

## 0x05 清楚地知道系统参数过滤情况

## 0x05.1 原生 GET , POST , REQUEST

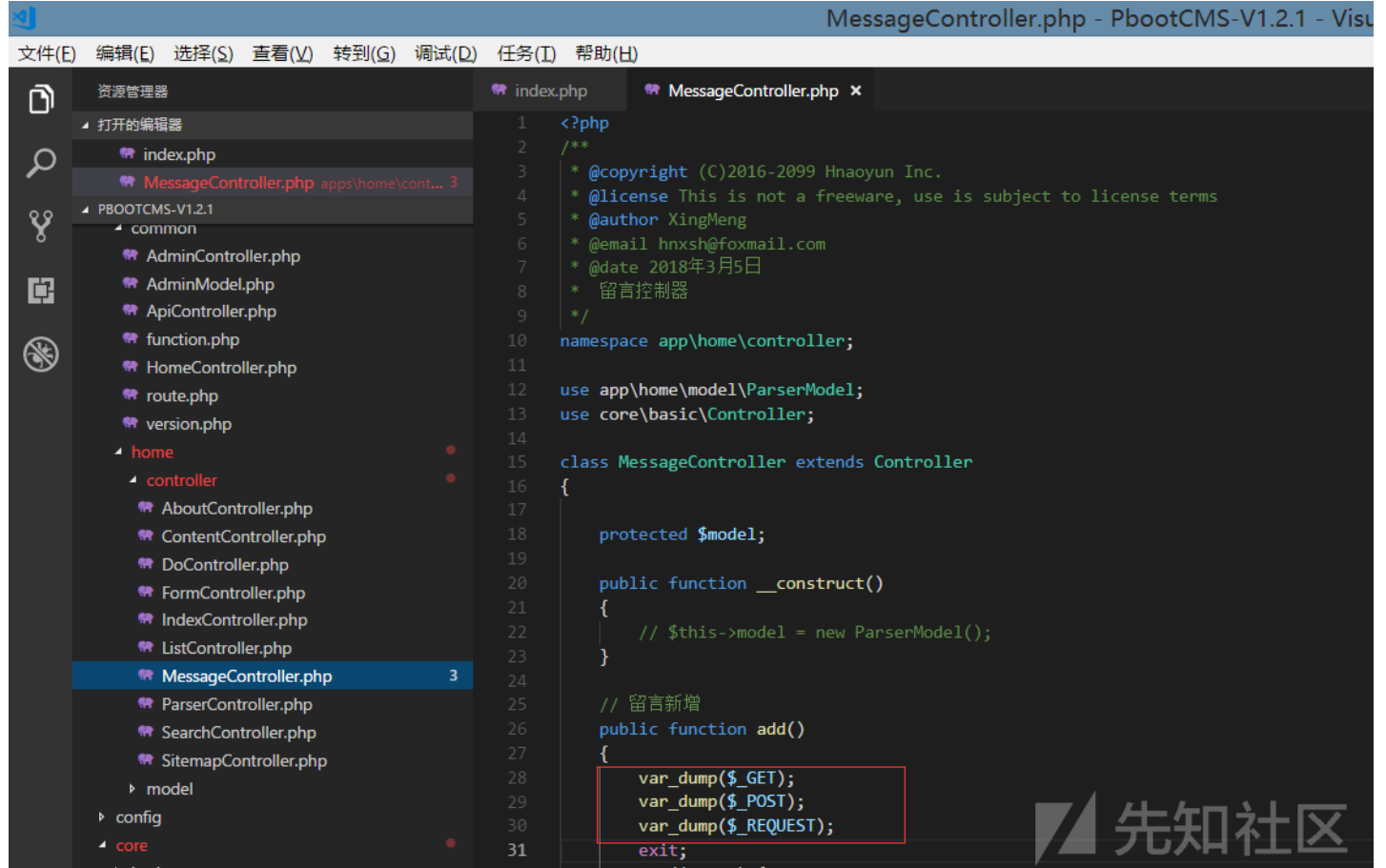
使用原生GET , POST , REQUEST变量是完全不过滤的

测试方法：

最简单的就是这样了

顺便找处系统中可外部访问的地方

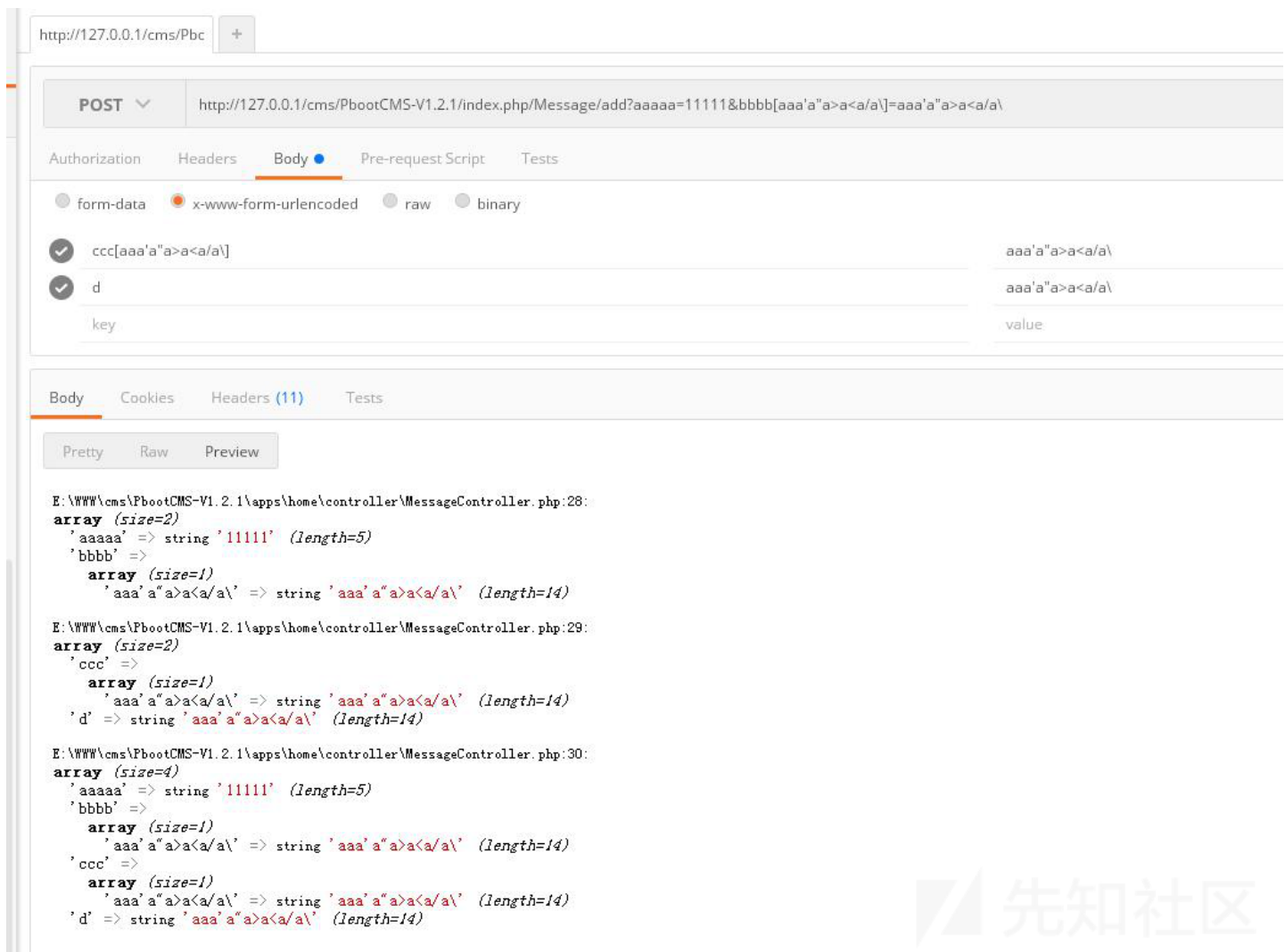
如图：



```
MessageController.php - PbootCMS-V1.2.1 - Visu
文件(E) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 任务(I) 帮助(H)

资源管理器
  打开的编辑器
    index.php
    MessageController.php apps\home\cont... 3
  PBOOTCMS-V1.2.1
    common
      AdminController.php
      AdminModel.php
      ApiController.php
      function.php
      HomeController.php
      route.php
      version.php
    home
      controller
        AboutController.php
        ContentController.php
        DoController.php
        FormController.php
        IndexController.php
        ListController.php
        MessageController.php 3
        ParserController.php
        SearchController.php
        SitemapController.php
      model
      config
      core

1  <?php
2  /**
3   * @copyright (C)2016-2099 Hnaoyun Inc.
4   * @license This is not a freeware, use is subject to license terms
5   * @author XingMeng
6   * @email hnxsh@foxmail.com
7   * @date 2018年3月5日
8   * 留言控制器
9   */
10 namespace app\home\controller;
11
12 use app\home\model\ParserModel;
13 use core\basic\Controller;
14
15 class MessageController extends Controller
16 {
17
18     protected $model;
19
20     public function __construct()
21     {
22         // $this->model = new ParserModel();
23     }
24
25     // 留言新增
26     public function add()
27     {
28         var_dump($_GET);
29         var_dump($_POST);
30         var_dump($_REQUEST);
31         exit;
32     }
33 }
```



