ISCC2019比赛部分writeup

原创

守卫者 于 2019-05-28 15:59:11 发布 3416 收藏 5
分类专栏： CTF
守卫者安全（www.zhaosimeng.cn）
本文链接：https://blog.csdn.net/weixin_42721957/article/details/89757844
版权

Misc

1.

附件地址

下载附件显示如下：

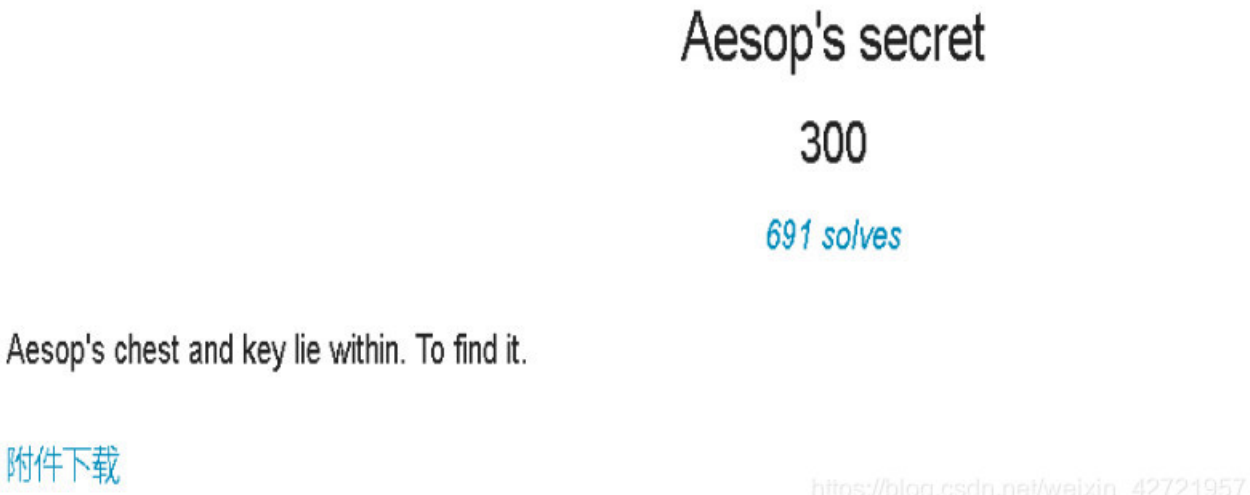
看了半天像是ascll码，但是有的已经超出范围了。

然后又觉得以0开头的是八进制的数，先进制转换。发现一段字符串看着像base64加密的

```
V2VsbCBkb25lIQoKIEZsYWc6IElTQ0N7TjBfMG5lX2Nhbl9zdDBwX3kwdX0K
```

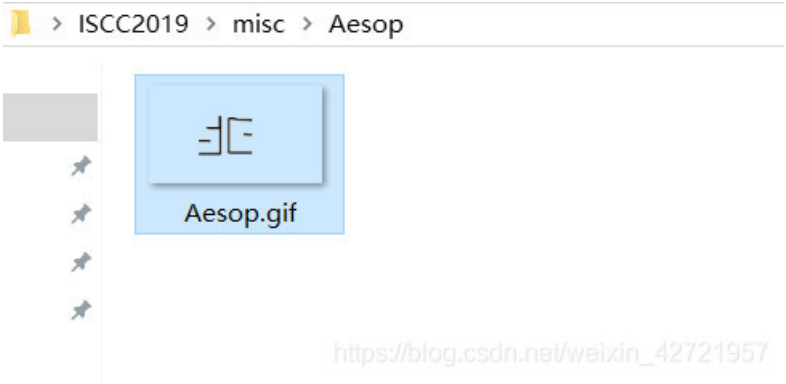
解密得到flag

2.Aesop's secret



附件地址

下载附件打开发现是一张gif图片，查看并没有什么卵用



然后记事本打开看看，发现最后有一堆字符串，比较可疑

结合到题目的线索，猜想应该是AES加密，而且密钥应该是ISCC

用在线解密网站进行解密，需要经过两次才能得到flag，尝试一次就换思路的小伙伴不要太郁闷

在线加密解密(采用Crypto-JS实现)

Feedback

加密/解密 | 散列/哈希 | BASE64 | 图片/BASE64转换

明文:

flag{DugUpADiamondADeepDarkMine}

加密算法:

- AES
- DES
- RC4
- Rabbit
- TripleDes

密码:

ISCC

加密 >

< 解密

密文:

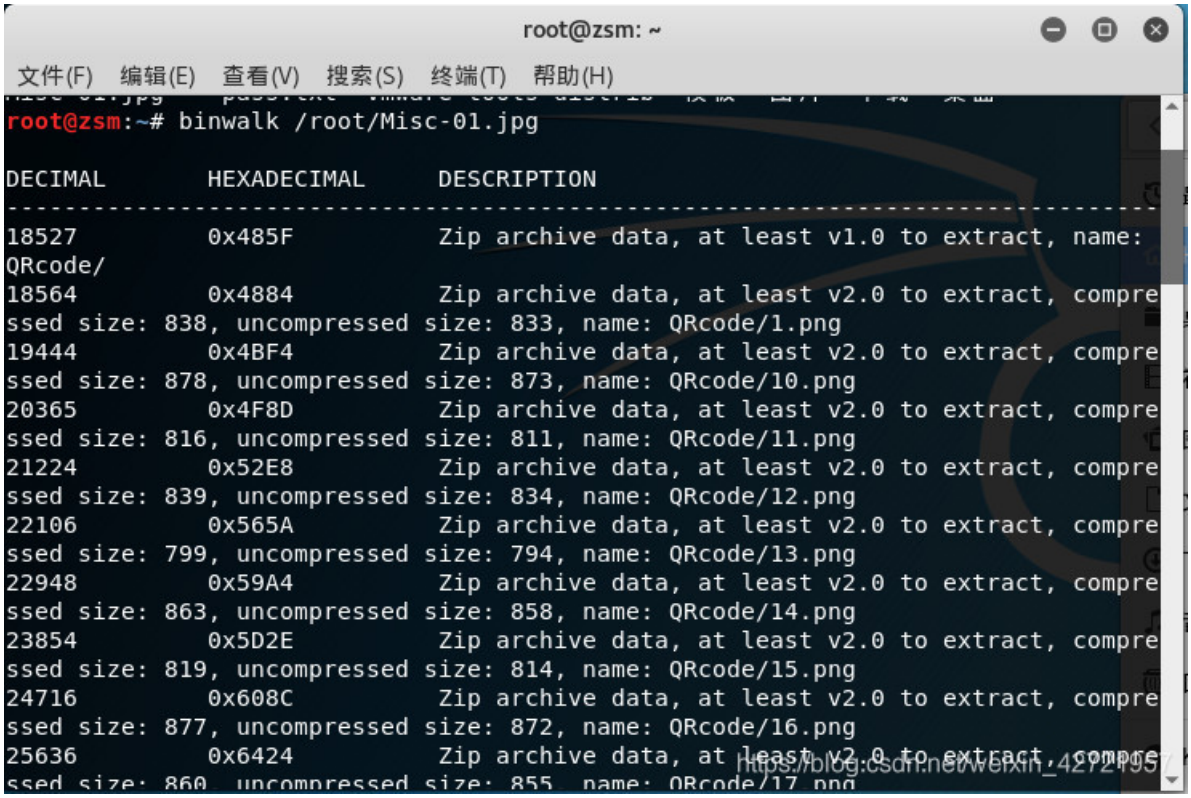
U2FsdGVkX18OvTUlZubDnmvk2lSAkb8Jt4Zv6UWpE7Xb43f8uzeFRU
KGMo6QaaNFHZriDDV0EQ/gt38Tw73tbQ==

