CatfishCMS 4.5.7 csrf getshell

一、漏洞简介

二、漏洞影响

CatfishCMS 4.5

三、复现过程

思路：

前台评论出插入 xss 代码->诱骗后台管理员访问网站-内容管理-评论管理-自动执行 xss 代码->通过 csrf 插入一条新文章->通过 csrf 清除缓存->在通过 js 访问前端任意页面生成缓存建立 shell
大概的想法就是这样做了。

后台创建文章方法

地址：application\admin\controller\Index.php

方法：write();

这个方法没有什么可以讲的只是后面的组合漏洞要使用到他

后台清除缓存方法

地址：application\admin\controller\Index.php

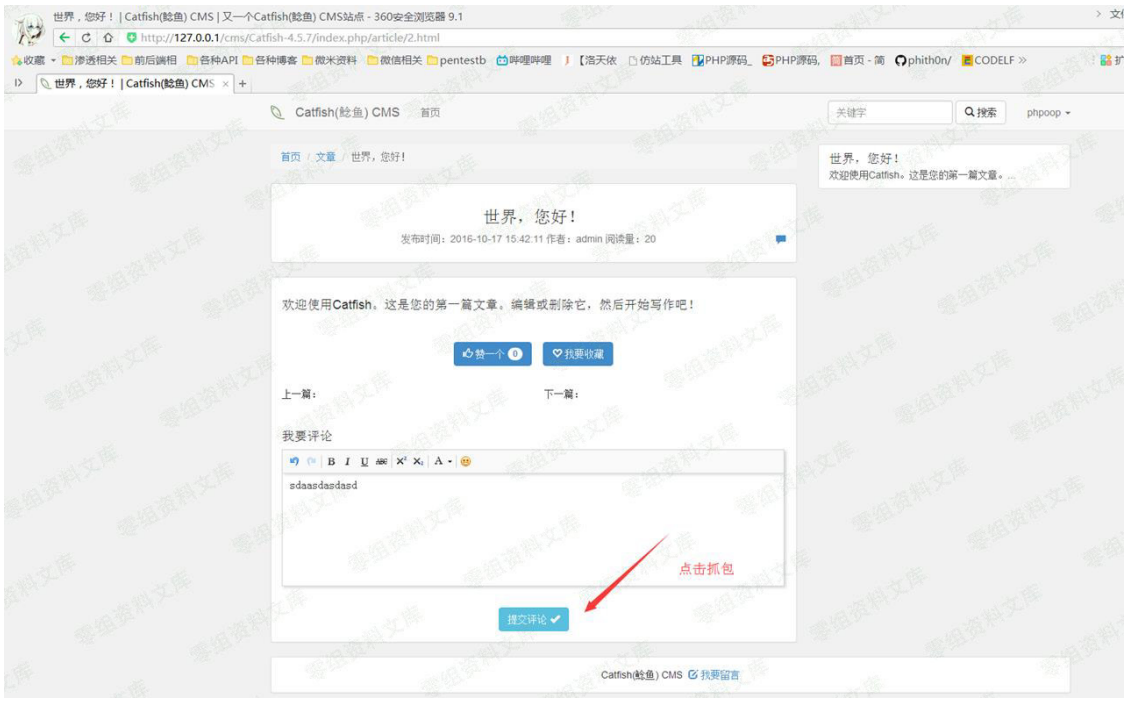
方法：clearcache()

