数据库实验-数据库编程

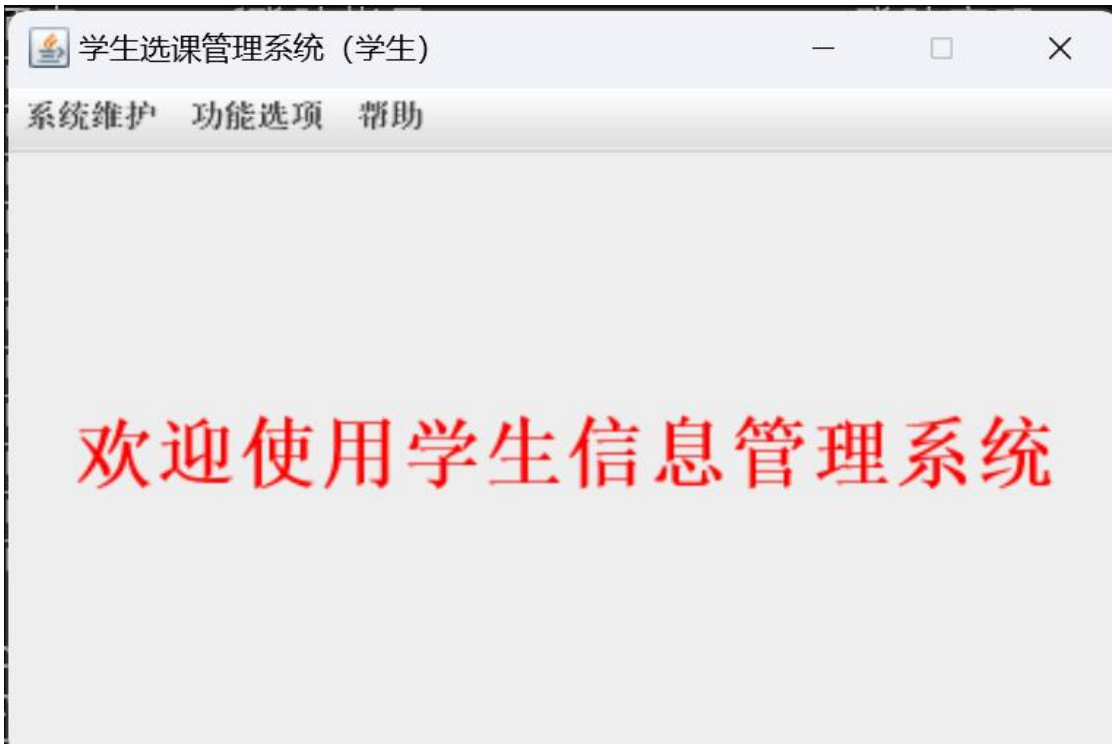
数据库小组：

组长：王鸣一　组员：章泽亮、孙溯阳

分工：实验操作：章泽亮、孙溯阳　　报告撰写：王鸣一

一、SQL 注入攻击

首先尝试运行并登录学生选课管理系统，



完成登录；

然后返回登陆页面尝试对该系统进行 SQL 注入攻击，有两种 SQL 攻击：

第一种攻击：

攻击逻辑：

SELECT * FROM 学生基本信息表 WHERE(学号='' OR '1'='1') -- ' AND 密码 ='xxxxx')

