安全性分析与防护

一、安全隐患分析

1. SQL 注入

- 危险点：使用字符串拼接构造 SQL 语句。
- 建议：使用 PreparedStatement 代替 Statement，避免直接拼接用户输入。

```
//不安全
String query = "SELECT * FROM users WHERE username='" + username + "'";
//更安全
PreparedStatement ps = conn.prepareStatement("SELECT * FROM users WHERE username = ?");
ps.setString(1, username);
```

2. XSS（跨站脚本攻击）

- 危险点：将用户输入直接输出到网页中而不做 HTML 编码。
- 建议：使用 StringEscapeUtils.escapeHtml4() 或前端框架中的 XSS 防护机制。

3. CSRF（跨站请求伪造）

- 危险点：用户登录后自动发送敏感请求（如删除账户）。
- 建议：为表单添加 CSRF Token 并在服务端验证。

4. 敏感信息明文存储

- 危险点：将数据库账号密码写死在代码或配置文件中，或以明文存储用户密码。
- 建议：

- 使用 application.properties（或 .env）进行配置管理。
- 用户密码使用哈希算法（如 bcrypt）加密存储。

5. 身份认证与会话管理不当

- 危险点：未检查用户身份即可访问受限资源。
- 建议：
- 使用 HttpSession 管理用户登录状态；
- 对关键控制器加认证注解（如 @PreAuthorize、@Secured）。

6. 文件上传漏洞

- 危险点：允许上传任意文件，可能被上传恶意 .jsp 或 .exe。
- 建议：
- 验证文件类型和扩展名；
- 存储路径应避免与 Web 路径一致；
- 重命名上传文件，避免执行。

二、添加安全性防护措施

1. 密文存储密码（使用 BCrypt 加密）

明文存储密码是严重安全漏洞。使用 BCrypt 等哈希算法可以抵御暴力破解和泄露攻击。

实现方法：

引入 spring-boot-starter-security（或手动引入 spring-security-crypto）后，使用以下代码加密密码：

```
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();

String hashedPassword = encoder.encode(plainPassword);
```

验证密码：

encoder.matches(plainPassword, hashedPasswordFromDB)

2. 用户认证与授权（Spring Security）

添加依赖

<!-- pom.xml -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

配置类

创建 SecurityConfig.java：

import org.springframework.context.annotation.*;

import
org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.*;

import org.springframework.security.core.userdetails.*;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

```
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf().disable() // 开发阶段可禁用，生产应启用
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/admin/**").hasRole("ADMIN")
                .requestMatchers("/user/**").hasAnyRole("USER", "ADMIN")
                .anyRequest().permitAll()
            )
            .formLogin()
            .and()
            .logout();
        return http.build();
    }
    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

配置用户认证（内存或数据库）

示例：内存用户

```
@Bean
```

```
public UserDetailsService userDetailsService() {
    UserDetails user = User.builder()
        .username("user")
        .password(passwordEncoder().encode("password"))
        .roles("USER")
        .build();
    UserDetails admin = User.builder()
        .username("admin")
        .password(passwordEncoder().encode("adminpass"))
        .roles("ADMIN")
        .build();
    return new InMemoryUserDetailsManager(user, admin);
}
```

示例：自定义用户详情服务（从数据库读取）

实现 UserDetailsService 接口，从数据库中加载用户和角色：

```
@Service
public class MyUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository userRepository;
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
```

```
        UserEntity user = userRepository.findByUsername(username);
        if (user == null) throw new UsernameNotFoundException("User not found");
        return User.withUsername(user.getUsername())
                .password(user.getPassword()) // 已加密
                .roles(user.getRole())
                .build();
    }
}
```

3. 使用 Apache Shiro

Apache Shiro 是另一个轻量级安全框架，适合不使用 Spring 的项目。

Maven 依赖

```
<dependency>
    <groupId>org.apache.shiro</groupId>
    <artifactId>shiro-spring</artifactId>
    <version>1.11.0</version>
</dependency>
```

示例配置（ShiroConfig.java）