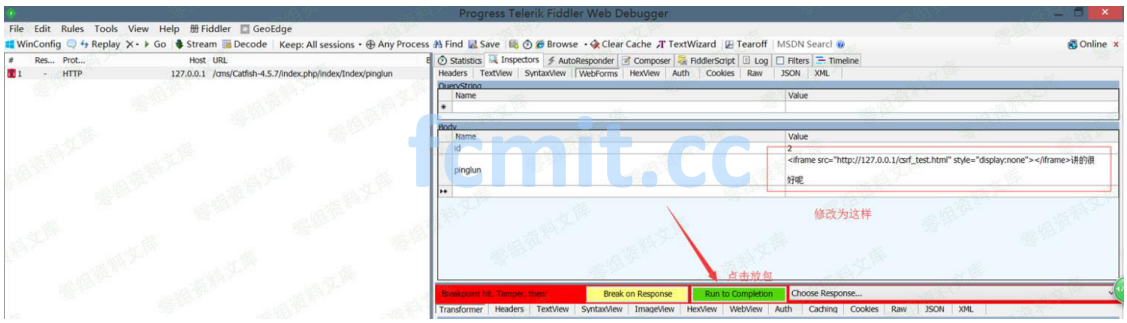
这个方法没有什么可以讲的只是后面的组合漏洞要使用到他

例子：

1， 准备好脚本

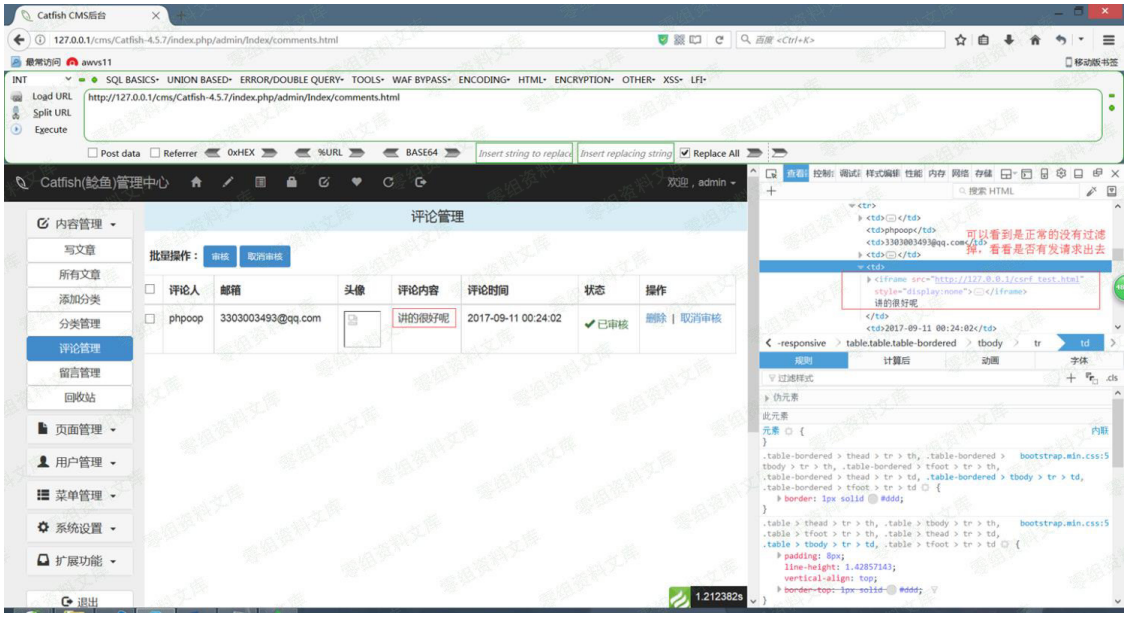
2，利用前面的 xss 漏洞，配合这个脚本形成 xsrf 漏洞

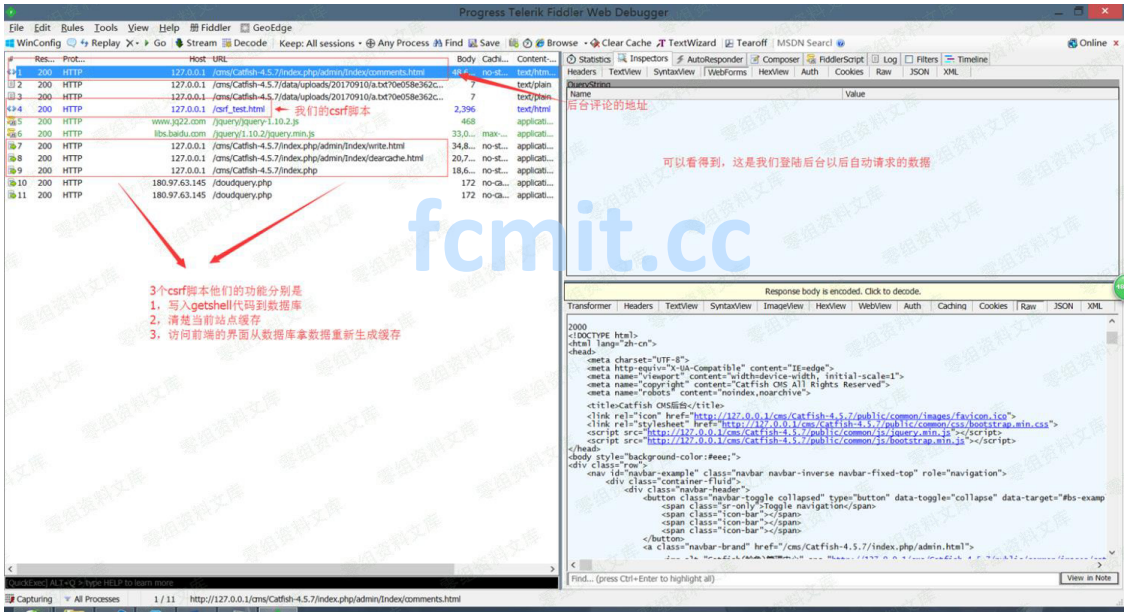


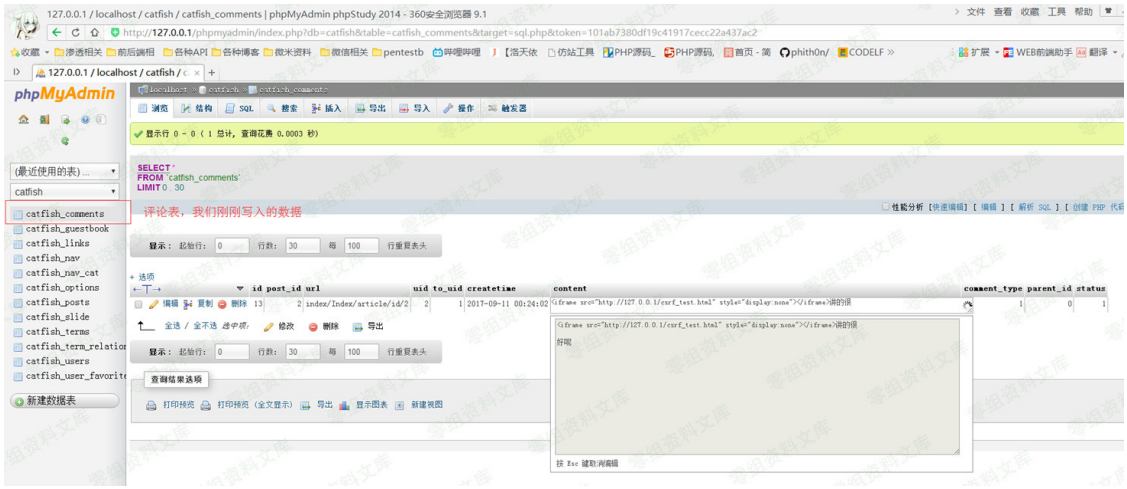


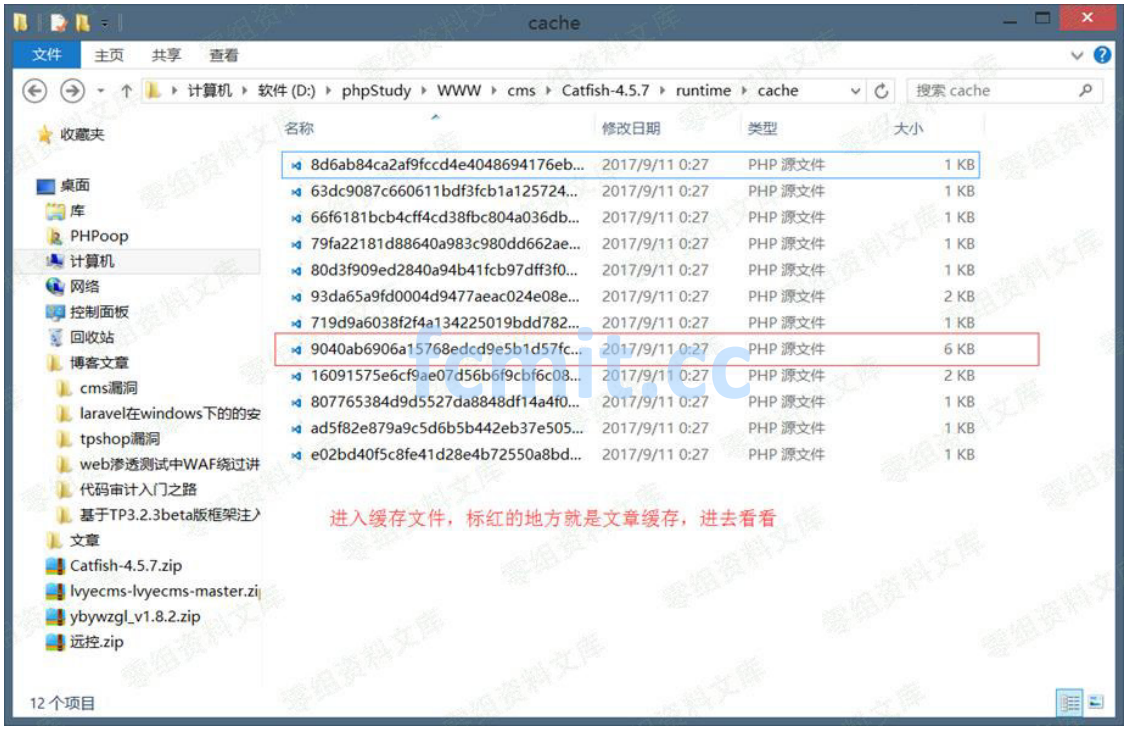
这样我们在前端的事情就完事了。接着我们模拟后台管理员进入后台的操作

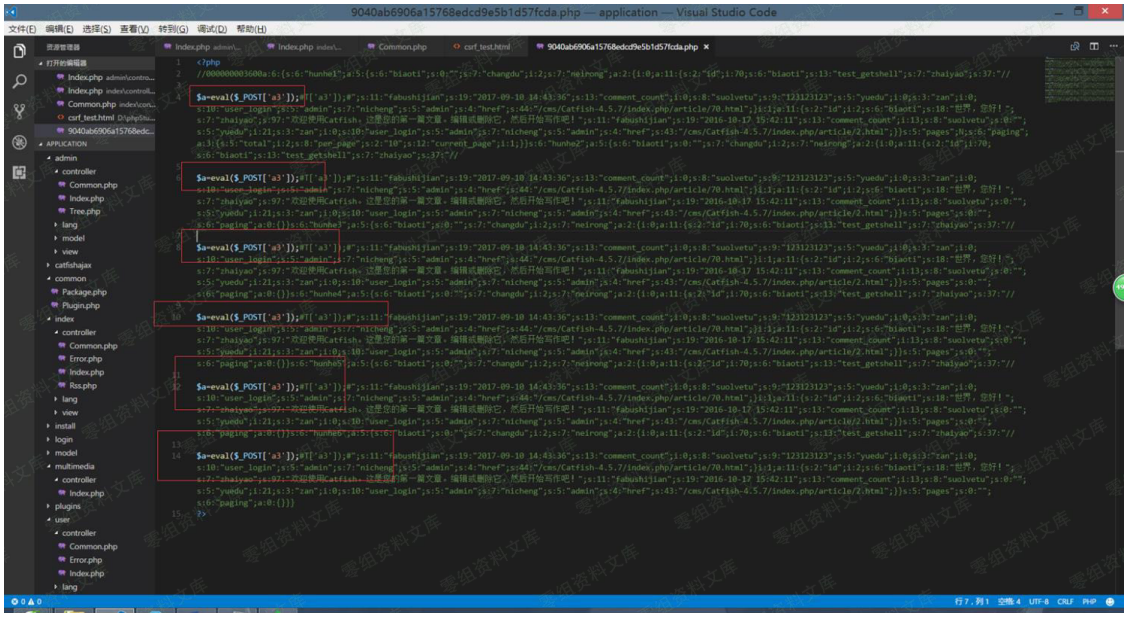
模拟的后端管理员操作：


