深入浅出 Spring Security

一场系统安全之旅

江南一点雨

华为云MVP、华为云云享专家

目录

1 会话简介

2 会话并发管理

3 会话固定攻击与防御

会话简介

什么是会话?

当浏览器调用登录接口登录成功后，服务端会和浏览器之间建立一个会话（Session），浏览器在每次发送请求时都会携带一个SessionId，服务端则根据这个SessionId来判断用户身份。当浏览器关闭后，服务端的Session并不会自动销毁，需要开发者手动在服务端调用Session销毁方法，或者等Session过期时间到了自动销毁。

在Spring Security中，与HttpSession相关的功能由SessionManagementFilter和SessionAuthenticationStrategy接口来处理，SessionManagementFilter过滤器将Session相关操作委托给SessionAuthenticationStrategy接口去完成。

会话并发管理

什么是会话并发管理?

会话并发管理就是指在当前系统中，同一个用户可以同时创建多少个会话，如果一台设备对应一个会话，那么也可以简单理解为同一个用户可以同时在多少台设备上进行登录。默认情况下，同一用户在多少台设备上登录并没有限制，不过开发者可以在Spring Security中对此进行配置。

会话并发管理

“挤下线”

当会话并发数达到限制时，新的会话将之前旧的会话“挤下线”，旧的登录会话失效。

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .and()
            .csrf()
            .disable()
            .sessionManagement()
            .maximumSessions(1);
}
```

会话并发管理

“限制登录”

当会话并发数达到限制时，新的会话将被限制创建，除非旧的会话主动退出登录。

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .and()
            .csrf()
            .disable()
            .sessionManagement()
            .maximumSessions(1)
            .maxSessionsPreventsLogin(true);
}
```

会话固定攻击与防御

什么是会话固定攻击

会话固定攻击（Session fixation attacks）是一种潜在的风险，恶意攻击者有可能通过访问当前应用程序来创建会话，然后诱导用户以相同的会话ID登录（通常是将会话ID作为参数放在请求链接中，然后诱导用户去单击），进而获取用户的登录身份。

会话固定攻击与防御

会话固定攻击步骤

（1）攻击者自己可以正常访问javaboy网站，在访问的过程中，网站给攻击者分配了一个sessionid。
（2）攻击者利用自己拿到的sessionid构造一个javaboy网站的链接，并把该链接发送给受害者。
（3）受害者使用该链接登录javaboy网站（该链接中含有sessionid），登录成功后，一个合法的会话就成功建立了。
（4）攻击者利用手里的sessionid冒充受害者。

会话固定攻击防御策略

Spring Security中从三方面入手防范会话固定攻击：

（1）Spring Security中默认自带了Http防火墙，如果sessionid放在地址栏中，这个请求就会直接被拦截下来（本书第8章会详细分析Http防火墙）。
（2）在Http响应的Set-Cookie字段中有httpOnly属性，这样避免了通过XSS攻击来获取Cookie中的会话信息，进而达成会话固定攻击。
（3）在用户登录成功后，改变SessionId，Spring Security中默认实现了该种方案。

会话固定攻击与防御

修改 SessionId 的四种策略

http.sessionManagement().sessionFixation().changeSessionId();

（1）changeSessionId()：用户登录成功后，直接修改HttpSession的SessionId即可，默认方案即此，对应的处理类是ChangeSessionIdAuthenticationStrategy。

（2）none()：用户登录成功后，HttpSession不做任何变化，对应的处理类是NullAuthenticatedSessionStrategy。

（3）migrateSession()：用户登录成功后，创建一个新的HttpSession对象，并将旧的HttpSession中的数据拷贝到新的HttpSession中，对应的处理类是SessionFixationProtection Strategy。

（4）newSession()：用户登录成功后，创建一个新的HttpSession对象，对应的处理类也是SessionFixationProtectionStrategy，只不过将其里边的migrateSessionAttributes属性设置为false。

谢 谢