ThinkPHP5.2.x反序列化利用链

mochazz / 2019-10-09 09:40:29 / 浏览数 5027 安全技术 漏洞分析 顶(0) 踩(0)

这个POP链来自 N1CTF2019 中的某道题目，不过和之前 ThinkPHP5.1.x反序列化POP链 差不多，只是当中替换了几个小 Gadget ，这里也记录一下。

环境搭建

```
➜ html composer create-project topthink/think=5.2.x-dev tp52x
➜ html cd tp52x
➜ tp52x ./think run
```

将 application/index/controller/Index.php 代码修改成如下：

```
<?php
namespace app\controller;

class Index
{
   public function index()
   {
       $u = unserialize($_GET['c']);
       return 'ThinkPHP V5.2';
   }
}
```

利用条件

有一个内容完全可控的反序列化点，例如： `unserialize(■■■■)`

存在文件上传、文件名完全可控、使用了文件操作函数，例如： `file_exists('phar://■■■■')`

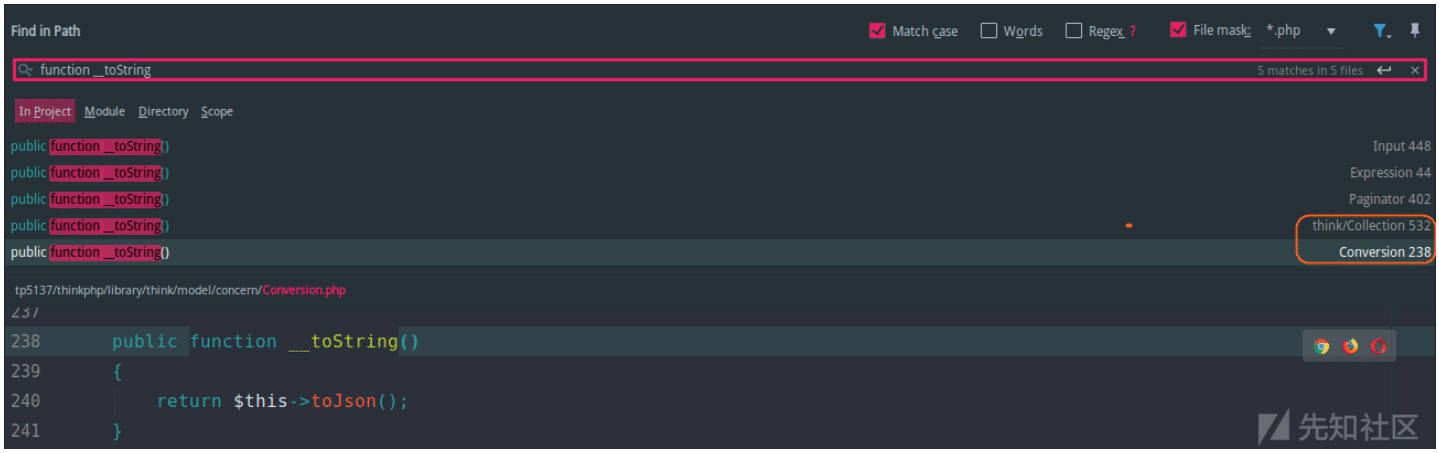
（满足以上任意一个条件即可）

POP链1

这个漏洞链个人认为比较有意思的是：通过 file_exists 函数触发类的 __toString 方法。下面，我们具体分析一下整个漏洞攻击链。

在 think\process\pipes\Windows 类的 __destruct 方法中，存在一个删除文件功能，而这里的文件名 $filename 变量是可控。如果我们将一个类赋值给 $filename 变量，那么在 file_exists($filename) 的时候，就会触发这个类的 __toString 方法。因为 file_exists 函数需要的是一个字符串类型的参数，如果传入一个对象，就会先调用该类 __toString 方法，获取其字符串返回值，然后再判断。（下图对应文件 vendor/topthink/framework/src/think/process/pipes/Windows.php ）