Catfish CMS后台
127.0.0.1/cms/Catfish-4.5.7/index.php/admin/Index/comments.html
Load URL
Split URL
Execute
http://127.0.0.1/cms/Catfish-4.5.7/index.php/admin/Index/comments.html
Catfish(鲶鱼)管理中心
欢迎，admin
评论管理
内容管理
写文章
所有文章
添加分类
分类管理
评论管理
留言管理
回收站
页面管理
用户管理
菜单管理
系统设置
扩展功能
退出
批量操作：
审核
取消审核
评论人
邮箱
头像
评论内容
评论时间
状态
操作
phpoop
3303003493@qq.com
讲的很好呢
2017-09-11 00:24:02
已审核
删除 | 取消审核
可以看到是正常的没有过滤掉，看看是否有发请求出去
1.212382s


Progress Telerik Fiddler Web Debugger
我们的csrf脚本
后台评论的地址
可以看得到，这是我们登陆后台以后自动请求的数据
3个csrf脚本他们的功能分别是
1，写入getshell代码到数据库
2，清楚当前站点缓存
3，访问前端的界面从数据库拿数据重新生成缓存
fcmit.cc


phpMyAdmin
catfish_comments
catfish_guestbook
catfish_links
catfish_nav
catfish_nav_cat
catfish_options
catfish_posts
catfish_slide
catfish_terms
catfish_users
评论表，我们刚刚写入的数据


