sql注入是比较常见的网络攻击方式之一，它不是利用操作系统的bug来实现攻击，而是针对程序员编程时的疏忽，通过sql语句，实现无账号登录，甚至篡改数据库。

sql注入攻击的总体思路

1.寻找sql注入的位置
2.判断服务器类型和后台数据库类型
3.针对不同的服务器和数据库特点进行sql注入攻击。

sql注入攻击实例

比如一个登录界面，要求输入用户名和密码：
可以这样输入实现免账号登录：
用户名：‘or 1 = 1 -
密码：
点登录，如若没有做特殊处理，那么这个非法用户就很得意的登录进去了。
能登录的原理分析：

```
String sql = "select * from user_table where username = '" + userName + "' and password = '" + password + "'";
```

当输入了上面的用户名和密码，上面的sql语句变成为：

```
select * from user_table where username = '' or 1 = 1 -- and password = ''
```

分析sql语句：
条件后面的username = '' or 1 = 1 用户名等于 '' 或 1 = 1 那么这个条件一定会成功；然后后面加两个-，这意味着注释，它将后面的语句注释，让他们不起作用，这样语句永远都能正确执行，用户轻易骗过系统，获取合法身份。

如果是执行下面的语句

```
SELECT * FROM user_table WHERE
username='' ;DROP DATABASE (DB Name) --' and password=''
```

….其后果可想而知…

怎么防止注入：
1.采用preparedStatement
采用预编译语句集，它内置了处理sql注入的能力，只要使用它的setXXX方法传值即可
使用好处：

1.代码的可读性和可维护性。
2.PreparedStatement尽最大可能提高性能。
3.最重要的一点是极大地提高了安全性。

原理：
sql注入只对sql语句的准备过程有破坏作用
而prepredStatement已经准备好了，执行阶段只是把输入串作为数据处理
而不在对sql语句进行解析，准备，因此也就避免了sql注入问题。

2.使用正则表达式过滤传入的参数
要引入的包：
import java.util.regex.*;
正则表达式：
private String CHECKSQL = “^(.+)\\sand\\s(.+)|(.+)\\sor(.+)\\s$”;
判断是否匹配：
Pattern.matches(CHECKSQL,targerStr);
下面是具体的正则表达式：
检测SQL meta-characters的正则表达式 ：
/(\%27)|(\’)|(\-\-)|(\%23)|(#)/ix
修正检测SQL meta-characters的正则表达式 ：/((\%3D)|(=))[^\n]*((\%27)|(\’)|(\-\-)|(\%3B)|(:))/i
典型的SQL 注入攻击的正则表达式 ：/\w*((\%27)|(\’))((\%6F)|o|(\%4F))((\%72)|r|(\%52))/ix
检测SQL注入，UNION查询关键字的正则表达式 ：/((\%27)|(\’))union/ix(\%27)|(\’)
检测MS SQL Server SQL注入攻击的正则表达式：
/exec(\s|\+)+(s|x)p\w+/ix
等等…..

3.字符串过滤
比较通用的一个方法：
（||之间的参数可以根据自己程序的需要添加）

```
public static boolean sql_inj(String str)
{
String inj_str = "'|and|exec|insert|select|delete|update|
count|*|%|chr|mid|master|truncate|char|declare|;|or|-|+|,";
String inj_stra[] = split(inj_str,"|");
for (int i=0 ; i &lt; inj_stra.length ; i++ )
{
if (str.indexOf(inj_stra[i])&gt;=0)
{
return true;
}
}
return false;
}
```

4.也可以在前端进行字符的判断。