3.最危险的地方就是最安全的地方

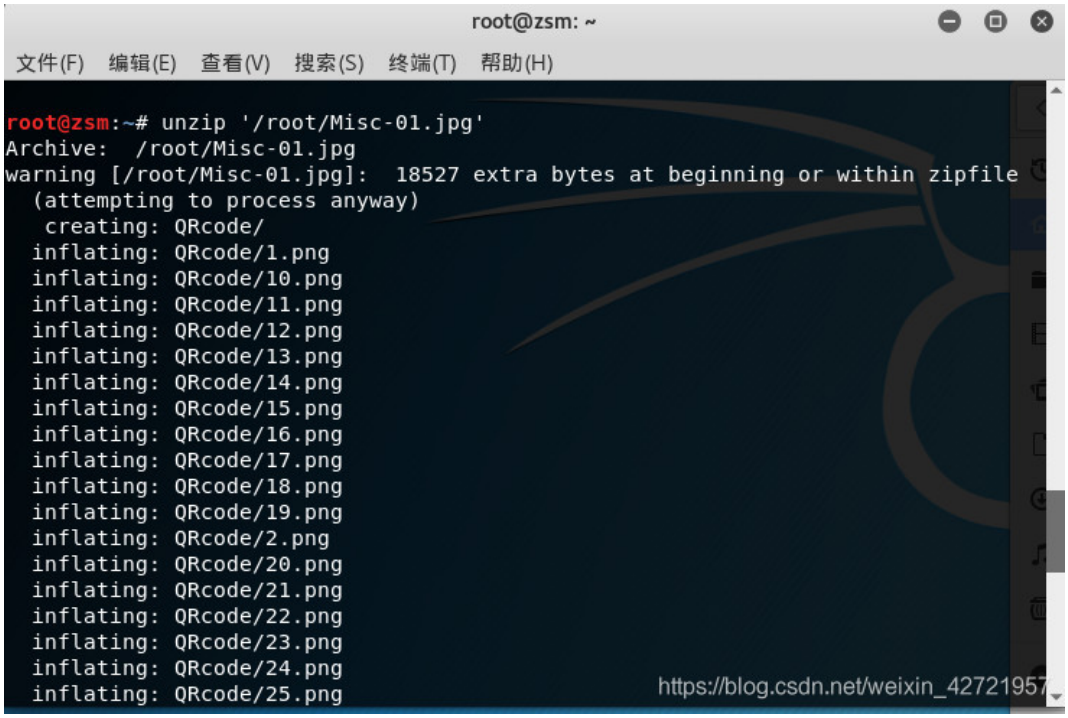
附件地址

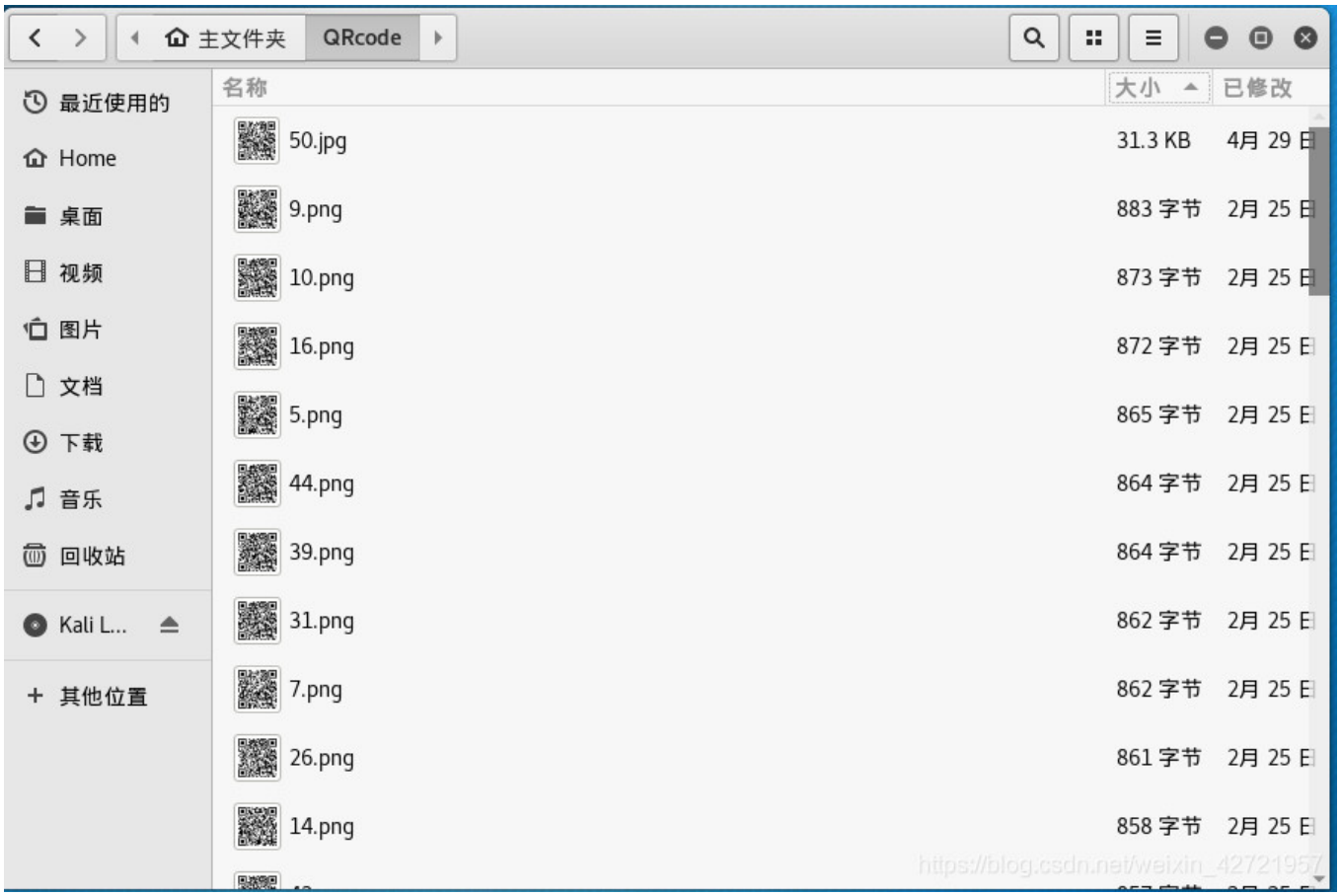
下载附件发现是一张图片，但是无法打开

用binwalk打开查看，似乎是一个压缩文件

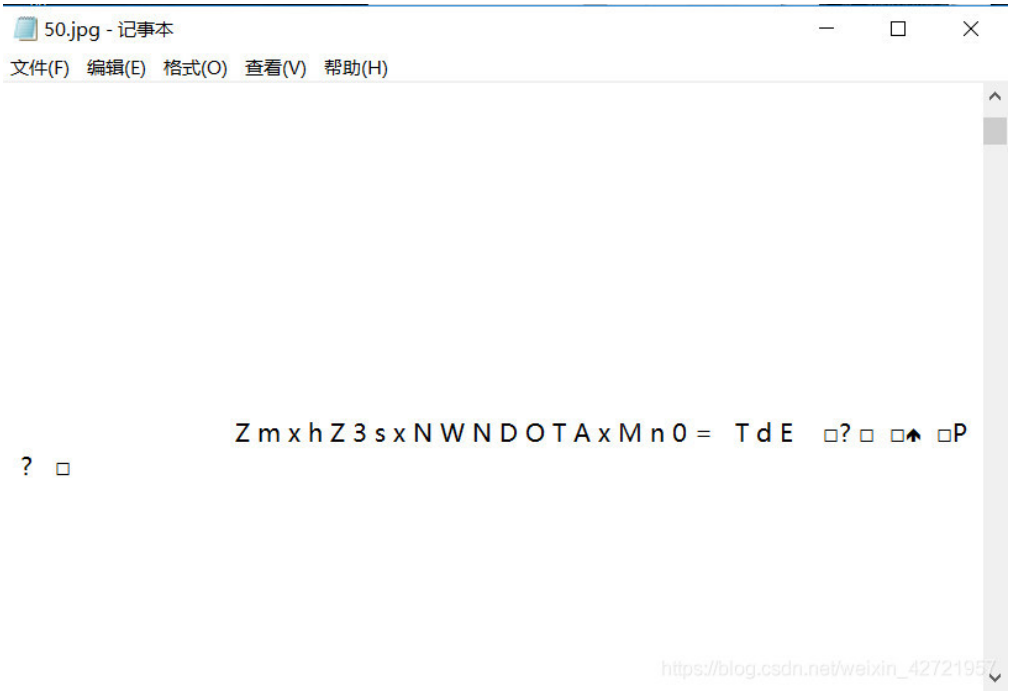


解压缩得到50个二维码图片





发现别的图片都特别小，只有50格外的大，应该藏有不一样的东西，用记事本打开