cache
文件
主页
共享
查看
计算机 ▸ 软件 (D:) ▸ phpStudy ▸ WWW ▸ cms ▸ Catfish-4.5.7 ▸ runtime ▸ cache
搜索 cache
名称
修改日期
类型
大小
8d6ab84ca2af9fccd4e4048694176eb... 2017/9/11 0:27 PHP 源文件 1 KB
63dc9087c660611bdf3fcb1a125724... 2017/9/11 0:27 PHP 源文件 1 KB
66f6181bcb4cff4cd38fbc804a036db... 2017/9/11 0:27 PHP 源文件 1 KB
79fa22181d88640a983c980dd662ae... 2017/9/11 0:27 PHP 源文件 1 KB
80d3f909ed2840a94b41fcb97dff3f0... 2017/9/11 0:27 PHP 源文件 1 KB
93da65a9fd0004d9477aeac024e08e... 2017/9/11 0:27 PHP 源文件 2 KB
719d9a6038f2f4a134225019bdd782... 2017/9/11 0:27 PHP 源文件 1 KB
9040ab6906a15768edcd9e5b1d57fc... 2017/9/11 0:27 PHP 源文件 6 KB
16091575e6cf9ae07d56b6f9cbf6c08... 2017/9/11 0:27 PHP 源文件 2 KB
807765384d9d5527da8848df14a4f0... 2017/9/11 0:27 PHP 源文件 1 KB
ad5f82e879a9c5d6b5b442eb37e505... 2017/9/11 0:27 PHP 源文件 1 KB
e02bd40f5c8fe41d28e4b72550a8bd... 2017/9/11 0:27 PHP 源文件 1 KB
进入缓存文件，标红的地方就是文章缓存，进去看看
12 个项目



漏洞原理与流程：

1,后台创建文章方法
地址：application\admin\controller\Index.php
方法：write();
这个方法没有什么可以讲只是单纯的从前端获取数据然后写入数据库罢了

2,后台清除缓存方法
地址：application\admin\controller\Index.php
方法：clearcache()
这个方法没有什么可以讲的。只是单纯的删除缓存数据

3,访问前端重新生成缓存
地址： application\index\controller\Index.php
方法：index()

```php
    $param = '';
    Hook::add('view_post',$this->plugins);
    Hook::listen('view_post',$param,$this->ccc);
    $param = '';
    Hook::add('home_plugin_name',$this->plugins);
    Hook::listen('home_plugin_name',$param,$this->ccc);
    if(!empty($param))
    {
        $this->assign('plugin_name', $param);
    }
    $this->slide();
    $data_links = Cache::get('links');
    if($data_links == false)
    {
        $data_links = Db::name('links')->where('link_location',1)->where('link_status',1)->field('link_url,link_name,link_image,link_target')->order('listorder')->select();
        Cache::set('links',$data_links,3600);
    }
    $this->assign('links', $data_links);

    //强操作执行方法 设置文章缓存
    $template = $this->receive('index');
    $this->assign('pageUrl', $this->getpage());
    if(Request::instance()->isMobile() && is_file(APP_PATH.'../public/'.$template.'/mobile/index.html'))
    {
        $htmls = $this->fetch(APP_PATH.'../public/'.$template.'/mobile/index.html');
    }
    else
    {
        $htmls = $this->fetch(APP_PATH.'../public/'.$template.'/index.html');
    }
    return $htmls;
}
```

跟进index方法，这里调用了一个函数，用来生成文章缓存的，跟进去查看

```php
<?php
/**
 * Project: Catfish.
 * Author: A.J
 * Date: 2016/9/29
 */
namespace app\index\controller;

use app\index\controller\Common;
use think\Db;
use think\Cache;
use think\Url;
use think\Request;
use think\Session;
use think\Validate;
use think\Hook;
use think\Lang;

class Index extends Common
{
    public function index()
    {
        Hook::add('home_top',$this->plugins);
        Hook::add('home_mid',$this->plugins);
        Hook::add('home_bottom',$this->plugins);
        Hook::add('home_extend',$this->plugins);
        Hook::add('home_side_top',$this->plugins);
        Hook::add('home_side_mid',$this->plugins);
        Hook::add('home_side_bottom',$this->plugins);

        // // var_dump( $this->params );
        Hook::listen('home_top',$this->params,$this->ccc);
        Hook::listen('home_mid',$this->params,$this->ccc);
        Hook::listen('home_bottom',$this->params,$this->ccc);
        Hook::listen('home_extend',$this->params,$this->ccc);
        Hook::listen('home_side_top',$this->params,$this->ccc);
```