- '1'='1'使条件恒真，若系统未过滤输入，可能直接返回第一条学

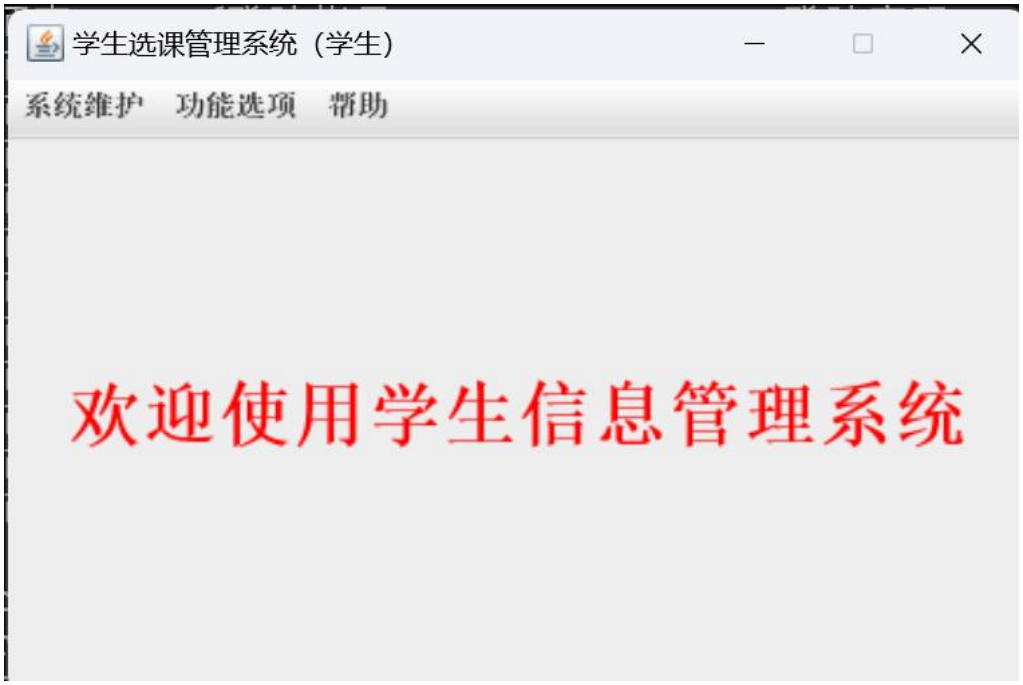
生记录，实现登录。

攻击过程：

在登录窗口页面，输入用户账号为“ ’ OR ’1’=’1’ ） -- ”，

任意输入登陆密码·····（任意字符），选择用户类型为“学生”

点击“确认”，登陆成功。



该攻击在登录界面输入注入代码使得系统执行被篡改的 SQL 语句，最终绕过验证，登陆成功。

第二种攻击：

攻击逻辑：

SELECT * FROM 学生基本信息表 WHERE(学号='' OR '1'='1' AND 密码 ='' OR '1'='1')

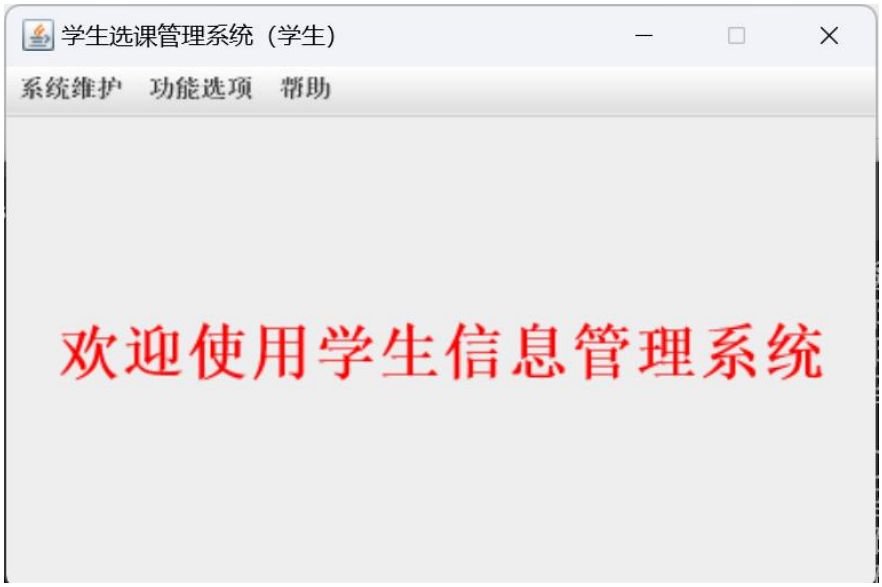
• 通过双重永真条件（'1'='1'）同时绕过学号和密码验证。

攻击过程：

在登录窗口页面，输入用户账号为“ ' OR '1'='1 ”，

输入登陆密码为“ ' OR '1'='1 ”，选择用户类型为“学生”