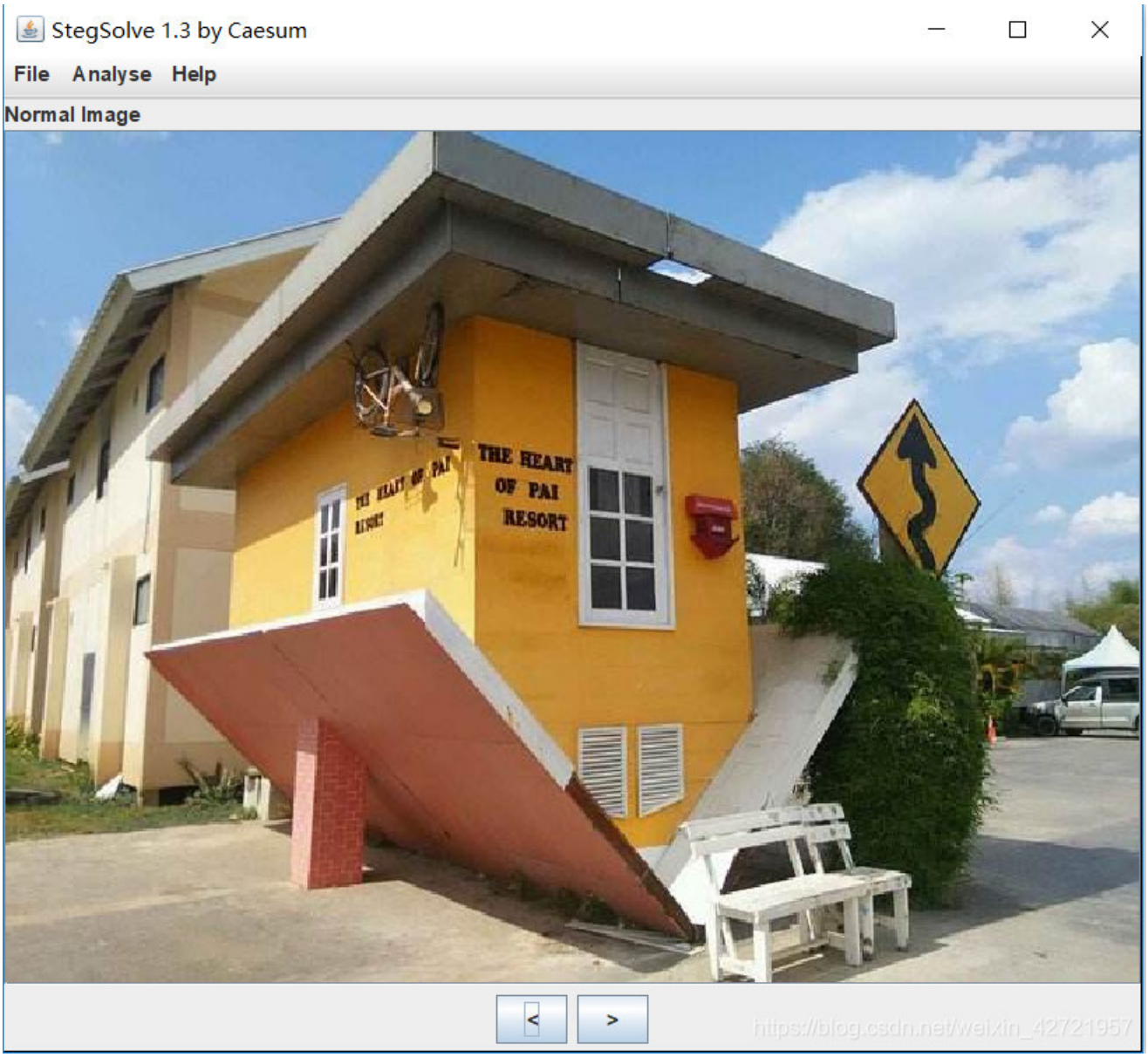
看到这一段字符串，就想到了base64，解密果断得到flag

4.倒立屋

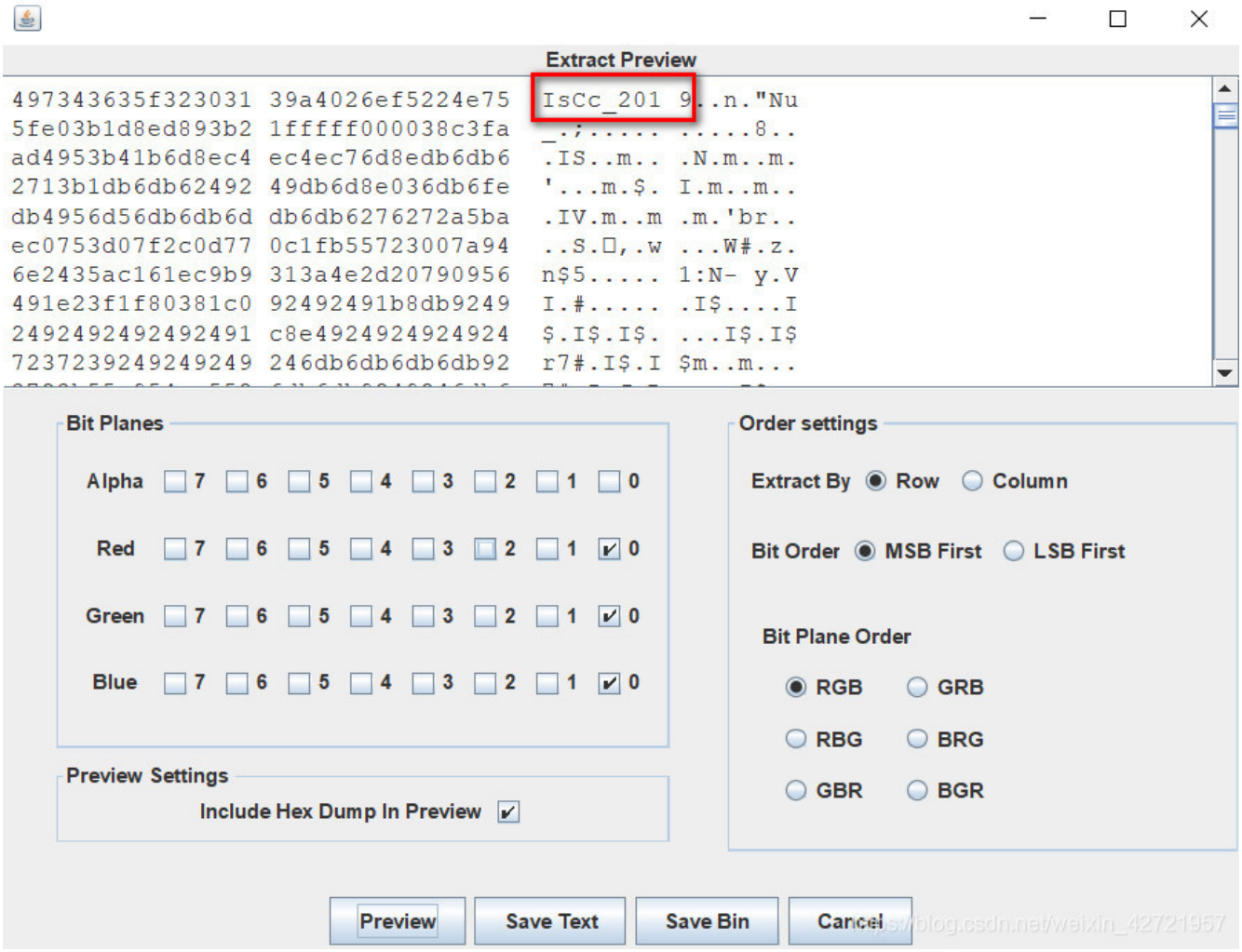
附件地址

附件是一个倒立的屋子的图片

用记事本和hex 编辑器打开没有发现什么异常，再用 StegSolve 打开看看



点 开Analyse的Data Extract，然后逐位试试，发现 RGB 都点到 0 的时候开头有一段奇怪的字符串。

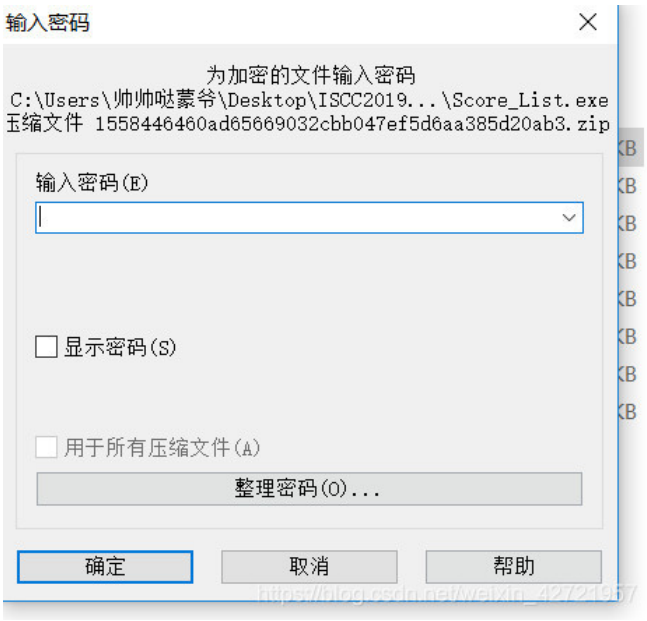


