

前言


今天看到一篇文章分析齐博CMS注入的文章：[齐博CMS激活验证处SQL注入](#)，得空分析了一下，总体感觉漏洞利用比较鸡肋。

漏洞分析

实际上齐博CMS是有对变量进行过滤的，但是本次注入点就是利用程序自带的编码，使得这些过滤形同虚设。


注入点在 inc/class.user.php 文件中的 get_passport 方法，可以清晰的看到SQL语句进行了变量拼接。

```
1 // inc/class.user.php
2 // 仅获取用户通行证的邮箱密码信息
3 function get_passport($value,$type='id') {
4     $sql = $type=='id' ? "uid='$value'" : "username='$value'";
5     $rs = $this->db_passport->get_one("SELECT * FROM {$this->memberTable} WHERE $sql");
6     return $rs;
7 }
```

 先知社区

同个文件的 get_allInfo 方法调用了 get_passport。

```
1 // inc/class.user.php
2 function get_allInfo($value,$type='id'){
3     global $webdb;
4     $array1=$this->get_passport($value,$type);
5     if(!$array1){
6         return ;
7     }
8     .....
9 }
```

 先知社区

变量加密处的位置在 do/activate.php 文件，代码如下：

```

1 // do/activate.php
2 <?php
3 require_once("global.php");
4 .....
5 elseif($job=='activate')
6 {
7     list($username,$password)=explode("\t",mymd5($md5_id,'DE'));
8     $rs=$userDB->get_allInfo($username,'name');
9
10
11     if($rs&&$rs[password]==$password)
12     {
13         $db->query("UPDATE {pre}memberdata SET `yz`='1' WHERE uid='$rs[uid]'");
14         refreshto("login.php","恭喜你, 你的帐号“{$username}”激活成功, 请立即登录, 体验会员特有的功能!",10);
15     }
16     else
17     {
18         showerr("帐号激活失败!");
19     }
20 }
21
22 if($username){
23     $rs=$userDB->get_allInfo($username,'name');
24     $email=$rs[email];
25 }
26 ?>

```



可以看到上图 第7行 代码, 将经过 mymd5 函数解密后的数据直接赋值给 \$username 和 \$password 两个变量, 并带入数据库查询。解密后的数据没有经过处理, 这是导致发生SQL注入的关键。我们可以看看 mymd5 函数的代码。

```

1 // inc/function.inc.php
2 function mymd5($string,$action="EN",$rand=''){ //字符串加密和解密
3     global $webdb;
4     if($action=="DE"){//处理+号在URL传递过程中会异常
5         $string = str_replace('QIBO|ADD','+', $string);
6     }
7     $secret_string = $webdb[mymd5].$rand.'5*j,.^&?.%#@!'; //绝密字符串, 可以任意设定
8     if(!is_string($string)){
9         $string=strval($string);
10    }
11    if($string=="") return "";
12    if($action=="EN") $md5code=substr(md5($string),8,10);
13    else{
14        $md5code=substr($string,-10);
15        $string=substr($string,0,strlen($string)-10);
16    }
17    // $key = md5($md5code.$_SERVER["HTTP_USER_AGENT"].$secret_string);
18    $key = md5($md5code.$secret_string);
19    $string = ($action=="EN"?$string:base64_decode($string));
20    $len = strlen($key);
21    $code = "";
22    for($i=0; $i<strlen($string); $i++){
23        $k = $i%$len;
24        $code .= $string[$i]^$key[$k];
25    }
26    $code = ($action == "DE" ? (substr(md5($code),8,10)==$md5code?$code:NULL) : base64_encode($code)."$md5code");
27    if($action=="EN"){//处理+号在URL传递过程中会异常
28        $code = str_replace('+','QIBO|ADD',$code);
29    }
30    return $code;
31 }

```



可以看到这个函数包含了加密与解密。然而要想利用这个注入点, 我们需要知道 \$webdb[mymd5] 的值, 而这个变量的值在每个网站搭建时会有一个初始值, 且都不一样, 这也是这个漏洞的鸡肋之处。我们可以登录后台查看到该变量对应的值。