父类的路径

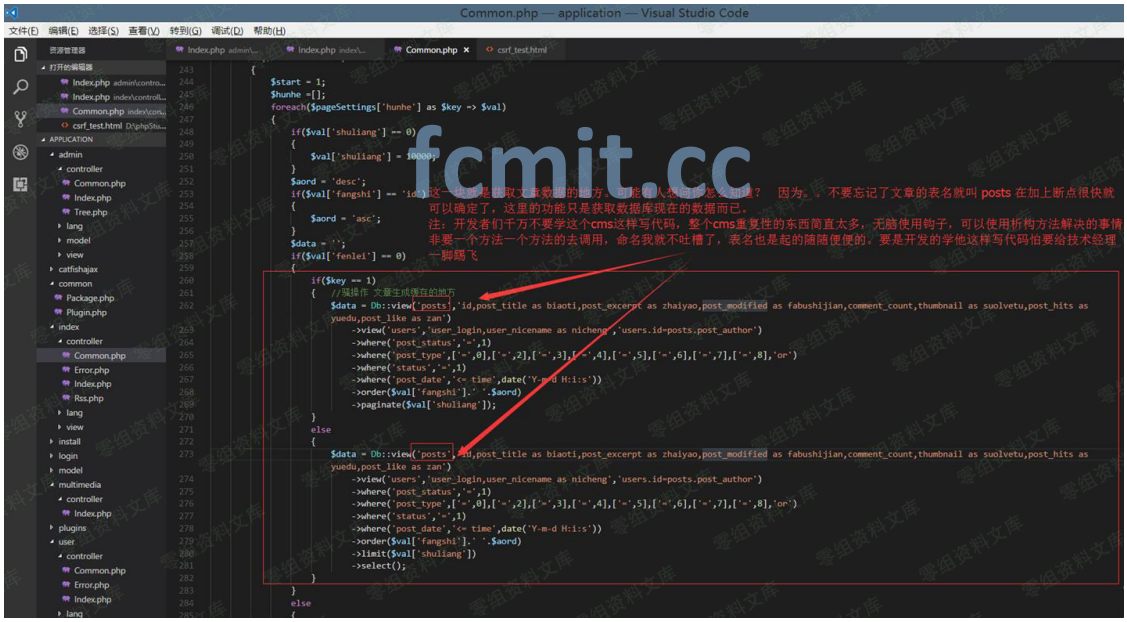
这里引入了一个父类，说明刚刚吊用的方法，可能就是父类的里面的

```php
protected function receive($source = '') //$source 缓存名字之一
{
    if(!isset($this->cocc) || $this->cocc != md5('Copyright owned by catfish CMS'))
        return false;



    $param = '';
    Hook::add('show_ready',$this->plugins);
    Hook::listen('show_ready',$param,$this->ccc);
    $domain = Cache::get('domain');
    if($domain == false)
    {
        $domain = Db::name('options')->where('option_name','domain')->field('option_value')->find();
        $domain = $domain['option_value'];
        Cache::set('domain',$domain,3600);
    }
    $this->assign('domain', $domain);
    $root = '';
    $dm = Url::build('/');
    if(strpos($dm,'/index.php') !== false)
    {
        $root = 'index.php/';
    }
    $this->assign('root', $root);

    $cqn = 'aWQ9ImNhdGZpc2giIHN0eWx1';
    $data_options = Cache::get('options');

    if($data_options == false)
    {
        $data_options = Db::name('options')->where('autoload',1)->field('option_name,option_value')->select();
        Cache::set('options',$data_options,3600);
    }
    $version = Config::get('version');
    $chn = 'PSJkaXNwbGF5Om5vbmU7Ig';
    $pushPage = '';

    if($this->actualDomain())
    {
        $pushPage = '<script src="'.$domain.'public/common/js/pushPage.js"></script>';
    }

    $this->assign('catfish', '<a href="http://www.'.$version['official'].'/" target="_blank" '.base64_decode($cqn.$chn.'==').'>'.$version['name'].' '.$version['description'].'
    '.$version['number'].'</a>  '.$pushPage);
    $template = 'default';
```

钩子方法

取缓存

如果没有缓存或是缓存过期时执行此方法



这一块就是获取文章数据的地方，可能有人会问为什么这么复杂？ 因为。。不要忘记了文章的表名就叫 posts 在加上断点很快就可以确定了，这里的功能只是获取数据库现在的数据而已。