直接提交当然不对，因为题目是倒立屋，所以倒过来提交就是flag。

5.解密成绩单

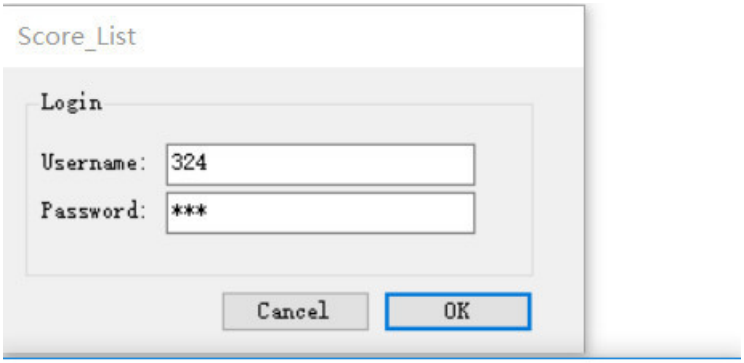
附件地址

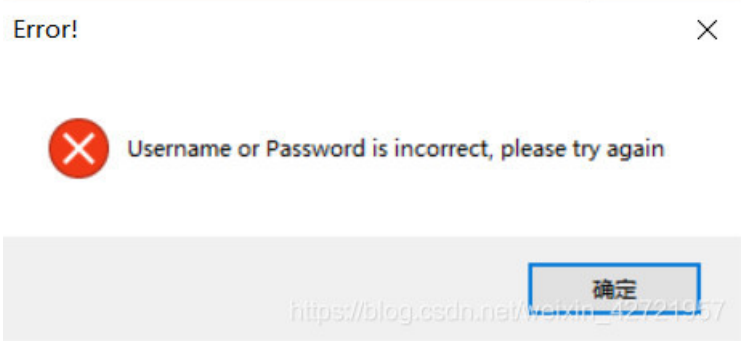
附件下载下来解压发现需要密码



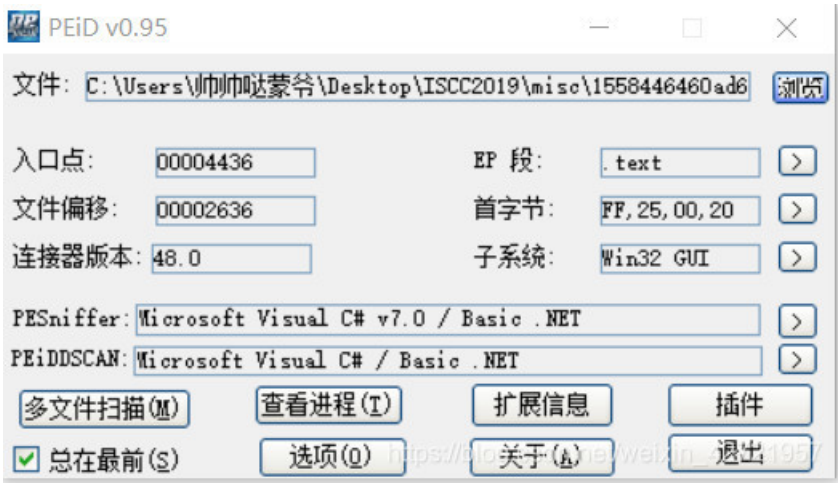
但是题目并没有给出过多的提示，所以这里就卡住了，队友说用360压缩可以不要密码，试了一下果然可以，还没搞懂这个原理

