

# 实例分析

本次实例分析选择[dm201710](#)中的 **sql注入漏洞** 来进行分析。首先，漏洞通告来自cnvd。

## DM企业建站系统v201710存在SQL注入漏洞

★ 关注(0)

CNVD-ID	CNVD-2017-34212
公开日期	2017-12-07
危害级别	高 (AV:N/AC:L/Au:N/C:C/I:N/A:N)
影响产品	DM企业建站系统 DM建站系统 v201710
漏洞描述	DM建站系统是由php+mysql开发的一套专门用于中小企业网站建设的开源cms。  DM企业建站系统v201710登录处存在SQL注入漏洞，攻击者可利用该漏洞获取数据库敏感信息。
参考链接	
漏洞解决方案	厂商尚未提供漏洞修复方案，请关注厂商主页及时更新： <a href="http://www.demososo.com/">http://www.demososo.com/</a>

从漏洞通告中可以发现一些有用的信息，漏洞位置在登陆处，搭建的时候提示后台登陆口位置在 **admindm-yourname/g.php** 文件中，打开这个问题，发现重定向到 **admindm-yournamemod\_common/login.php** 文件中，所以漏洞触发点应该就在这个文件中。

```
<?php
Header( string: "Location: mod_common/login.php");
exit;
//登录文件在mod_common/login.php
?>
```

打开 **admindm-yournamemod\_common/login.php** 这个文件，一眼就看到漏洞位置，截取部分相关代码。

```

1 if($act=='login'){
2
3 $user= @htmlentities(trim($_POST['user']));
4 $ps= @htmlentities(trim($_POST['password']));
5
6
7 if(strlen($user)<2 or strlen($ps)<2){
8     alert('字符不够 sorry,user need more long');    jump($jumpv);
9 }
10
11 require_once WEB_ROOT.'component/dm-config/mysql.php';
12 // $salt = '00';is in config.php
13 $pscrypt= crypt($ps, $salt);
14 //echo $pscrypt;
15 $ss_P="select * from ".TABLE_USER." where email='$user' and
16 ps='$pscrypt' order by id desc limit 1";
17 // echo $ss_P;exit;
18 if(getnum($ss_P)>0){
19     $row=getrow($ss_P);
20     $userid=$row['id'];

```

第15行 很明显存在sql注入漏洞，通过拼接的方式直接插入到select语句中。第15行 中的 `$user` 变量是通过post方式页面提交上来，内容可控。但是这里的 第3行 代码调用 `htmlentities` 函数针对post提交上来的值进行处理。

跟进这个 `htmlentities` 函数，函数位置在 `component/dm-config/global.common.php` 文件中，截取部分相关代码。

```

1 function htmlentities($v){
2     return htmlentities(trim($v),ENT_NOQUOTES,"utf-8");
3 }

```

这个函数是调用 `htmlentities` 函数针对输入的数据进行处理。前面我们已经介绍过了这个函数的用法，这里这个函数的可选参数是 `ENT_NOQUOTES`，也就是说两种引号都不转换。下面我们来看个小例子：



```

<?php
$query = $_GET['query'];
echo htmlentities($query, quote_style: ENT_NOQUOTES, charset: "utf-8");
?>

```

view-source:http://127.0.0.1/index.php?query=<a href='http://www.baidu.com'>"test"</a>

```

1 &lt;a href='http://www.baidu.com'&gt;"test"&lt;/a&gt;

```

这里我猜测开发者应该是考虑到了xss的问题，但是由于 `htmlentities` 这个函数选择的参数出现了偏差，导致这里我们可以引入单引号造成注入的问题。

我们看看最新版是怎么修复，使用beyond compare对比两个版本代码的差别。

```
}
return $return_string;
} //end func

function htmlentitiesesdm($v){
    return htmlentities(trim($v), ENT_NOQUOTES, "utf-8");
} //end func
```

```
}
return $return_string;
} //end func

function htmlentitiesdm($v){
    return htmlentities(trim($v), ENT_QUOTES, "utf-8");
} //end func
```

新版修复的时候将可选参数修改为 **ENT\_QUOTES**，这个参数的作用就是过滤单引号加双引号，我们来看看下面这个例子，就很容易明白了这个参数的作用了。

```
<?php
$query = $_GET['query'];
echo htmlentities($query, quote_style: ENT_QUOTES, charset: "utf-8");
```

view-source:http://127.0.0.1/index.php?query=<a href='http://www.baidu.com'>"test" :/a>

```
1 &lt;a href='http://www.baidu.com'>"test" :/a>
```

## 漏洞验证

这里因为没有回显，所以是盲注，下面是验证截图。

```
POST
/dm201710//admindm-yourname/mod_common/login.php?act=login
HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:47.0)
Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/dm201710//admindm-yourname/mod_common/login.php
Cookie: UM_distinctid=164fac63977476-09d478c1a4e2108-48576f-13c680-164fac63978714; CNZZDATA1256279252=1392133828-1533211776-%7C1533211776
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 91

user=admin' AND SLEEP(5)--
```

```
HTTP/1.1 200 OK
Server: nginx/1.12.1
Date: Thu, 02 Aug 2018 13:55:28 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/5.5.38
Set-Cookie: isadmin=n; expires=Thu, 02-Aug-2018 23:55:28 GMT;
Max-Age=36000; path=/
Content-Length: 230

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /><script>window.alert("用户名或密码不对！sorry,user or password is incorrect")</script><script LANGUAGE='javascript'>window.location='login.php'</script>
```

0 matches

499 bytes | 5,012 millis

## 漏洞修复

针对 **htmlentities** 这个函数，我们建议大家在使用的時候，尽量加上可选参数，并且选择 **ENT\_QUOTES** 参数。

```

1 <?php
2 $query = $_GET['query'];
3 //echo "<a href='/images/size.php?' .
4 //    htmlentities($query) . "'>link</a>";    //未修复前
5 echo "<a href='/images/size.php?' .
6     htmlentities($query,ENT_QUOTES) . "'>link</a>";    //修复后
7 ?>

```

我们看看对比的效果

Load URL

Split URL

Execute

http://127.0.0.1/index.php?query=a'onclick%3dalert(1)%2f%2f=c

☐ Enable Post data
☐ Enable Referrer

1 <a href='<u>/images/size.php?a&#039;onclick=alert(1)//=c</u>'>link</a>