点击“确认”，登陆成功。



该攻击在同样在登录界面输入注入代码使得系统执行被篡改的 SQL 语句，最终绕过验证，登陆成功。

二、SQL 注入防御

1、防御方法：

使用 PreparedStatement 替代 Statement

改写 loginDispose 方法的代码，以防止 SQL 注入：

改写为：

```
private void loginDispose() {
    try {
        Class.forName(DataBaseInfo.drive);
        loginConnection                                    =
DriverManager.getConnection(DataBaseInfo.url,
DataBaseInfo.username, DataBaseInfo.password);
    } catch (ClassNotFoundException cnfex) {
        cnfex.printStackTrace();
        JOptionPane.showMessageDialog(LoginFrame.this,
cnfex,
            "学生选课管理系统", JOptionPane.WARNING_MESSAGE);
        System.exit(1);
    } catch (SQLException sqlex) {
```

```
        sqlex.printStackTrace();
        JOptionPane.showMessageDialog(LoginFrame.this,
            "无法连接到 SQL SERVER ，\n 请确认 SQL SERVER 是否运行
\n 或数据源设置是否正确！",
            "学生选课管理系统", JOptionPane.WARNING_MESSAGE);
        System.exit(1);
    }

    try {
        String loginUserName = myTextField.getText().trim();
        String          loginPassword          =          new
String(passwordField.getPassword());

        if (loginUserName.isEmpty()) {
            JOptionPane.showMessageDialog(LoginFrame.this,
                "用户名不能为空！", "登陆",
JOptionPane.WARNING_MESSAGE);
            return;
        }

        String loginQuery = "";
        PreparedStatement pstmt = null;


        if (selectedItem.equals("教师")) {

            loginQuery = "SELECT * FROM 教师表 WHERE 登陆帐号
= ? AND 登陆密码 = ?";

        } else if (selectedItem.equals("管理员")) {

            loginQuery = "SELECT * FROM 管理员 WHERE 用户名 = ?
AND 密码 = ?";

        } else { // 学生

            loginQuery = "SELECT * FROM 学生基本信息表 WHERE 学
号 = ? AND 密码 = ?";
```

```
        }

        pstmt                                                    =
loginConnection.prepareStatement(loginQuery);
        pstmt.setString(1, loginUserName);
        pstmt.setString(2, loginPassword);

        loginResultSet = pstmt.executeQuery();
        boolean Records = loginResultSet.next();

        if (!Records) {
            JOptionPane.showMessageDialog(LoginFrame.this,"
没有此用户或密码错误");
            return;
        } else {
            login = 1;
        }

        loginConnection.close();
    } catch (SQLException sqlex) {
        JOptionPane.showMessageDialog(LoginFrame.this,
sqlex,
            "学生选课管理系统", JOptionPane.WARNING_MESSAGE);
    }
}
```

2、改动重点说明：

改动项	原因
使用 PreparedStatement	防止 SQL 注入
? 占位符替代拼接字符串	避免将用户输入变成 SQL 语法
.setString()	将输入安全绑定为参数
.trim() 过滤前后空格	防止无意输入干扰验证

使用 PreparedStatement 代替 Statement，通过 Connection.prepareStatement(sql）预编译 SQL 模板，其中参数用

“？”占位符表示。（SQL 语句中的变量部分用 ？ 占位符表示，用户输入通过 setString() 方法绑定。）

彻底避免将用户输入直接拼接到 SQL 语句中，从根源上消除输入数据篡改 SQL 语法的可能性。

预编译的 SQL 模板在数据库中被固定为指令结构，用户输入通过后续绑定的方式传递，确保输入数据无法改变原 SQL 语义。即使输入中包含 ’、OR、-- 等特殊字符，数据库引擎也会将其视为普通字符串，而非可执行的 SQL 片段。

例如，输入 ’ OR ’1’=’1 会被转义为 \’ OR \’1\’=\’1，作为普通字符串匹配，而非逻辑条件。

3、再次 SQL 注入攻击试验（防御检验）

在登陆页面，输入用户账号为“ ’ OR ’1’=’1’ ） -- ”，

任意输入登陆密码•••••（任意字符），选择用户类型为“学生”，点击确定，

弹出显示“没有此用户或密码错误”的消息，防御 SQL 注入攻击成功。


登录窗口
消息
没有此用户或密码错误
确定
确定
取消
关于