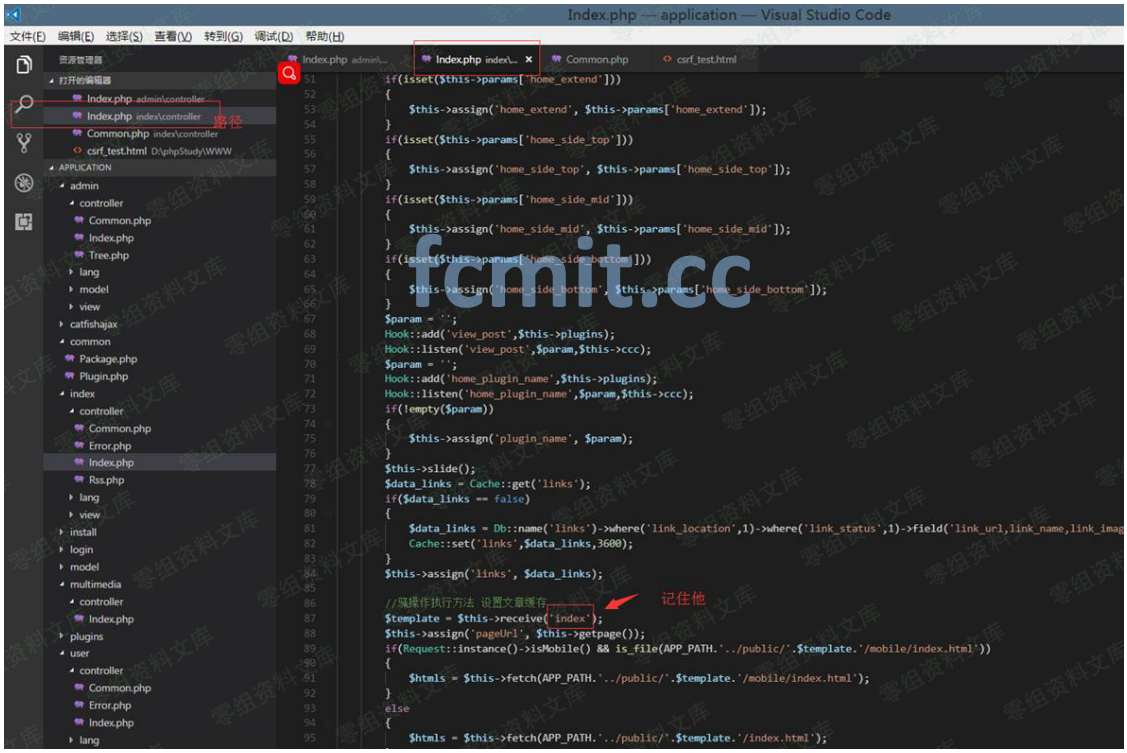
注：开发者们千万不要学这个cms这样写代码，整个cms重复性的东西简直太多，无脑使用钩子，可以使用析构方法解决的事情非要一个方法一个方法的去调用，命名我就不吐槽了，表名也是起的随随便便的。要是开发的学他这样写代码怕要给技术经理一脚踢飞

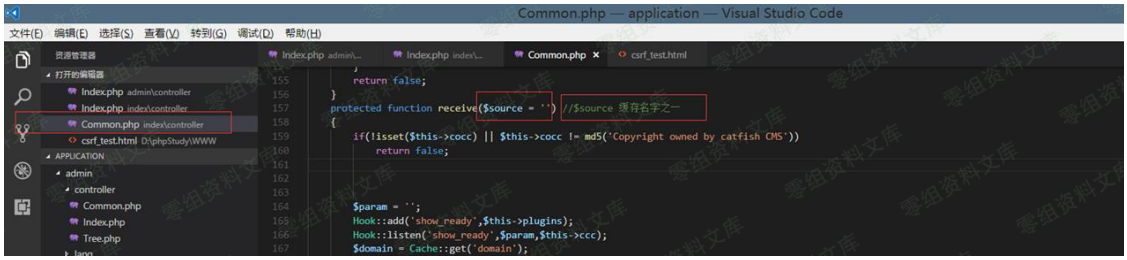
```php
            $pages = $data->render();
            $pageArr = $data->toArray();
            $data = $this->addLargerPicture($pageArr['data']);
            unset($pageArr['data']);
        }
        else
        {
            $pages = '';
            $pageArr = [];
        }
        $hunhe['hunhe'.$start] = [
            'biaoti' => $val['biaoti'],
            'changdu' => count($data),
            'neirong' => $this->addArticleHref($data),
            'pages' => $pages,
            'paging' => $pageArr
        ];
        $start++;
    }
    //骚操作 实际执行的文章缓存方法
    Cache::set('hunhe_'.$source.$page,$hunhe,3600);
}
```

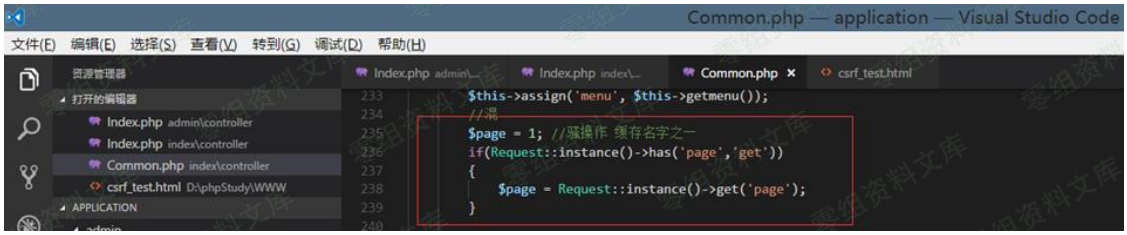
看过我前面发的那个简书的文章，大家看到这里应该就懂了吧？我们写入的文章数据经过这里，因为我在写入的数据中写入了回车符，TP缓存里面使用的又是 // 的注释导致了绕过换行，最终导致了getshell的风险。TP官方说了，都是开发者的错，谁让你们自己不按TP的文档来的~~~~ 所以这个漏洞就是开发者的错咯