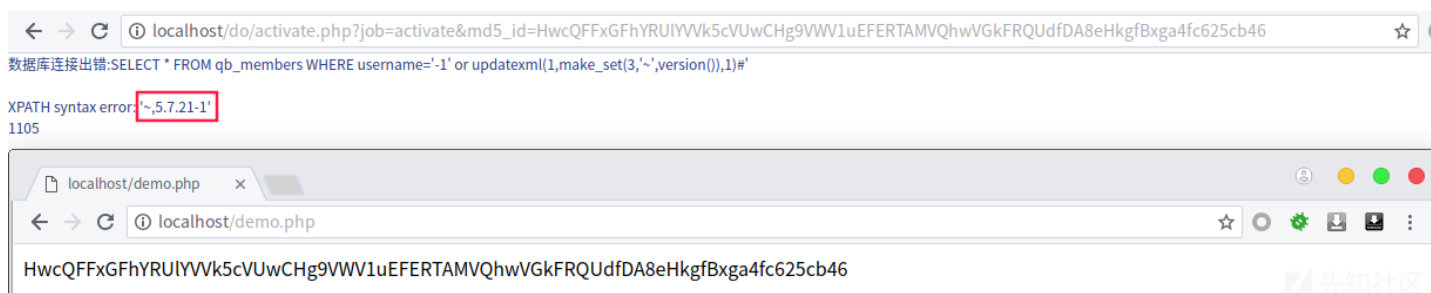


漏洞利用

要想利用这个漏洞，我们只需要知道 \$webdb[mymd5] 的值，并将相关函数抽取出来，加密我们的 payload 即可。具体如下：



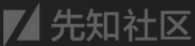
然后利用生成的加密字符串构造如下payload，即可成功注入SQL语句：



结语

后面想试试通过前台获取 \$webdb[mymd5] 的值，或者说控制该值，发现都不是很容易，于是作罢。

```
→ html grep -Rni "\$webdb\[mymd5\]"
admin/template/center/config.htm:351:         <input type="text" size="60" name="webdb[mymd5]" value="\$webdb[mymd5]">
admin/template/blend/set.htm:80:         <input type="text" name="webdb[mymd5]" value="\$webdb[mymd5]" size="50"
>
inc/function.inc.php:607:     \$secret_string = \$webdb[mymd5].\$rand.'5*j,.^&?.%#@!'; //绝密字符串,可以任意设定
demo.php:3:     \$webdb[mymd5]='atbbm5jbma';
demo.php:7:     \$secret_string = \$webdb[mymd5].\$rand.'5*j,.^&?.%#@!'; //绝密字符串,可以任意设定
do/activate.php:17:     if(!\$webdb[mymd5])
do/activate.php:19:         \$webdb[mymd5]=rand(10);
do/activate.php:20:         \$db->query("REPLACE INTO {$pre}config (`c_key`,`c_value`) VALUES ('mymd5','\$webdb[mymd5]')");
do/activate.php:21:         write_file(R00T_PATH."data/config.php","\$webdb['mymd5']='\$webdb[mymd5]';",'a')
;
do/sendpwd.php:14:     if(!\$webdb[mymd5])
do/sendpwd.php:16:         \$webdb[mymd5]=rand(10);
do/sendpwd.php:17:         \$db->query("REPLACE INTO {$pre}config (`c_key`,`c_value`) VALUES ('mymd5','\$webdb[mymd5]')");
do/sendpwd.php:18:         write_file(R00T_PATH."data/config.php","\$webdb['mymd5']='\$webdb[mymd5]';",'a')
;
→ html
```



点击收藏 | 0 关注 | 1

[上一篇：Java沙箱逃逸走过的二十个春秋（五）](#) [下一篇：Fileless恶意软件检测](#)

1. 5 条回复



[sera](#) 2018-10-06 21:35:15

这种需要全局变量覆盖的是不是都算鸡肋漏洞

0 回复Ta



[mochazz](#) 2018-10-06 22:01:13

[@sera](#) 不是说全局变量覆盖的漏洞是鸡肋漏洞，而是在这个漏洞中，想通过前台获取 \$webdb[mymd5] 变量的值比较困难，导致漏洞利用难度变大，影响变小，所以说它鸡肋。如果能从前台获取到 \$webdb[mymd5] 变量的值，那漏洞影响就不一样了。

0 回复Ta



[sera](#) 2018-10-06 22:53:36

[@mochazz](#) emmmm表哥好，其实是因为我昨天第一次做审计，发现一个cms的漏洞利用跟这个类似，都是解码后或反序列化后可以绕过过滤..但是需要一个类似于这个的\$webdb[mymd5]一样的值 /doge
。。它也是install的时候初始化的...然后没找到前台获取它的方法，只能靠register_global=on的时候可以设置这个值来绕过....

0 回复Ta



[Bojack](#) 2018-10-10 09:36:08

[@mochazz](#) 加密的key是从后台或者配置文件中获取的话，好像挺多类似的框架都是这样做的吧，不过拼接方面确实不应该。

0 回复Ta



[mochazz](#) 2018-10-14 09:47:41

[@Bojack](#) 这可 \$webdb[mymd5] 可以爆破出来，参考[QiboCMS从SQL注入到getshell](#)

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](#)