基于JAAS和J2EE Web容器的验证与授权

姜 伟，杜平安，李 磊

(电子科技大学机械电子工程学院 成都 610054)

【摘要】在Borland应用服务器的基础上，使用JAAS与J2EE Web容器内在的安全机制，并借助Oracle数据库的用户验证，实现了Web应用中对用户的验证和授权。把用户能访问到的资源控制到页面级，将开发阶段需要考虑的安全问题转移到部署阶段，实现了应用逻辑与安全逻辑的彻底分离。实践表明，使用JAAS可以提高整个系统的开发效率，而Web容器提供的验证与授权可以很好地和数据库安全域相结合。

关 键 词 验证； 授权； JAAS； J2EE Web容器； 安全
中图分类号 文献标识码 A

Implementation of Authentication and Authorization Based on JAAS and J2EE Web Container

JIANG Wei，DU Ping-an，LI Lei

(School of Mechatronics Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract To implement the authentication and authorization in a Web application based on Browser/Server model. JAAS and J2EE Web Container's security realm, combining with Oracle's self authentication, are used to authenticate and authorize users who want to access the Web application. The resources that a user can access are limited at Web page level and the security issue considered in development phase is moved to deployment. The business logic and rights management are isolated so that programmers are no need to write codes in each page to examine whether the user have rights to access it. The results show that using Java Authentication and Authorization Service (JAAS) can enhance the entire system's development efficiency and the security mechanism provided by Web Container can work with the database's security realm well.

Key words authentication; authorization; JAAS; J2EE Web container; security

在大中型 Web 应用中，实现系统的安全性是系统架构师必须考虑的事。系统的安全性主要体现在两部分：验证和授权。验证(Authentication)表示确定一个用户就是他所声称的那个人；授权(Authorization)或访问控制，表示已通过验证的用户只能访问那些允许他访问的资源。不同的用户对不同的资源拥有不同的访问权限，如何对众多的用户进行验证和授权是构建系统的关键。在早期的 Web 应用中，程序员在每个页面编写检查权限的代码来实现对页面的保护。如果权限发生变化，则需对应用程序级的代码进行修改，整个系统的可维护性和代码可重用性不高，开发效率较低。而利用 JAVA 验证与授权服务(Java Authentication and Authorization Service，JAAS)提供的验证授权机制和 Web 容器内在的安全机制，可以在部署阶段来实现对用户的授权，并能借助其他安全域(如数据库)来实现对用户的验证。

1 JAAS提供的验证和授权

JAAS是J2EE规范中定义的一种验证授权框架。它提供的验证机制是一种可插入式的，即当前应用能在保持已有验证机制不变的情况下加入新的验证方式，而不用更改应用程序级的代码；系统管理员在配置文件中决定采用哪些验证技术以及它们的验证顺序。因而非常适合企业在已经拥有一套验证机制的情况下使用。

在JAAS框架之下，开发人员只需在应用程序层和登录上下文(LoginContext)交互。LoginContext之下是一组动态配置的登录模块(LoginModule)对象，LoginModule是调用特定验证机制的接口，对一个具

收稿日期：2005 – 10 – 08
基金项目：国家863/CIMS主题资助项目(2003AA411210)
作者简介：姜 伟(1981 –)，男，硕士，主要从事CAD/CAE/CAM以及制造业信息化方面的研究.

体实现了LoginModuel接口的类，就是一种验证方式。开发系统时可以根据需要使用J2EE1.4中包含的几种 LoginModule 实现类 (如 JndiLoginModule 、Krb5LoginModule 、 NTLoginModule 、 UnixLoginModule)，或编写自己的LoginModule，或使用应用服务器提供的相关实现。

JAAS 框架通过一个配置文件来指定要使用的验证机制，并对验证模块进行封装。开发人员无需为如何调用验证模块编写任何代码，系统会根据配置文件中定义的验证机制采取相应的验证方式。如果验证成功，就会返回一个包含验证信息的主体(Subject)，这个验证信息将会用于授权过程。

JAAS 配置文件(config_jass)如下[1-7]：

```
myRealm {
com.borland.security.provider.authn.JDBCLoginM-odule required
DRIVER= oracle.jdbc.driver.OracleDriver
URL= "jdbc:oracle:thin:@localhost:1521:mydatabase"
DBTYPE=ORACLE USERNAME=test
PASSWORD=test;};
```