解压出来是一个可执行文件，随便输入一个尝试了一下

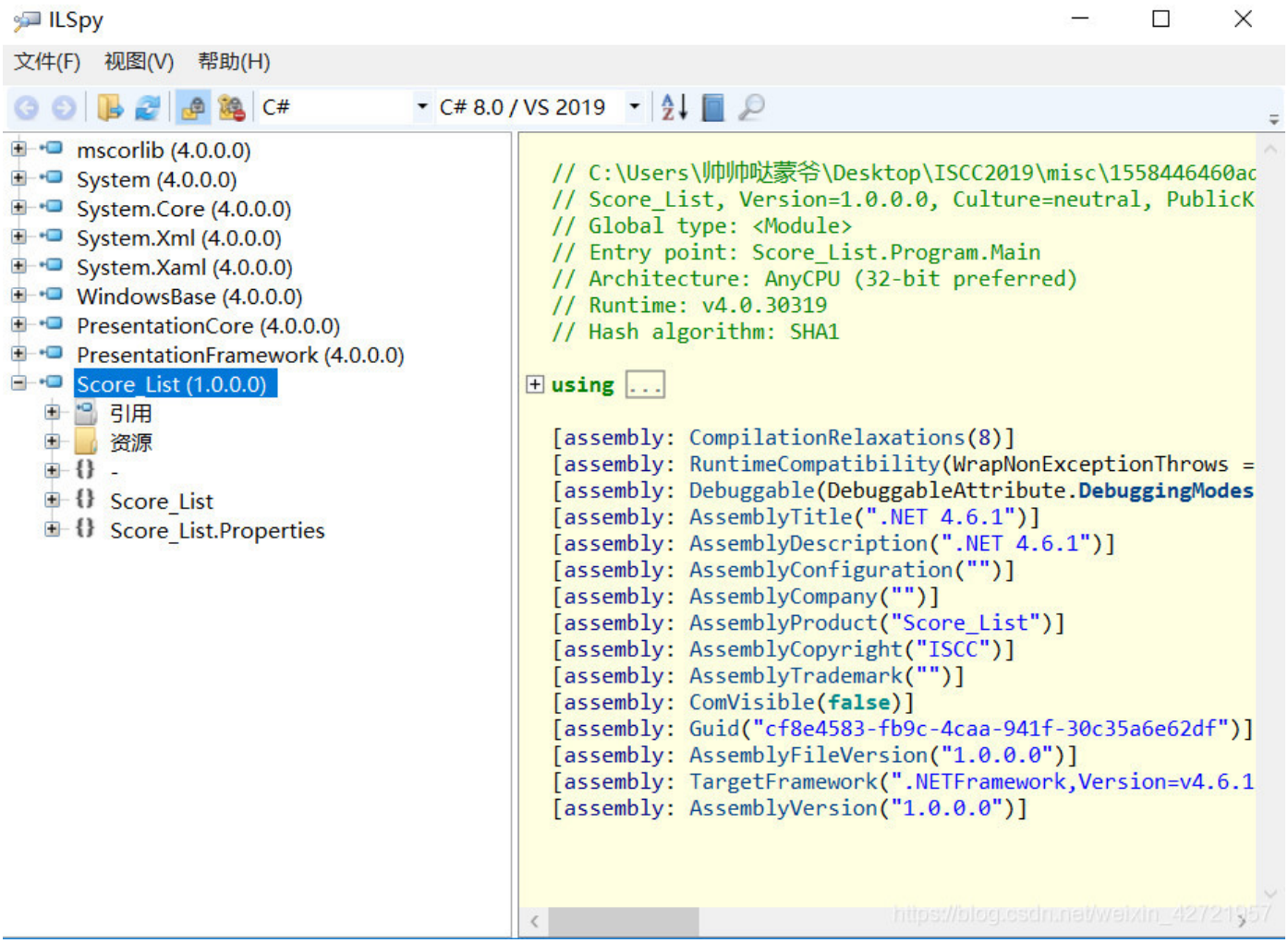




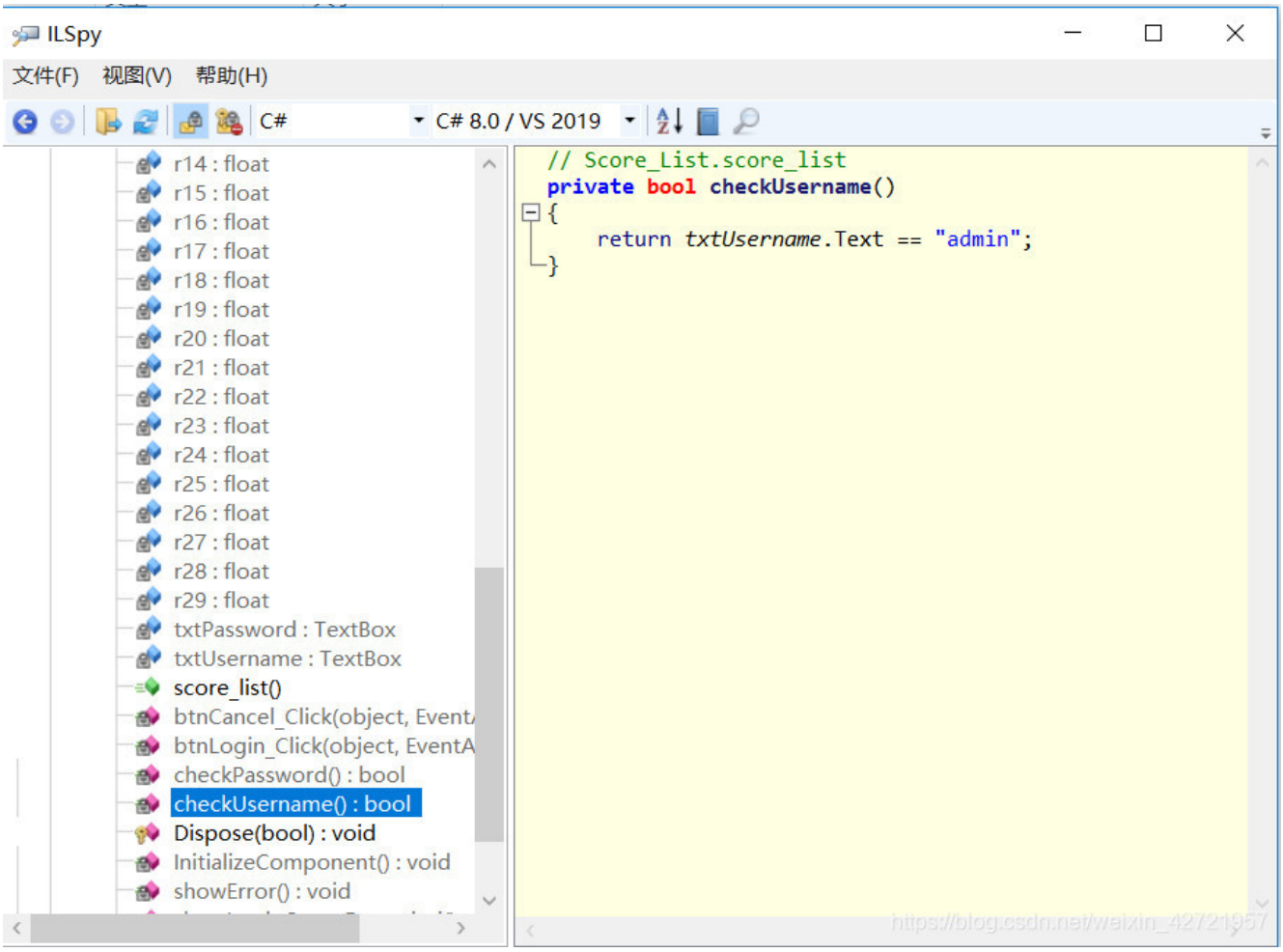
然后用PEID 看看

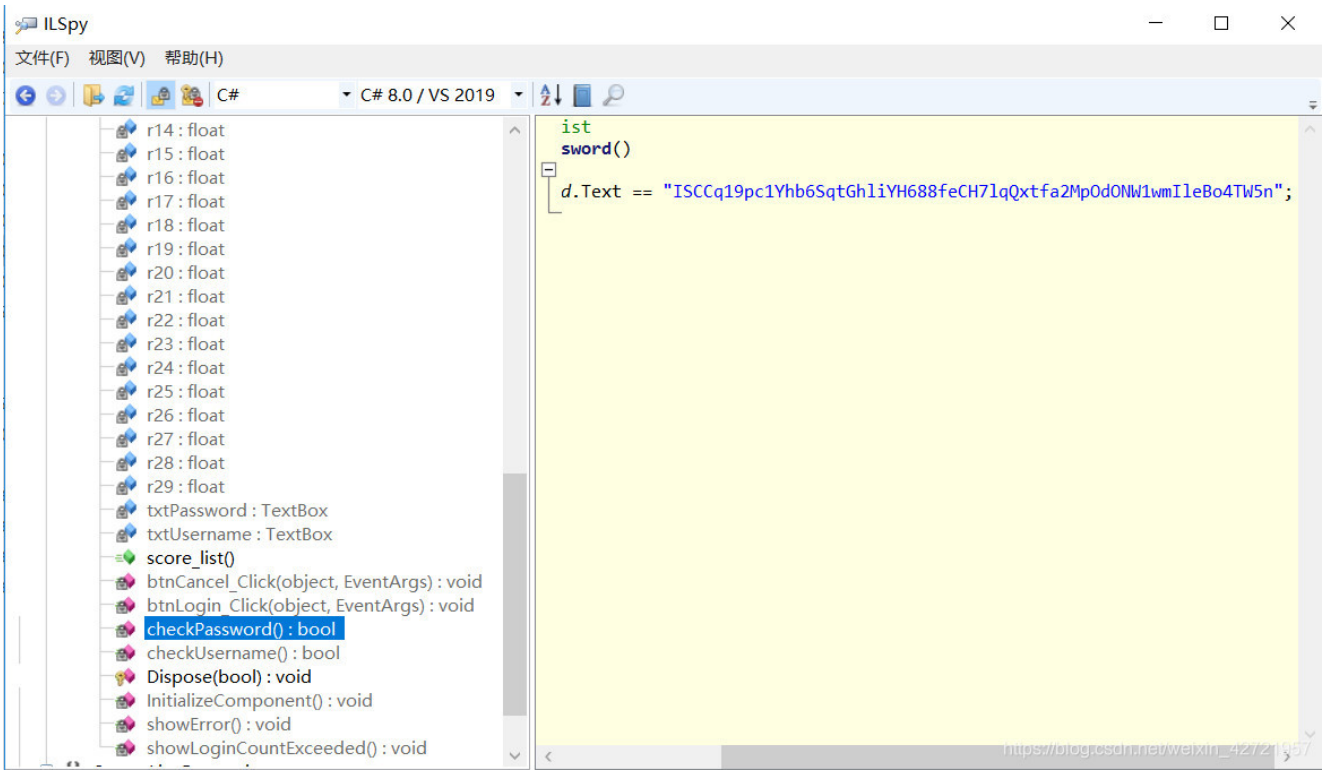


所以.Net 反编译一下看看　反编译工具

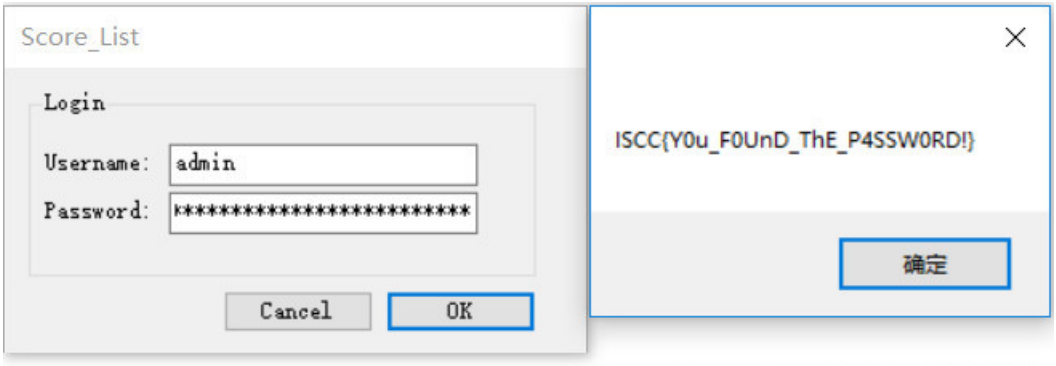


这里点开发现两个特殊的函数方法