接下来，我们就来寻找可利用的 __toString 方法。全局搜索到的 __toString 方法其实不多，这里有两处都可以利用。它们的区别在于利用 think\Collection 构造的链要多构造一步，我们这里只分析链较短的 think\model\concern\Conversion 。



如下图所示，原先 ThinkPHP5.1.x 中 $relation->visible($name) 已经不见了，其实这段代码被移到了 第180行 的 appendAttrToArray 方法中。这里，我们先关注 第172行 的 getAttr 方法，这里传入的 $key 变量是来自第158行的可控变量 $data 。（下图对应文件 vendor/topthink/framework/src/think/model/concern/Conversion.php ）

在 getAttr 方法中，程序先通过第451行的 getData 获取了 $value 变量。从下图右侧的获取过程中，可以看出最终获得的 $value 变量值可控。然后在第457调用 getValue 方法，传入该方法的前两个变量值均可控，最后一个 $relation 值为 false 。我们跟进 getValue 方法，看其具体代码。（下图对应文件 vendor/topthink/framework/src/think/model/concern/Attribute.php ）

```php
447     public function getAttr(string $name)
448     {
449         try {
450             $relation = false;
451             $value    = $this->getData($name);
452         } catch (InvalidArgumentException $e) {
453             $relation = true;
454             $value    = null;
455         }
456
457         return $this->getValue($name, $value, $relation);
458     }
469     protected function getValue(string $name, $value, bool $relation = false)
470     {
471         // 检测属性获取器
472         $fieldName = $this->getRealFieldName($name);
473         $method    = 'get' . App::parseName($name, 1) . 'Attr';
474
475         if (isset($this->withAttr[$fieldName])) {
476             if ($relation) {...}
479
480             $closure = $this->withAttr[$fieldName];
481             $value   = $closure($value, $this->data);
482         } elseif (method_exists($this, $method)) {
497         return $value;
498     }
```

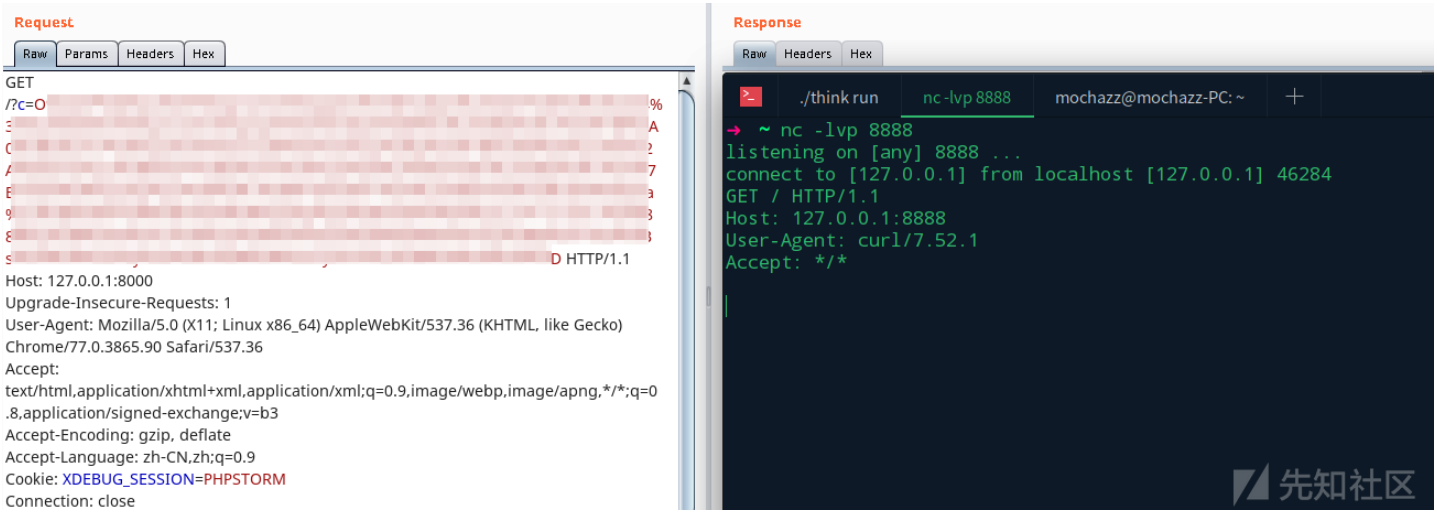
```php
262 public function getData(string $name = null)
263 {
264     if (is_null($name)) {
265         return $this->data;
266     }
267
268     $fieldName = $this->getRealFieldName($name);
269
270     if (array_key_exists($fieldName, $this->data)) {
271         return $this->data[$fieldName];
272     } elseif (array_key_exists($name, $this->relation)) {
273         return $this->relation[$name];
274     }
275
276     throw new InvalidArgumentException('property not exis
277 }
177 protected function getRealFieldName(string $name): string
178 {
179     return $this->strict ? $name : App::parseName($name);
180 }
```