上述配置表明，容器使用名称为 myRealm 的安全域，调用 Borland 企业应用服务器(BES)提供的 JDBCLoginModule 登录模块。

JAAS 的授权机制将保证已通过验证的用户，有权访问他所能访问的受保护的资源。在 Web 应用中，资源是指用户使用 HTTP 方法，通过 URL 访问的一个文件或 url-patten。而对用户权限的控制是依据其拥有的角色来实现的[2]。在 JDBCLoginModule 类中，会为通过验证的用户“授权”：首先，将通过验证的用户名传到 JDBCLoginModule 中，调用其中的一个方法——从数据库的数据字典 dba_role_privs 中查出该用户拥有的所有角色；代码段如下：

```
return getUserGroups("select granted_role from dba_role_privs where grantee ='" + s.toUpperCase() + "'"); //获得当前登录用户拥有的所有角色，其中 s 是当前登录用户。
```

其次，J2EE Web 容器根据发布描述符 web. xml 中<security-constraint>元素下定义的角色和资源匹配列表，确定当前登录的用户有权访问哪些资源。

2 J2EE Web容器内在的安全机制

一个实现了 J2EE 规范的容器必须提供内在的安全机制。这样，所有的验证和授权都可以由容器处理，从而将应用的开发人员解脱出来。系统管理员根据容器提供的一系列接口和一个配置文件来实现系统的安全性，而不用在每个页面中都编写检查权限的代码。在整个系统的开发过程中，开发人员就可以专心致力于业务的实现，而不用考虑权限问题，从而提高开发效率。

2.1 容器提供的验证

基于 J2EE 的 Web 容器提供了三种类型的验证机制[3-4]：基本方式、基于表单(FORM)的方式和交互(MUTUAL)方式。多数 Web 应用都使用基于 FORM 的验证方式，因为它允许用户定制自己的验证(登录)和错误(验证失败)页面，非常适合 Web 应用。本文使用基于 FORM 的方式来验证用户。

在发布描述符 web.xml 中的 login-config 元素下定义了验证机制的类型、登录和登录失败页面的 URL。具体配置如下：

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/loginFailed.jsp</form-error-page>
  </form-login-config>
</login-config>
```

在登录页面 login.jsp 中 login form 必须包含两个字段，用来输入用户名和密码。这两个字段的 name 必须是: j_username 和 j_password ；而且 form 必须提交到名为 j_security_jeck 的动作。

j_security_check 动作属性值是一个特殊的 URI，由容器识别。当用户提交此表单时，容器会根据当前配置的安全域 myRealm 将 j_username 和 j_password 作为参数传递到 JDBCLoginModule 中，对用户进行验证。当用户输入的用户名、密码和 Oracle 数据库中的一个真实用户相匹配时，验证通过；否则验证失败，返回 loginFailed.jsp 页面。

2.2 容器提供的授权

J2EE Web 容器通过定义在发布描述符 Web.xml 中的安全角色与资源匹配，实现对 Web 资源的授权。Servlet 规范所定义的安全机制描述了如何为一个 Web 应用指定访问控制约束。但是对某个页面的访问权限要授予一个角色(Role)而不是直接授予一个用户或一个组[4-5]。一旦用户通过验证(这里，用户必须是 Oracle 数据库中的一个真实用户才能通过验

证)，容器就会继续调用 JDBCLoginModule 为通过验证的用户获取安全角色，并根据用户想要访问的资源检查该用户拥有的角色是否属于发布描述符中<auth-constraint>元素下已定义的一个角色。如果用户不属于这些角色中的任何一个，容器就会向客户端返回一个错误消息(HTTP403 错误)。此时可以在发布描述符(web.xml)中配置<error-page>元素[6-7]，让容器在捕获此错误代码后返回一个“提示权限不足”的页面 deny.jsp。配置如下：

```
<error-page>
    <error-code>403</error-code>
    <location>/deny.jsp</location>
</error-page>
```

web.xml 中，<security-constraint>元素定义了受保护的 Web 资源和有权限访问这些资源的角色列表。配置如下：

