一、SQL注入漏洞

1.1什么是SQL注入?

SQL注入：是指web应用程序对用户输入数据的合法性没有判断或过滤不严，攻击者可以在web应用程序中事先定义好的查询语句的结尾上添加额外的SQL语句，在管理员不知情的情况下实现非法操作，以此来实现欺骗数据库服务器执行非授权的任意查询，从而进一步得到相应的数据信息。

简单来说就是，使用某些手段，在原来的SQL语句基础上，添加了一段SQL并执行。从而达到某一些不被管理员允许的目的。

一般使用SQL注入，主要是拿数据库里面的数据

那怎么去拿到数据库里面的数据?

1：获得库名

```
SELECT DATABASE();
SELECT SCHEMA_NAME FROM information_schema.SCHEMATA;
```

2：获得表名

```
SELECT * from information_schema.`TABLES` WHERE TABLE_SCHEMA='schooldb'
```

3：获得列名

```
SELECT * from information_schema.`COLUMNS` WHERE TABLE_SCHEMA='schooldb' and
TABLE_NAME ='userinfo'
```

4：获得表里的数据

```
SELECT userID,userName,userPass from userinfo
```

SQL语句准备好了，万事俱备只欠东风，接下来就要使用SQL注入的方式，将这些SQL语句注入到服务器。能不能直接输入这个sql语句呢？SELECT DATABASE(); 显然不能，原因很简单 ，登录的逻辑就是拿着用户名和密码到数据库里面进行比对，那肯定会执行一个这样的SQL语句

```
SELECT * from users WHERE username='SELECT DATABASE()' and ....;
```

这个SQL语句 铁定是要报错的，我们来分析一下这个SQL语句，我们真正想执行的SQL语句是 SELECT DATABASE()，换句话说SELECT DATABASE() 这个语句是不是要作为一个独立的SQL语句执行。

所以第一件事情就是把一条SQL语句拆分成两条SQL语句？ 怎么拆？ 加引号闭合 在加分号 加注

SELECT * from users WHERE username='';SELECT DATABASE() #' and ....;

这个SQL语句 执行还是会报错，原因是不管PHP 还JAVA 还是pyhon 都只能一次性执行一条SQL语句。

怎么在一条语句里面执行两个查询语句呢？ 是不是可以借助一个关键 union，

```
SELECT * from users WHERE username='' union SELECT DATABASE() #' and ....;
```

但是union关键字有一个限制 两条SQL语句查询的字段数要相同 所以说我们还要获得前一个sql语句 字段数。 怎么获得这个数据？ 可以借助关键字 order’ by

```
SELECT * from users WHERE username='' ORDER BY 1
```

```
SELECT * from userinfo WHERE userName= ''   union SELECT DATABASE(), 1,2 # and
userPass= ''
发现数据没拿到，但也没报错 ，显示1
```

```
SELECT * from userinfo WHERE userName= ''   union SELECT 1,DATABASE(),2 # and
userPass= ''
```

1：获得库名

```
SELECT * from userinfo WHERE userName= ''   union SELECT 1,DATABASE(),2 # and
userPass= ''
```

2：获得表名

发现有，三条数据，但页面上是不是，只能显示一个数据啊？ 怎么办 使用 GROUP_CONCAT 函数

```
SELECT * from userinfo WHERE userName= '
'   union SELECT 1, GROUP_CONCAT(table_Name),2 from information_schema.`TABLES`
WHERE TABLE_SCHEMA='schooldb' # and userPass= ''
```

3：获得列名

```
SELECT * from userinfo WHERE userName= '
'   union SELECT 1,GROUP_CONCAT(column_name) ,2 from information_schema.`COLUMNS`
WHERE TABLE_SCHEMA='schooldb' and TABLE_NAME ='userinfo' #and userPass= ''
```

4：获得数据

```
SELECT * from userinfo WHERE userName= '
'   union SELECT userID,userName,userpass FROM userinfo  #and userPass= ''
```

原因和解决办法

Statment不能防止sql注入

“+”号直接拼接参数（要溯源对参数进行过滤）

```
//登陆
public User login(Connection con,User user) throws Exception{
    User resultUser=null;
    String sql="select * from t_user where userName='"+user.getUserName()+"' and password='"+user.getPassword()+'
    java.sql.Statement stmt = con.createStatement();
    //Statement stmt=con.createStatement();
    ResultSet res = stmt.executeQuery(sql);
    if(res.next()){
        resultUser=new User();
        resultUser.setUserName(res.getString("userName"));
        resultUser.setPassword(res.getString("password"));
    }
    return resultUser;
}
```