可以看到在 getValue 方法中，使用了动态调用（上图第481行），而且这里的 $closure、$value、$this->data 均可控。我们只要让 $closure='system' 并且 $value='要执行的命令' ，就可以触发命令执行。但是上面的 Attribute、Conversion 是 trait ，不能直接用来构造 EXP ，我们得找使用了这两个 trait 的类。

```php
vendor/topthink/framework/src/think/Model.php
56  abstract class Model implements JsonSerializable, ArrayAccess
57  {
58      use model\concern\Attribute;
59      use model\concern\RelationShip;
60      use model\concern\ModelEvent;
61      use model\concern\TimeStamp;
62      use model\concern\Conversion;
1015 }

vendor/topthink/framework/src/think/model/Pivot.php
13  namespace think\model;
14
15  use think\Model;
16
17  class Pivot extends Model{...}
```


这里我们找到了符合条件的 Pivot 类，所以这条链的 EXP 如下（例如这里执行 `curl 127.0.0.1:8888` ）：



POP链2

第二条POP链其实不太好用，需要目标站点可以上传 route.php 文件，且知道上传后文件的存储路径，下面我们来看下具体POP链。

这个POP链的前半部分，和原先 ThinkPHP5.1.x 中的POP链是一样的。只不过在执行到下图第193行时， ThinkPHP5.1.x 中的POP链会去触发 Request 类的 __call 方法，而在 ThinkPHP5.2.x 中移除了 Request 类的 __call 方法，所以我们需要寻找新的可用 __call 方法。