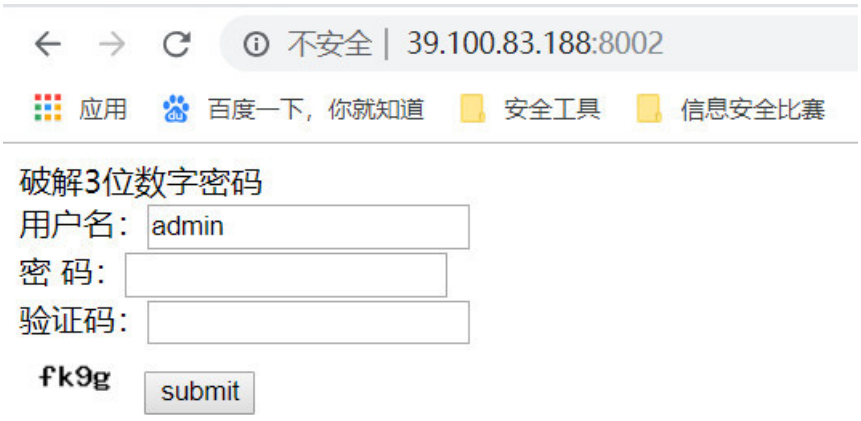
从这里我们就发现了用户名和密码，再次登录得到了flag



Web

1.

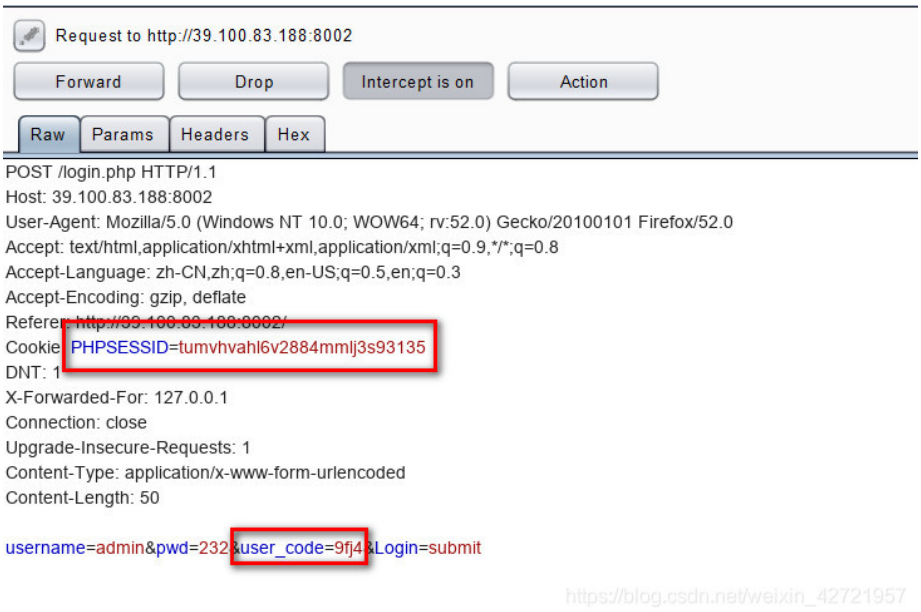
打开链接发现是一个登录界面。马上想到暴力破解。可是这里还有验证码，所以就很尴尬了



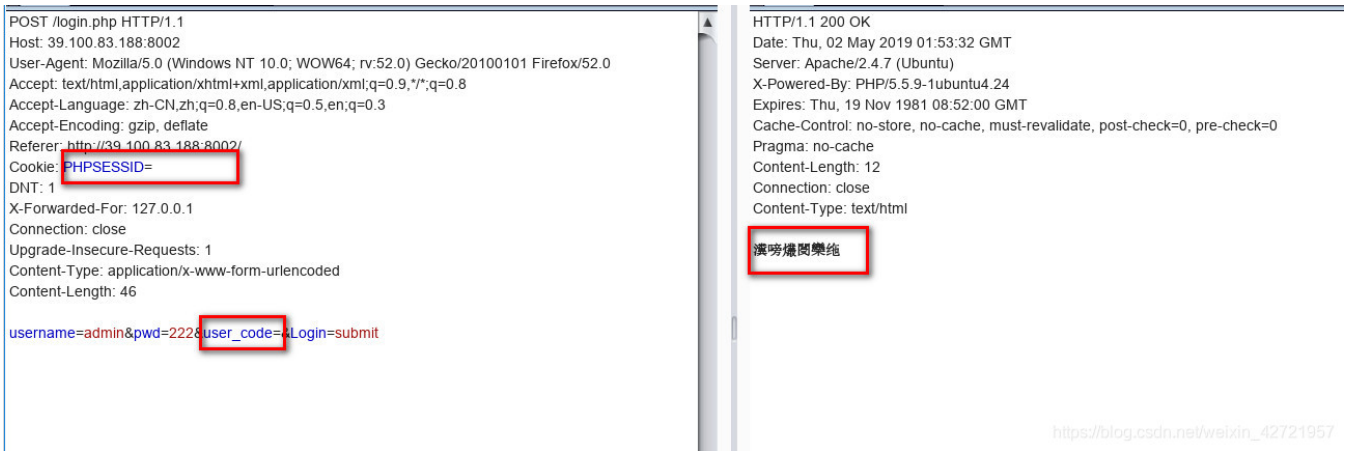
这里我找到了一个类似的比赛题目：https://blog.csdn.net/dongyanwen6036/article/details/77921407

