2020/5/7-笔记：SQL注入攻击的原理以及怎样避免SQL注入

所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意）的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。

怎么防止SQL注入，使用存储过程来执行所有的查询；检查用户输入的合法性；将用户的登录名、密码等数据加密保存。

SQL注入攻击的原理：

假如一个网站的登录功能的SQL语句字符串是这样写的：

"select * from WEBUSERS where ID = '"+ID+"' and PASSWORD = '"+passWord+"'"

如果用户在ID输入框中随便输入数字，例如：12345678，在密码输入框中输入这样一串字符：'or 1 = '1

那么这个SQL查询语句就会变成这样：

select * from WEBUSERS where ID = '12345678' and PASSWORD = ''or 1 = '1'

无论ID输入什么，这条SQL语句是永远都有返回结果的，这样即使不使用账号和密码也可以登录该网站，甚至可以填入一些恶意的SQL代码，给网站的数据库造成威胁。

SQL注入攻击的解决办法：

可以使用 ‘？’ 占位符创建SQL语句再用预处理PreparedStatement的方法处理SQL语句。

例如：

原DAO的代码：


```
1 public int dealLogin_1(String ID, String passWord) {//字符串拼接的方法
2         ConnectOracleDBUtil dbUtil = new ConnectOracleDBUtil();
3         // 调用方法加载数据库驱动
4         dbUtil.ConnectOracle();
5         // 调用方法创建数据库连接
6         Connection stmt = ConnectOracleDBUtil.getConnection();
7         // 创建查询语句
8         String sql0 = "select * from WEBUSERS where ID = '"+ID+"' and PASSWORD = '"+passWord+"'";
9         Statement stmt0;
10        try {
11            // 发送语句并执行
12            stmt0 = stmt.createStatement();
```

```
            // 返回结果
            ResultSet rs0 = stmt0.executeQuery(sql0);
            //对结果进行处理
            if(rs0.next()) {
                System.out.println("账号和密码匹配!");
                return 1;
            }else {
                System.out.println("账号和密码不匹配！");
                return 2;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //调用方法关闭与数据库的连接
            dbUtil.closed();
        }
        return 0;
    }
```

使用预处理PreparedStatement的DAO代码：

```
public int dealLogin_2(String ID, String passWord) {//占位符预处理的方法
        ConnectOracleDBUtil dbUtil = new ConnectOracleDBUtil();
        // 调用方法加载数据库驱动
        dbUtil.ConnectOracle();
        // 调用方法创建数据库连接
        Connection stmt = ConnectOracleDBUtil.getConnection();
        // 调用方法创建查询语句
        String sql1 = "select * from WEBUSERS where ID = ? and PASSWORD = ?";
        // 发送语句并执行
        PreparedStatement stmt1 = null;
        try {
            stmt1 = stmt.prepareStatement(sql1);
            stmt1.setString(1, ID);
            stmt1.setString(2, passWord);
            // 返回结果
            ResultSet rs1 = stmt1.executeQuery();
            if(rs1.next()) {
                System.out.println("账号和密码匹配!");
                return 1;
            }else {
                System.out.println("账号和密码不匹配！");
                return 2;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //调用方法关闭与数据库的连接
            dbUtil.closed();
        }
        return 0;
    }
```

预处理prepareStatement避免SQL注入漏洞的原理：

因为SQL语句在程序运行前已经进行了预编译，在程序运行时第一次操作数据库之前，SQL语句已经被数据库分析，编译和优化，对应的执行计划也会缓存下来并允许数据库已参数化的形式进行查询，当运行时动态地把参数传给PreprareStatement时，即使参数里有敏感字符如 or '1=1'也数据库会作为一个参数一个字段的属性值来处理而不会作为一个SQL指令，如此，就起到了SQL注入的作用了！

先来说说，什么是java中的Statement：Statement是java执行数据库操作的一个重要方法，用于在已经建立数据库连接的基础上，向数据库发送要执行的SQL语句。具体步骤：

1.首先导入java.sql.*；这个包。

2.然后加载驱动，创建连接，得到Connection接口的的实现对象，比如对象名叫做conn。

3.然后再用conn对象去创建Statement的实例，方法是：Statement stmt = conn.creatStatement("SQL语句字符串");

Statement 对象用于将 SQL 语句发送到数据库中。实际上有三种 Statement 对象，它们都作为在给定连接上执行 SQL语句的包容器：Statement、PreparedStatement（它从 Statement 继承而来）和CallableStatement（它从 PreparedStatement 继承而来）。它们都专用于发送特定类型的 SQL 语句：Statement 对象用于执行不带参数的简单 SQL 语句；PreparedStatement 对象用于执行带或不带参数的预编译 SQL 语句；CallableStatement 对象用于执行对数据库已存储过程的调用。

综上所述，总结如下：Statement每次执行sql语句，数据库都要执行sql语句的编译，最好用于仅执行一次查询并返回结果的情形，效率高于PreparedStatement.但存在sql注入风险。PreparedStatement是预编译执行的。在执行可变参数的一条SQL时，PreparedStatement要比Statement的效率高，因为DBMS预编译一条SQL当然会比多次编译一条SQL的效率高。安全性更好，有效防止SQL注入的问题。对于多次重复执行的语句，使用Prepared

Statement效率会更高一点。执行SQL语句是可以带参数的，并支持批量执行SQL。由于采用了Cache机制，则预编译的语句，就会放在Cache中，下次执行相同的SQL语句时，则可以直接从Cache中取出来。

```
1 PreparedStatement pstmt  =  con.prepareStatement("UPDATE EMPLOYEES  SET name= ? WHERE ID = ?");
2 pstmt.setString(1, "李四");
3 pstmt.setInt(2, 1);
4 pstmt. executeUpdate();
```

那么CallableStatement扩展了PreparedStatement的接口，用来调用存储过程，它提供了对于输入和输出参数的支持，CallableStatement 接口还有对 PreparedStatement 接口提供的输入参数的sql查询的支持。

PreparedStatement: 数据库会对sql语句进行预编译，下次执行相同的sql语句时，数据库端不会再进行预编译了，而直接用数据库的缓冲区，提高数据访问的效率（但尽量采用使用？号的方式传递参数），如果sql语句只执行一次，以后不再复用。 从安全性上来看，PreparedStatement是通过？来传递参数的，避免了拼sql而出现sql注入的问题，所以安全性较好。

在开发中，推荐使用 PreparedStatement。