thinkphp/library/think/model/concern/Conversion.php

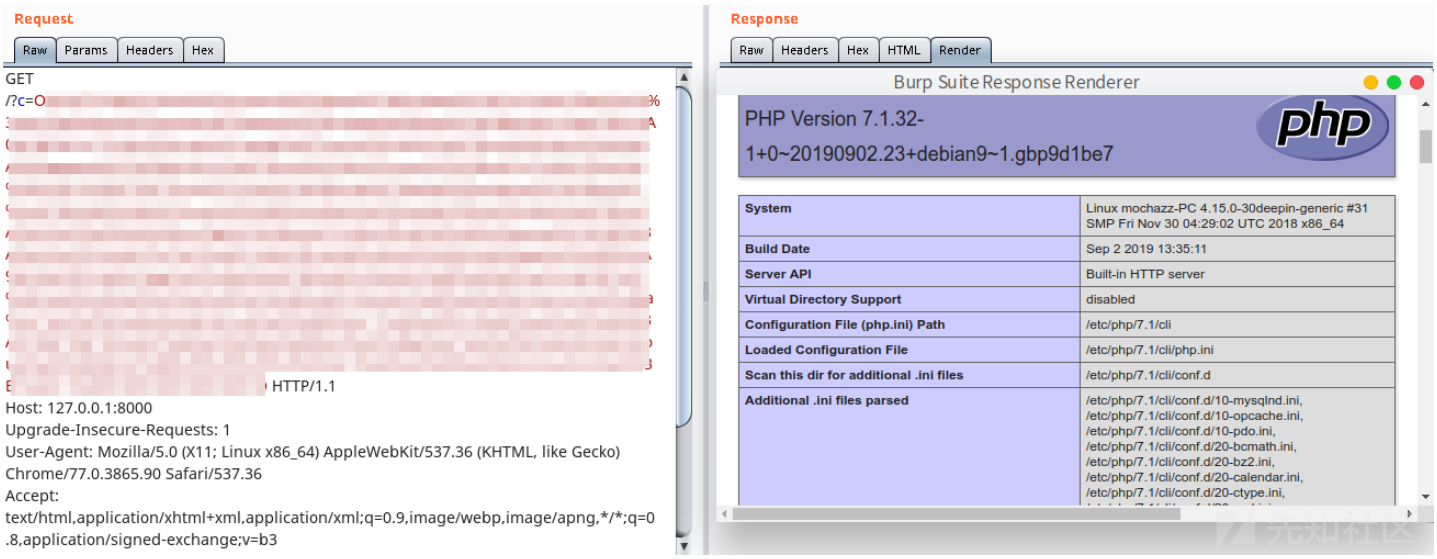
```
25  trait Conversion
26  {
238     public function __toString()
239     {
240         return $this->toJson();
241     }
222     public function toJson($options = JSON_UNESCAPED_UNICODE)
223     {
224         return json_encode($this->toArray(), $options);
225     }
127     public function toArray(): array
128     {
178         // 追加属性（必须定义获取器）
179         foreach ($this->append as $key => $name) {
180             $this->appendAttrToArray($item, $key, $name);
181         }
182
183         return $item;
184     }
185
186     protected function appendAttrToArray(array &$item, $key, $name)
187     {
188         if (is_array($name)) {
189             // 追加关联对象属性
190             $relation = $this->getRelation($key);
191
192             if (!$relation) {
193                 $relation = $this->getAttr($key);
194                 $relation->visible($name);
195             }
215     }
```

$relation完全可控

这里，我们使用 Db 类的 __call 方法，因为该方法可以实例化任意类（下图第203行）。结合 Url 类的 __construct 方法，从而进行文件包含。如果攻击者可以上传 route.php 文件，并知道文件存储位置，即可 getshell 。

```
vendor/topthink/framework/src/think/Db.php
18  class Db
19  {
199     public function __call($method, $args)
200     {
201         $class = $this->config['query'];
202
203         $query = new $class($this->connection);   实例化任意类
204
205         return call_user_func_array([$query, $method], $args);
206     }
207 }
    vendor/topthink/framework/src/think/Url.php
15  class Url
16  {
41      public function __construct(App $app, array $config = [])
42      {
43          $this->app    = $app;
44          $this->config = $config;
45
46          if (is_file($app->getRuntimePath() . 'route.php')) {
47              // 读取路由映射文件
48              $app->route->import(include $app->getRuntimePath() . 'route.php');
49          }
50      }
402 }
    vendor/topthink/framework/src/think/App.php
24  class App extends Container
25  {
271     public function getRuntimePath(): string
272     {
273         return $this->runtimePath;   可控
274     }
591 }
```


最终，这条链的 EXP 如下（这里我事先上传了 route.php 到 /tmp/ 目录下）：



PS：为了避免不必要的麻烦，文中EXP均已删除。

参考

[N1CTF2019 sql_manage出题笔记]

[thinkphp v5.2.x 反序列化利用链挖掘]

上一篇：how2heap 问题汇总(下) 下一篇：浅谈 ThinkPHP 中的注入

1. 1 条回复

Hitman 2019-10-14 17:09:14

这个值得研究

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