里面提到发现&vcode= 和PHPSESSID= 后面的数据删了就绕过了验证码（验证码存在SESSION中，后台校验验证码时未判断是否存在SESSION)

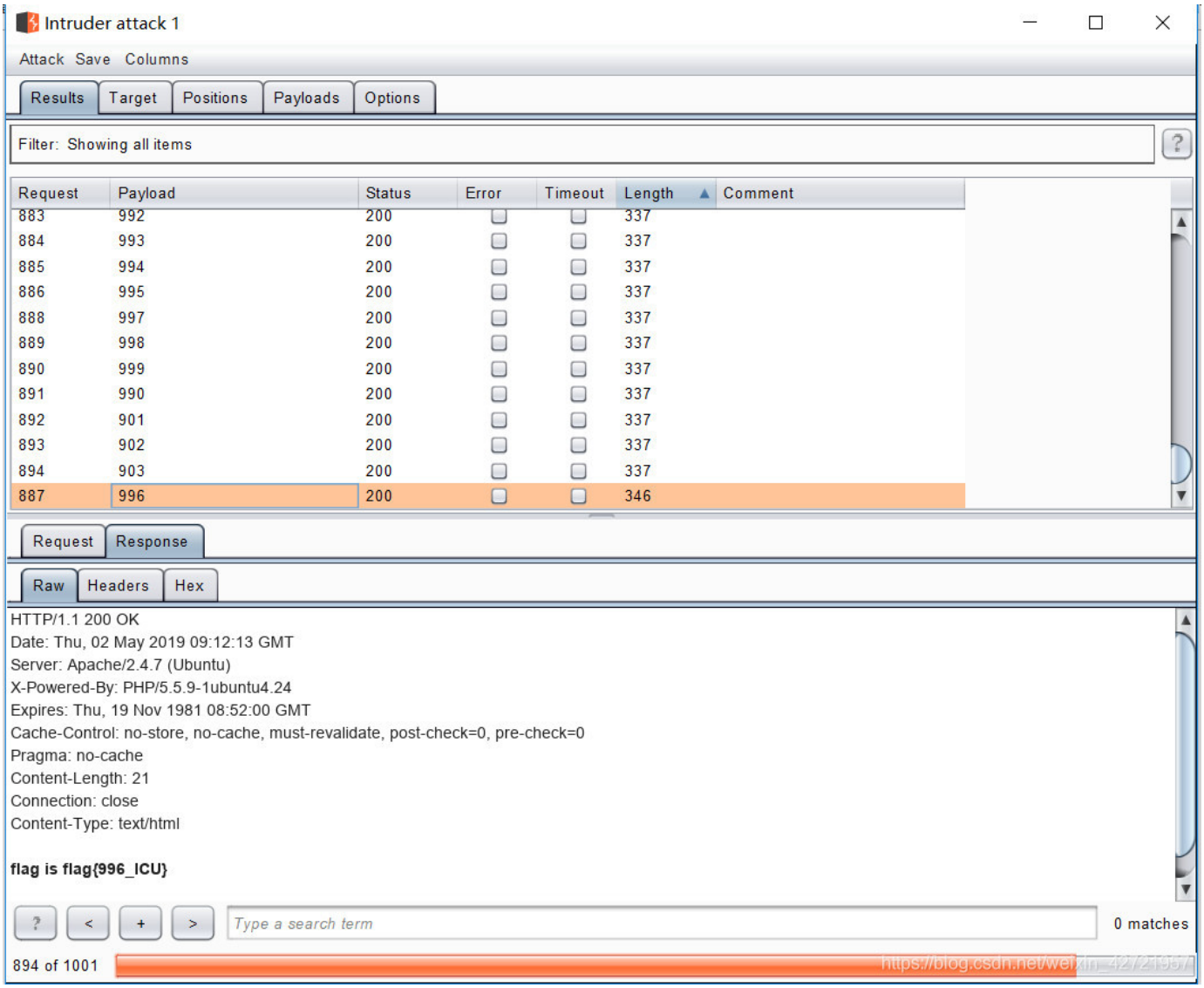
所以我想这里应该是类似的情况，所以祭出我的burpsuite吧



删除这里的两个数据，查看返回页面，居然确实是密码错误而不是验证码错误。事实证明暴力破解可行了



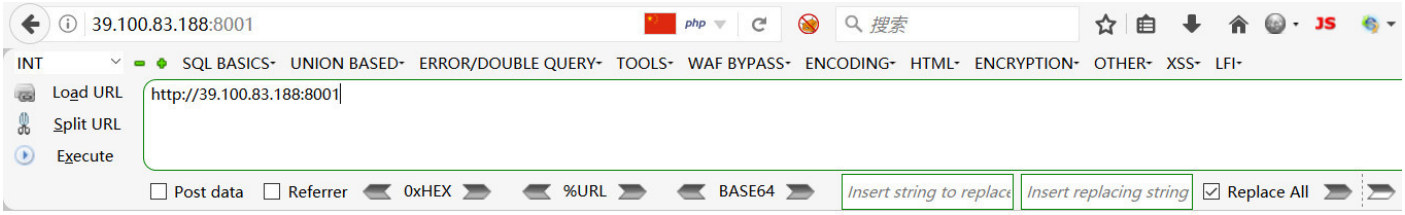
最坑爹最想不到的是数字密码居然是。。。996



果然程序员跟web狗最终都是要进ICU的啊！！！

2.

打开链接发现有php代码显示如下：




```php
<?php
error_reporting(0);
require 'flag.php';
$value = $_GET['value'];
$password = $_GET['password'];
$username = '';

for ($i = 0; $i < count($value); ++$i) {
        if ($value[$i] > 32 && $value[$i] < 127) unset($value);
        else $username .= chr($value[$i]);
        if ($username == 'w3lc0me_To_ISCC2019' && intval($password) < 2333 && intval($password + 1) > 2333) {
                echo 'Hello '.$username.'!', '<br>', PHP_EOL;
                echo $flag, '<hr>';
        }
}

highlight_file(__FILE__);
```


源代码：

```php
<?php
error_reporting(0);
require 'flag.php';
$value = $_GET['value'];
$password = $_GET['password'];
$username = '';

for ($i = 0; $i < count($value); ++$i) {
    if ($value[$i] > 32 && $value[$i] < 127) unset($value);
    else $username .= chr($value[$i]);
    if ($username == 'w3lc0me_To_ISCC2019' && intval($password) < 2333 && intval($password + 1) > 2333) {
        echo 'Hello '.$username.'!', '<br>', PHP_EOL;
        echo $flag, '<hr>';
    }
}

highlight_file(__FILE__);
```

大概意思就是包含了flag.php文件，然后有三个参数，其中value参数的值不能在32到127中间，也就是不能直接用ascll码代替。然后password参数用了intval()方法后要满足两个条件就得到flag。

