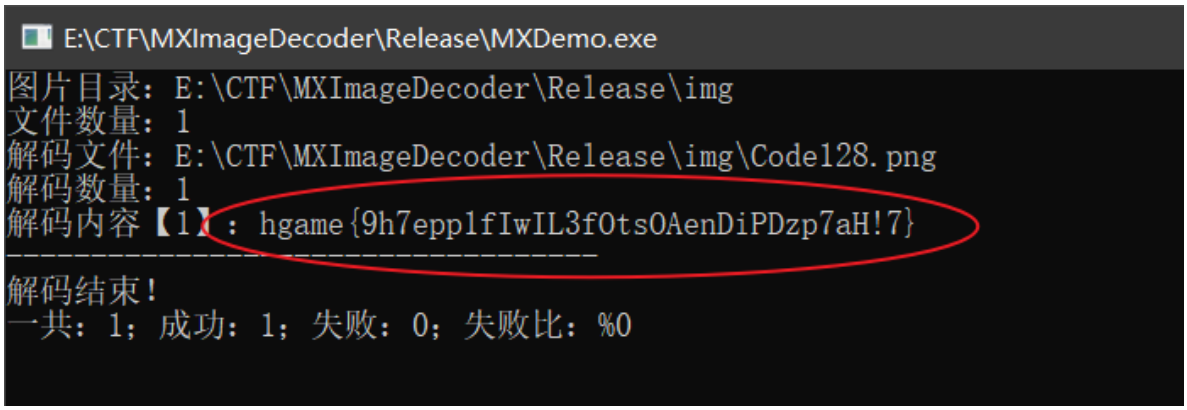
Code - 条形码生成器

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。





终于`ak`了`Misc`，好开心!