黑客派

常见漏洞防御 之 防 SQL 注入的三种方式

作者：zml2015

原文链接：https://hacpai.com/article/1488852456737

来源网站：黑客派

看以下三种SQL语句
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></script>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342" data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></ins>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<pre class="brush: java">String sql1 = "select * from user where username = '"+username+"' and password = '"+password+"'";
<pre><code class="highlight-chroma">   String sql2 ="select * from user where username = username and password = :password";

</code></pre>

推荐写法是sql2、sql3使用的别名或者通配符的形式进行传值，可以避免SQL注入，sql1就有了见的典型SQL注入问题，假设username随便传个值，比如'张三'，传入的password的值为
 ' or 1 = 1' 则 拼接的SQL语句成了
<pre class="brush: sql">select * from user where username ='张三' and password = '' or 1 = 1 ''</pre>
很明显无论前面的and为真还是假，只要or后面为真，则就可使查询为true，导致SQL注入 所以可能避免字符串拼接的形式进行SQL传值，但有些场景可能必须采用字符串拼接的形式，比如Mybati在按要求排序时：
<pre class="brush: sql">order by #{order} #{sort}</pre>
用#{}的形式取值就相当于字符串拼接，但是因为排序DESC、AESC这种是SQL关键字，不能进转义，所以就必须采用字符串拼接的形式了
如果采用这种形式还要避免SQL注入的话，就需要对传入的值进行SQL过滤，来避免SQL注入
那么OWASP提供了一个防御sql注入的Esapi包，这个包中的encodeForSQL方法能对sq注入进行很好的防御。
<pre class="brush: java">//防止Oracle注入
ESAPI.encoder().encodeForSQL(new OracleCodec(),queryparam)
//防止mysql注入
ESAPI.encoder().encodeForSQL(new MySQLCodec(Mode.STANDARD),queryparam) //Mode.SANDARK为标准的防注入方式，mysql一般用使用的是这个方式
//防止DB2注入
ESAPI.encoder().encodeForSQL(new DB2Codec(),queryparam) </pre>
使用不同的数据库，使用的过滤方法不同
下面我们就用MySQL为例字分析encodeForSQL函数做了什防御。具体函数过程就不跟踪了，直接分析最后调用了哪个方法。根据代码可知最后调用的是encodeharacter方法。

<pre class="brush: java">public String encodeCharacter(char[] immune, Character c) {
    char ch = c.charValue();
<pre><code class="highlight-chroma">   // check for immune characters
  if (containsCharacter(ch, immune)) {
    return ""+ch;
  }

} not found render function for node [type=NodeHTMLEntity, Tokens=<]not found render unction for node [type=NodeHTMLEntity, Tokens=<]/pre
ot found render function for node [type=NodeHTMLEntity, Tokens=>]not found render funct on for node [type=NodeHTMLEntity, Tokens=>]

```
</code></pre>

<p><span> </span></p>
<p><span><span>上述方法中containsCharacter函数是不进行验证的字符串白名单，Codec.get exForNonAlphanumeric函数查找字符传中是否有16进制，没有返回空值。</span></span></p>
<p> </p>
<p><span><span>而encodeCharacterANSI和encodeCharacterMySQL才是防御的重点，我们一下这两个函数的不同，如果选择的我们选择是Mode.ANSi模式，则字符串则进入下面的函数，可看到这个函数对单撇号和双撇号进行了转义。</span></span></p>
<p> </p>
<pre class="brush: java">private String encodeCharacterANSI( Character c ) {
    if ( c == '\'' )
        return "\'\'";
    if ( c == '\"' )
        return "";
    return ""+c;
} </pre>
<p><span> </span></p>
<p><span><span>如果选择的是Mode.STANDARD模式，则字符串则进入下面的函数，可以看到个函数对单撇号和双撇号、百分号、反斜线等更多的符号进行了转换，所以使用时推荐使用标准模式</span></span></p>
<p> </p>
<pre class="brush: java">private String encodeCharacterMySQL( Character c ) {
    char ch = c.charValue();
    if ( ch == 0x00 ) return "\\0";
    if ( ch == 0x08 ) return "\\b";
    if ( ch == 0x09 ) return "\\t";
    if ( ch == 0x0a ) return "\\n";
    if ( ch == 0x0d ) return "\\r";
    if ( ch == 0x1a ) return "\\Z";
    if ( ch == 0x22 ) return "\\\"";
    if ( ch == 0x25 ) return "\\%";
    if ( ch == 0x27 ) return "\\'";
    if ( ch == 0x5c ) return "\\\\";
    if ( ch == 0x5f ) return "\\_";
    return "\\" + c;
} </pre>
<p><span> </span></p>
<p>写个单元测试：</p>
<pre class="brush: java">@org.junit.Test
    public void testESAPI() {
        String username = "zhangsan";
        String password = "' or 1=1'";
<pre><code class="highlight-chroma">    String sql1 = "select * from user where username '"+username+"' and password = '"+password+"'";


</code></pre>

<p>//        String sql2 ="select * from user where username = :username and password = :pa
```

sword";
//   String sql3 = "select * from user where username = ? and password = ?";
System.out.println("过滤前： "+sql1);
<pre><code class="highlight-chroma">   username = ESAPI.encoder().encodeForSQL(new
ySQLCodec((Mode.STANDARD)), username);
    password = ESAPI.encoder().encodeForSQL(new MySQLCodec((Mode.STANDARD)), passw
rd);
    sql1 = "select * from user where username = '"+username+"' and password = '"+passwor
+"'";

}not found render function for node [type=NodeHTMLEntity, Tokens=<]not found render fu
ction for node [type=NodeHTMLEntity, Tokens=<]/pre
ot found render function for node [type=NodeHTMLEntity, Tokens=>]not found render funct
on for node [type=NodeHTMLEntity, Tokens=>]
</code></pre>

<a title="SQL注入" href="https://link.hacpai.com/forward?goto=http%3A%2F%2Foexaoy
pt.bkt.clouddn.com%2F17-3-7%2F65534187-file_1488853885373_5b30.png" class="fancybox"
target="_blank" rel="nofollow ugc"><img title="SQL注入" src="https://static.hacpai.com/ima
es/img-loading.svg" alt="SQL注入" width="1233" height="578" data-src="http://oexaoy5pt.b
t.clouddn.com/17-3-7/65534187-file_1488853885373_5b30.png">

我们介绍了利用绑定变量、通配符和利用esapi三种方式对sql注入进行防御，
的建议是尽量使用绑定变量或者通配符的是形式进行防注入，安全性能都比较好，如果不得不使用字
串拼接的形式，则使用esapi进行sql过滤
最后给个esapi的maven支持形式
<pre class="brush: xml">   <dependency>
        <groupId>org.owasp.esapi</groupId>
        <artifactId>esapi</artifactId>
        <version>2.1.0.1</version>
    </dependency></pre>

</pre></pre></pre>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>