根据结果就可以确定当使用了原生GET, POST, REQUEST变量带入数据库之类,是有可能产生注入,储蓄xss之类的

## 0x05.2 系统外部变量获取函数 get(), post(), request()

路径：PbootCMS-V1.2.1\core\function\helper.php

函数\_1 : function get(

函数\_2 : function post(

函数\_3 : function request(

这3个函数 `get()`, `post()`, `request()`

函数：filter(

最后都会调用 filter 函数

这个函数没有用所以直接看 filter 里面调用的方法 escape\_string函数进行最终过滤

因为filter无用的内容太多所以我们忽略掉直接看escape\_string函数

路径：PbootCMS-V1.2.1\core\function\handle.php

函数：function escape\_string(

```
// ██████████
function escape_string($string, $dropStr = true)
{
    if (! $string)
        return $string;
    if (is_array($string)) { // ██████
        foreach ($string as $key => $value) {
            $string[$key] = escape_string($value);
        }
    } elseif (is_object($string)) { // ██████
        foreach ($string as $key => $value) {
            $string->$key = escape_string($value);
        }
    } else { // ██████
        if ($dropStr) {
            $string = preg_replace('/(0x7e)|(0x27)|(0x22)|(updatexml)|(extractvalue)|(name_const)|(concat)/i', '', $string);
        }
    }
}
```



[illegible]



下图为触发点查看：



## 0x07 系统情况初步集合

经过0x05.1与0x05.2 还有0x06组合下来我们其实可以确定了一些基本漏洞了

### 5.1 反应给我的情况

首先是xss漏洞

储蓄xss

如果是使用了 5.1 的外部变量并且入库的时候没有转义的话，那么就会产生xss

反射xss

反不动 只要 GET 请求中出现了 A-Z a-z 0-9 之外的数，就会直接报错

然后是sql注入漏洞

如果是使用了 5.1 的外部变量并且入库的时候没有转义的话，那么就会产生sql注入

### 5.2 反应给我的情况

首先是xss漏洞

储蓄xss

如果是使用了 5.2 的外部变量入库的，想找xss 那么就要看使用了 htmlspecialchars\_decode 函数的地方，否则的话就只能查看类似这种点 <img src=#{variate}}> variate 可控的情况

反射xss

射不动 只要 GET 请求中出现了 A-Z a-z 0-9 之外的数，就会直接报错

然后是sql注入漏洞

key 没有过滤所以如果我能够控制 key 进入sql的话，那么就基本上百分之90有sql注入了

### 6.0 给我的情况

底层全是字符串拼接，只要能够引入 ' 或是 \ 就可以造成注入

#### 5.1 只要我们可以控制就有注入

5.2 因为所有的 value 都会进行 htmlspecialchars 函数的html实体编码 addslashes 函数转义所以利用value 注入不现实，那么我们找注入的关键点就是查看key了

嗯，可以看到这就很明了了，我的审计收集工作也正式算是做完了。

接下来就是正式审计了。

不过其实到这一步，基本上在看两眼搜索一下打打 debug 就可以确定漏洞了。

点击收藏 | 0 关注 | 1

[上一篇：新型网络钓鱼活动事件分析](#) [下一篇：浅析PHP正则表达式的利用技巧](#)

1. 3 条回复



[SetObject](#) 2018-12-11 16:38:47

跟表哥学思路，坐等表哥更Xin!

0 回复Ta

---



[Thinker](#) 2018-12-15 22:50:24

老哥你这个是什么插件呀？

0 回复Ta

---



[phpoop](#) 2018-12-26 15:59:27

[@z66fr\\*\\*\\*\\*](#) vs code

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

[热门节点](#)

---

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)