实例分析

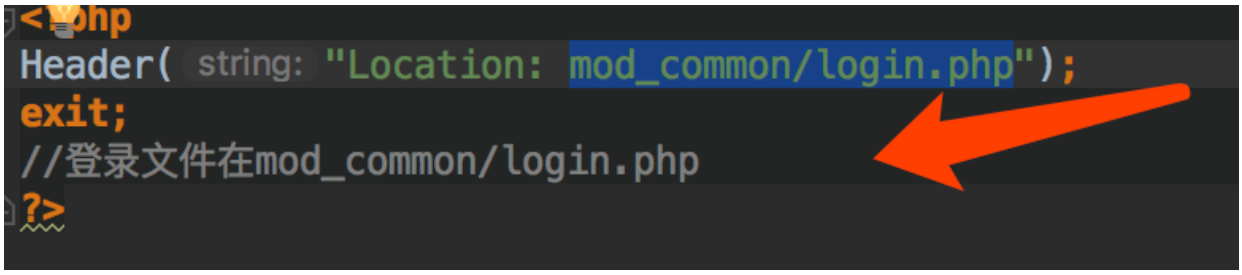
本次实例分析选择dm201710中的 **sql注入漏洞** 来进行分析。首先，漏洞通告来自cnvd。

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： http://www.demososo.com/

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。



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。

```php
if($act=='login'){

$user= @htmlentitiesdm(trim($_POST['user']));
$ps= @htmlentitiesdm(trim($_POST['password']));


if(strlen($user)<2 or strlen($ps)<2){
    alert('字符不够 sorry,user need more long');    jump($jumpv);
}

require_once WEB_ROOT.'component/dm-config/mysql.php';
     // $salt = '00';is in config.php
  $pscrypt= crypt($ps, $salt);
     //echo $pscrypt;
 $ss_P="select * from ".TABLE_USER."  where  email='$user' and
 ps='$pscrypt'  order by id desc limit 1";
    // echo $ss_P;exit;
        if(getnum($ss_P)>0){
                    $row=getrow($ss_P);
                    $userid=$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 **htmlentitiesdm** 函数针对post提交上来的值进行处理。

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

```php
function htmlentitiesdm($v){
   return htmlentities(trim($v),ENT_NOQUOTES,"utf-8");
}
```

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


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

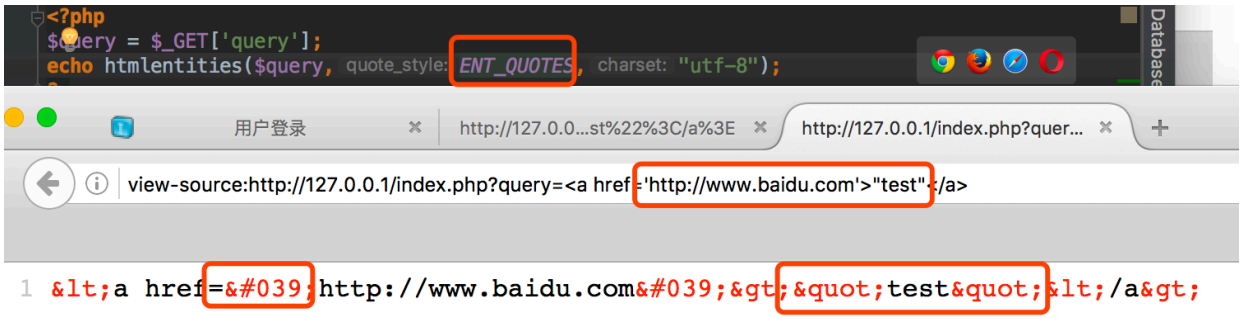
这里我猜测开发者应该是考虑到了xss的问题，但是由于 **htmlentities** 这个函数选择的参数出现了偏差，导致这里我们可以引入单引号造成注入的问题。

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

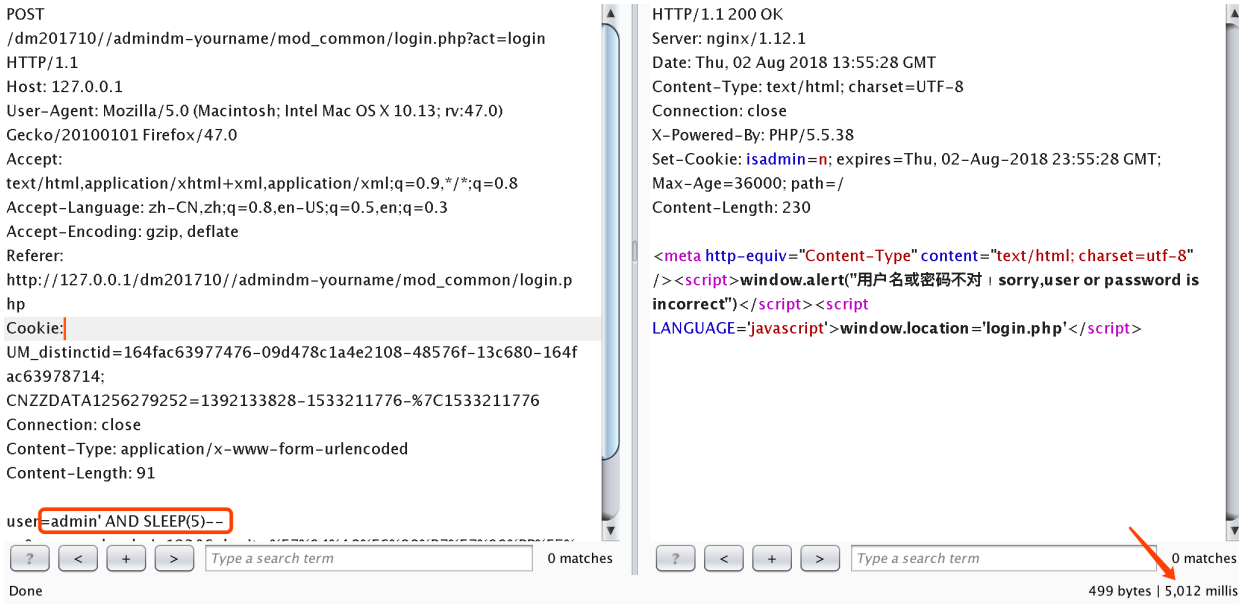
```
   }
   return $return_string;
}//end func

function htmlentitdm($v){
   return htmlentities(trim($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。



漏洞验证

这里因为没有回显，所以是盲注，下面是验证截图。



漏洞修复

针对 **htmlentities** 这个函数，我们建议大家在使用的时候，尽量加上可选参数，并且选择 **ENT_QUOTES** 参数。

```php
<?php
$query = $_GET['query'];
//echo "<a href='/images/size.php?" .
//    htmlentities($query) . "'>link</a>";      //未修复前
echo "<a href='/images/size.php?" .
    htmlentities($query,ENT_QUOTES) . "'>link</a>";  //修复后
?>
```

我们看看对比的效果