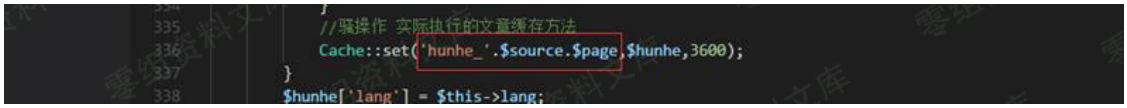
缓存的名字由来

缓存的名字组成就是比较简单的了。









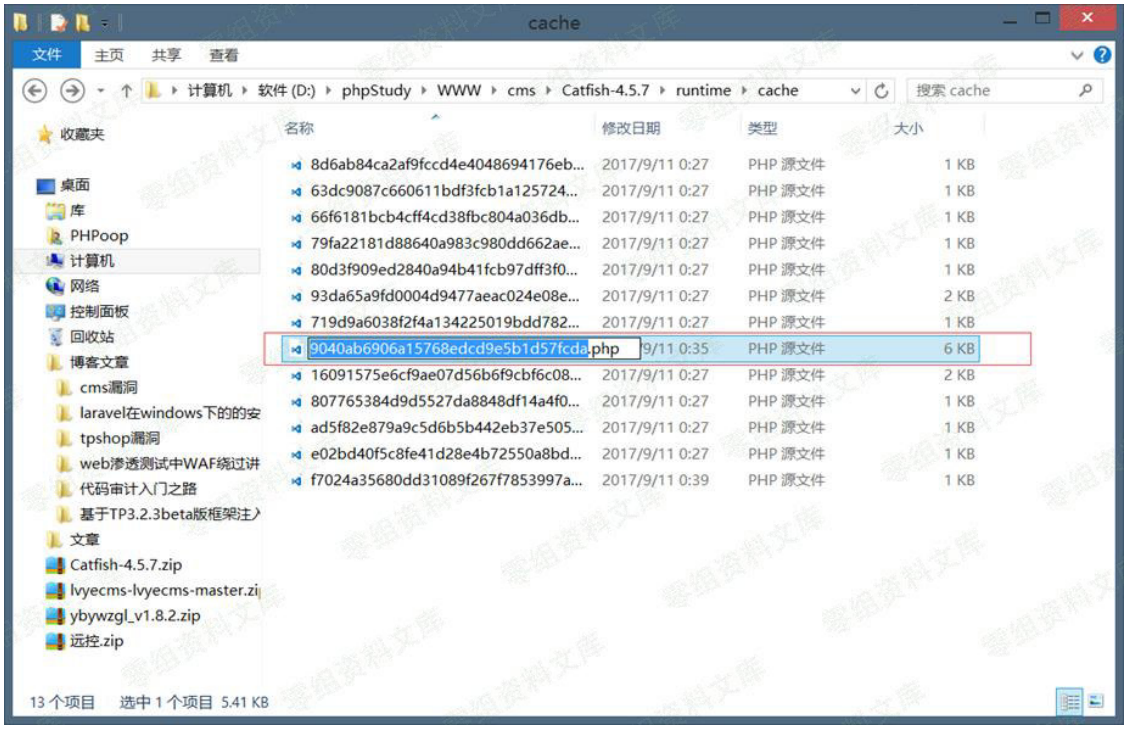
这上面几幅图就是缓存的名字了什么意思呢？很简单

首先是从 index 目录里面的 index 模块下面的 index 方法
调用了一个方法
$template
= $this->receive('index'); = index

然后是 ndex 目录里面的 Common 模块里面的 receive 方法
获取了变量$source 值 = index
获取了变量$page 值 = 1

Cache::set('hunhe_'.$source.$page,$hunhe,3600); 缓存方法
最后就是

MD5(hunhe_index1) = 9040ab6906a15768edcd9e5b1d57fcda



后记：

使用此方法的话，尝试一下在 url 中输入

```
http://www.xxxxxxx.com/runtime
http://www.xxxxxxx.com/runtime/cache
http://www.xxxxxxx.com/runtime/cache/8d6ab84ca2af9fccd4e4048694176ebf.php
```

按顺序输入如果前两个访问得到的结果是 403　最后的结果不是 403 或是 404 而是返回正常的页面，那么说明站点的缓存目录是可以访问的，这个时候可以使用此漏洞。配合 xss+csrf 获取 getshell