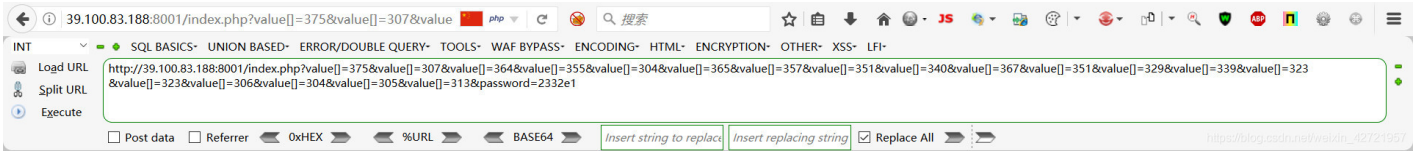
这里有两个技巧：

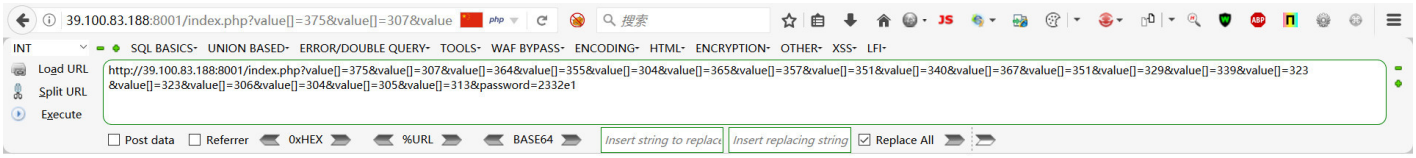
（1）chr函数在转换时会自动取模256,所以我们只需要在原本ascii码基础上+256即可

（2）intval()在处理16进制时存在问题,但强制转换时时正常的

所以这个题的payload是：

http://39.100.83.188:8001/index.php？value[]=375&value[]=307&value[]=364&value[]=355&value[]=304&value





Hello w3lc0me_To_ISCC2019!
flag{8311873e241ccad54463eaa5d4efc1e9}

```php
<?php
error_reporting(0);
require 'flag.php';
$value = $_GET['value'];
$password = $_GET['password'];
$username = '';

for ($i = 0; $i < count($value); ++$i) {
        if ($value[$i] > 32 && $value[$i] < 127) unset($value);
        else $username .= chr($value[$i]);
        if ($username == 'w3lc0me_To_ISCC2019' && intval($password) < 2333 && intval($password + 1) > 2333) {
                echo 'Hello '.$username.'!', '<br>', PHP_EOL;
                echo $flag, '<hr>';
        }
}

highlight_file(__FILE__);
```

3.

打开链接发现php源码如下：


```
<?php
error_reporting(0);
include("flag.php");
$hashed_key = 'ddbafb4eb89e218701472d3f6c087fdf7119dfdd560f9d1fcbe7482b0feea05a';
$parsed = parse_url($_SERVER['REQUEST_URI']);
if(isset($parsed["query"])){
        $query = $parsed["query"];
        $parsed_query = parse_str($query);
        if($parsed_query!=NULL){
                $action = $parsed_query['action'];
        }

        if($action==="auth"){
                $key = $_GET["key"];
                $hashed_input = hash('sha256', $key);
                if($hashed_input!==$hashed_key){
                        die("<img src='cxk.jpg'>");
                }

                echo $flag;
        }
}else{
        show_source(__FILE__);
}?>
```


源代码：

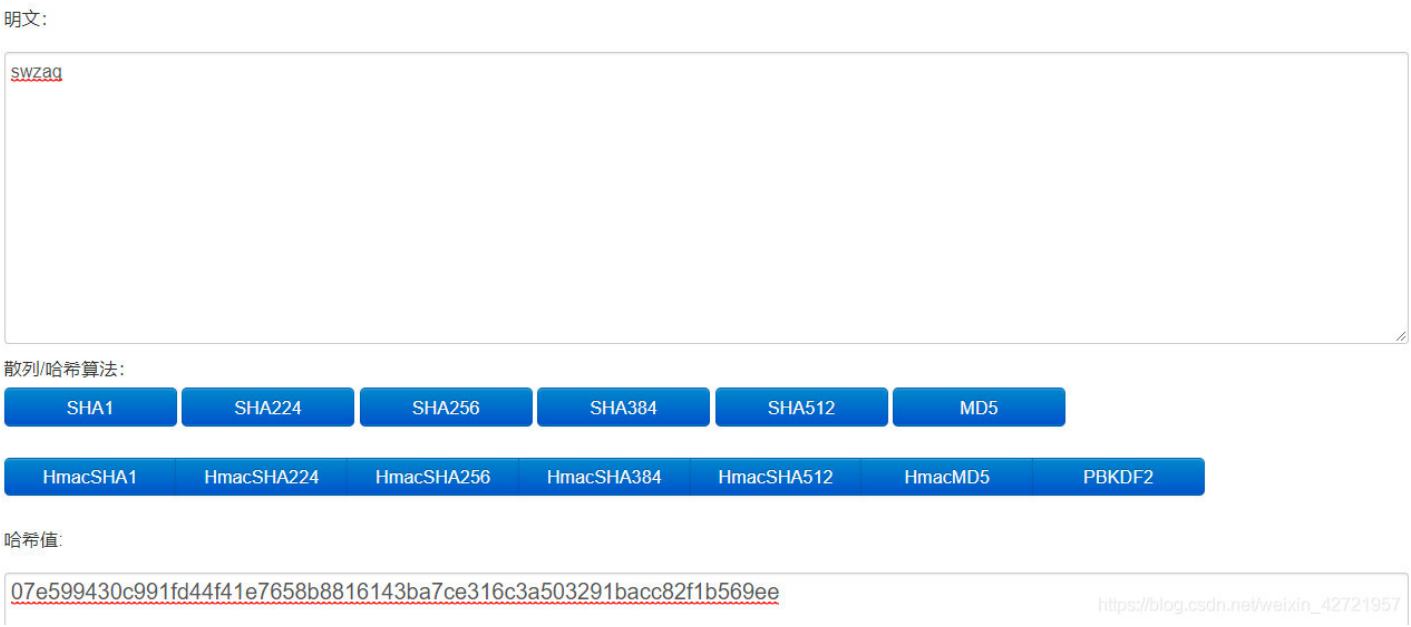
```
<?php
error_reporting(0);
include("flag.php");
$hashed_key = 'ddbafb4eb89e218701472d3f6c087fdf7119dfdd560f9d1fcbe7482b0feea05a';
$parsed = parse_url($_SERVER['REQUEST_URI']);
if(isset($parsed["query"])){
    $query = $parsed["query"];
    $parsed_query = parse_str($query);
    if($parsed_query!=NULL){
        $action = $parsed_query['action'];
    }

    if($action==="auth"){
        $key = $_GET["key"];
        $hashed_input = hash('sha256', $key);
        if($hashed_input!==$hashed_key){
            die("<img src='cxk.jpg'>");
        }

        echo $flag;
    }
}else{
    show_source(__FILE__);
}?>
```

这里简单审计一下，大概意思就是当发现传入的参数中 action 为 auth，并且 key 和 hashed_key 的值相等时，就给出 flag。

这里可以用到一个非常危险的函数 parse_str，具体情况请看 https://www.php.net/manual/zh/function.parse-str.php

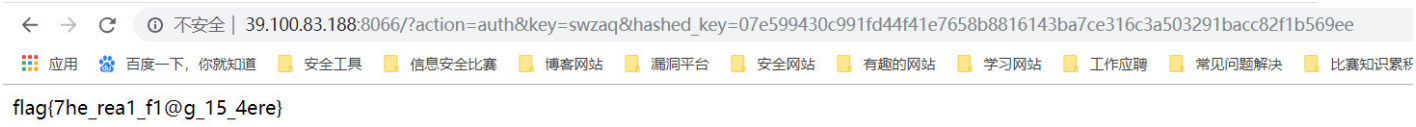
如果传入的是 query_string（形如first=value&arr[]=foo+bar&arr[]=baz ），那么就会将其解析为变量（设置变量 first=value, arr[0]=foo bar，arr[1]=baz）

所以利用这一漏洞，我们就可以实现变量覆盖了，直接将 hashed_key 覆盖为我们想要的值即可。这里我选择覆盖 sha256(“swzaq”) = 07e599430c991fd44f41e7658b8816143ba7ce316c3a503291bacc82f1b569ee



payload 如下：

```
/?action=auth&key=swzaq&hashed_key=07e599430c991fd44f41e7658b8816143ba7ce316c3a503291bacc82f1b569ee
```



flag{7he_rea1_f1@g_15_4ere}