解决办法

PreparedStatement预处理

```
public User login(Connection con,User user) throws Exception{
    User resultUser=null;
    String sql="select * from t_user where userName=? and password=?";
    PreparedStatement pstmt=con.prepareStatement(sql);
    pstmt.setString(1, user.getUserName());
    pstmt.setString(2, user.getPassword());
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()){
        resultUser=new User();
        resultUser.setUserName(rs.getString("userName"));
        resultUser.setPassword(rs.getString("password"));
    }
    return resultUser;
```

like：当预处理遇到like

```
public ResultSet gradeList(Connection con,PageBean pageBean,Grade grade)throws Exception{
    StringBuffer sb=new StringBuffer("select * from t_grade");
    if(grade!=null && StringUtil.isNotEmpty(grade.getGradeName())){
        sb.append(" and gradeName like '%"+grade.getGradeName()+"%'");

    }
    if(pageBean!=null){
        sb.append(" limit "+pageBean.getStart()+","+pageBean.getRows());

    }
    PreparedStatement pstmt=con.prepareStatement(sb.toString().replaceFirst("and", "where"));
        return pstmt.executeQuery();
}
```

预处理遇到like—正确处理方法

```
public ResultSet gradeList(Connection con,PageBean pageBean,Grade grade)throws Exception{
    StringBuffer sb=new StringBuffer("select * from t_grade");
    int apd = 0;
    if(grade!=null && StringUtil.isNotEmpty(grade.getGradeName())){
        sb.append(" and gradeName like ?");
        apd += 1;
    }
    if(pageBean!=null){
        sb.append(" limit ?,?");
        apd += 2;
    }
    PreparedStatement pstmt=con.prepareStatement(sb.toString().replaceFirst("and", "where"));
    if(apd == 3){
        pstmt.setString(1, "%"+grade.getGradeName()+"%");
        pstmt.setLong(2, pageBean.getStart());
        pstmt.setLong(3, pageBean.getRows());
    }else if(apd == 2){
        pstmt.setLong(1, pageBean.getStart());
        pstmt.setLong(2, pageBean.getRows());
    }else if(apd == 1){
        pstmt.setString(1, "%"+grade.getGradeName()+"%");
    }

    return pstmt.executeQuery();
}
```

当预处理遇到in

不正确写法

```
public int gradeDelete(Connection con,String delIds)throws Exception{
    String sql="delete from t_grade where id in("+delIds+")";
    PreparedStatement pstmt=con.prepareStatement(sql);
    return pstmt.executeUpdate();
}
```

正确写法

```java
//安全的删除方法
public int gradeDelete(Connection con,String delIds)throws Exception{
    String num = "";
    String[] spl = delIds.split(",");
        for(int i=0;i<spl.length;i++){
            if(i==0){
                num += "?";
            }else{
                num += ",?";
            }
        }
        String sql = "delete from t_grade where id in("+num+")";
        PreparedStatement pstmt=con.prepareStatement(sql);
        try {
            for(int j=0;j<spl.length;j++){
                pstmt.setInt(j+1,Integer.parseInt(spl[j]));
            }
            return pstmt.executeUpdate();
        } catch (Exception e) {
            // TODO: handle exception
        }

    return 0;

}
```

5：SQL注入如何防止?

1、预编译

2、内容过滤

3、引号转义

4、错误信息

5、数据库权限

6、数据加密

7、应用防火墙

二、Mybatis

SQL注入-Mybatis

${}直接拼接　存在漏洞

```xml
    <select id="selectByPrimaryName" resultMap="BaseResultMap" parameterType="java.lang.String" >
    select
    <include refid="Base_Column_List" />
    from tb_admin
    where username = ${username}
</select>
```

#{} 预编译

```xml
<select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer"
    select
    <include refid="Base_Column_List" />
    from tb_admin
    where id = #{id,jdbcType=INTEGER}
</select>
```

SQL注入-Mybatis—like防注入

Mysql:

```
select * from t_user where name like concat('%', #{name}, '%')
```

Oracle:

```
select * from t_user where name like '%' || #{name} || '%'
```

Sql Server:

```
select * from t_user where name like '%' + #{name} + '%'
```

SQL注入-Mybatis—in防注入

```
<if test="paramBrands != null" >•and brand.brand_id in <foreach
collection="paramBrands" item="perBrand" open="(" close=")" separator=","> #
{perBrand.brandId}</foreach></if>
```