```
@Configuration
public class ShiroConfig {
    @Bean
    public Realm realm() {
```

```
        SimpleAccountRealm realm = new SimpleAccountRealm();
        realm.addAccount("user", "password", "USER");
        realm.addAccount("admin", "adminpass", "ADMIN");
        return realm;
    }
    @Bean
    public SecurityManager securityManager(Realm realm) {
        DefaultSecurityManager securityManager = new
DefaultSecurityManager();
        securityManager.setRealm(realm);
        return securityManager;
    }
    @Bean
    public ShiroFilterFactoryBean shiroFilter(SecurityManager
securityManager) {
        ShiroFilterFactoryBean filter = new ShiroFilterFactoryBean();
        filter.setSecurityManager(securityManager);
        Map<String, String> filterChain = new LinkedHashMap<>();
        filterChain.put("/admin/**", "authc, roles[ADMIN]");
        filterChain.put("/user/**", "authc, roles[USER]");
        filterChain.put("/**", "anon");
        filter.setFilterChainDefinitionMap(filterChain);
        return filter;
```

```
        }
    }
```

三、安全性测试用例

1. SQL 注入测试用例

用例编号	测试目标	请求位置	测试输入	预期结果
SQL-01	登录接口注入	/login 参数 username	' OR '1'='1	登录失败，返回错误提示
SQL-02	注册接口注入	/register 参数 email	test@test.com'); DROP TABLE users; --	数据库无异常，返回注册失败
SQL-03	查询商品	/search?keyword=	' OR 1=1 --	不应返回所有数据，不应报错

建议：后端使用 PreparedStatement 预编译语句防止 SQL 注入。

2. XSS 测试用例

用例编号	测试目标	输入位置	测试输入	预期结果
XSS-01	用户评论区	comment 参数	<script>alert('xss')</script>	页面应转义，不执行脚本
XSS-02	用户名字段	username	<img src=x onerror=alert(1)>	页面正常显示，未弹窗
XSS-03	搜索框回显	q	<svg/onload=alert(1)>	搜索结果中不会触发脚本

建议：输出时做 HTML 转义，例如使用：

- Java: StringEscapeUtils.escapeHtml4(input)
- JSP: <c:out value="${userInput}"/>

3. CSRF 测试用例

用例编号	测试目标	请求位置	测试操作
CSRF-01	删除商品	模拟已登录用户访问 <img src="http://yourapp/delete?id=1">	操作失败，需 CSRF Token
CSRF-02	修改密码	模拟表单提交，无 Token	修改失败，提示无效请求

建议：启用 CSRF 保护（Spring Security 默认启用），前端表单加上隐藏字段：

<input type="hidden" name="_csrf" value="${_csrf.token}">

4. 认证与授权绕过测试

用例编号	测试目标	路径	测试操作
AUTH-01	未登录访问受限页面	/user/dashboard	应重定向到登录页面
AUTH-02	普通用户访问管理员接口	/admin/addProduct	返回 403 Forbidden
AUTH-03	修改用户 ID 尝试越权	/user/edit?id=2	拒绝操作，不允许更改他人信息

5. 密码安全测试用例

用例编号	测试目标	测试内容	预期结果
PASS-01	注册时密码弱	使用 123456 或 password	返回密码强度不足提示
PASS-02	数据库存储密码	查看数据库中的密码字段	不应是明文，应是加密（如 bcrypt）
PASS-03	登录密码错误	输入错误密码登录	登录失败，提示错误，不暴露具体原因

6. 敏感信息泄露测试用例

用例编号	测试目标	请求	预期结果
INFO-01	访问 .git/ 路径	/webapp/.git/config	返回 403 或 404
INFO-02	报错信息暴露	故意请求无效字段或 SQL 错误	不应暴露堆栈信息或 SQL 语句

7. HTTP 安全头测试

使用 curl 或 Burp 检查是否启用安全头：

curl -I https://yourapp.com

应包含如下头部：

- Strict-Transport-Security
- X-Content-Type-Options: nosniff

- X-Frame-Options: DENY
- Content-Security-Policy