```
<security-constraint>
<web-resource-collection>
    <web-resource-name>AdminPages</
web-resource-name>
    <description> accessible by authorised
users </description>
    <url-pattern>/admin/*</url-pattern>
    <http-method>GET</http-method>
</web-resource-collection>
<auth-constraint>
    <description>These are the roles who
have access</description>
    <role-name>manager</role-name>
</auth-constraint>
</security-constraint>
```

基于上述约束，当用户访问 admin 目录下的所有文件时，该用户必须通过验证并且拥有 manager 角色才能访问，否则提示无权访问。

3　基于JAAS与Web容器的验证和授权实现方法

基于上述 JAAS 安全框架和 J2EE Web 容器提供的验证和授权，在某油田公司的一个实际项目中，由于要求保留已有 Oracle 数据库端的用户验证机制，即采用真实的数据库用户作为最终登录系统的用户，所有登录系统的用户都事先在数据库端创建好。实际开发时选用了 BES 服务器提供的 JDBCLoginModule 作为本系统的验证方式。它使用标准的 JDBC 数据库接口与数据库交互，调用数据库已有的验证机制，判断登录的用户是否属于 Oracle 数据库中的真实用户。

实现系统安全性的关键步骤如下[7-8]：

(1) 在Oracle中创建用户和角色，为角色设置激活密码并授予相应的权限；

(2) 配置config.jaas，确定容器使用的安全域和选择哪些登录模块作为验证方法；

(3) 配置web.xml，定义受保护的Web资源和有权限访问这些资源的角色列表；

(4) 对该应用施加安全选项，并发布应用。

需要配置的几个关键页面是：登录页面login.jsp、用户名密码错误页面error.jsp、验证通过但权限不足页面deny.jsp。当用户在login.jsp页面提交表单时，由于采用了JAAS安全域，J2EE Web容器会自动生成LoginContext对象，并通过LoginContext的login()方法调用JDBCLoginModule，进而再完成对用户的验证和授权[9-10]。代码段如下：

```
LoginContext lc = new LoginContext (“myRealm”);
//调用config.jaas中的myRealm配置块。
lc.login();//登录，调用myRealm配置块下指定的
JDBCLoginModule。
```

当LgoinContext.login()方法被调用时，它会调用 JDBCLoginModule 的 login() 方法，在其中使用 CallbackHandler对象与Oracle数据库交互完成对用户的验证和授权。以上过程都是容器自动完成的，无需用户干预。详细的验证、授权流程如图2所示。

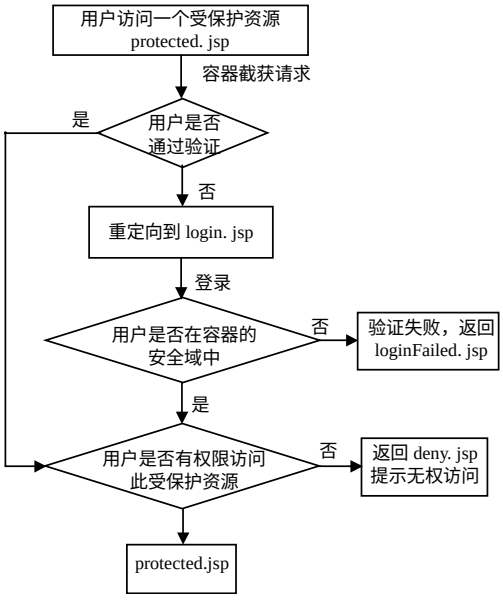

图 2　验证、授权流程

当容器对一个用户的验证和授权成功完成后，会把当前用户所拥有的角色和资源匹配列表保存到服务器的会话(session)中，在该用户 session 不过期的情况下，如果访问其他页面，服务器将直接从 session 中获得他所能访问的对应资源列表并进行判断其是否有足够的权限。在应用发布后，如果数据库端的角色和对应权限发生变化，将不会反映到客户端。用户只有重新登录，才能再次获得他的角色资源匹配列表；如果对配置文件 web.xml 和 config.jaas 进行了修改，则需要重新发布应用，使新的权限生效。

通过上述的分析可以看出，管理员把用户能访问的资源控制到了页面级。所有需要保护的资源和能访问的对应角色都事先在配置文件中写好。如果需要对用户的权限进行更改的话，只需要修改配置文件，然后重新发布应用即可。将开发阶段需要考虑的安全问题转移到部署阶段，实现了应用逻辑与安全逻辑的彻底分离，使整个系统有相当高的可维护性和升级性。

另外开发人员还可以根据当前用户是谁(属于什么角色)，对页面的内容进行个性化设置。如管理员拥有删除账号的权利，则在页面上显示删除按钮。代码段如下：

```
if(request.isUserInRole("manager")){//add a delete button}
```

4 结 束 语

本文把 JAAS 验证和授权机制用于 J2EE Web 应用中，利用 J2EE Web 容器内在的安全机制，以 FORM 验证的方式，借助第三方提供的验证机制(Login-Module)，并与数据库安全域协同工作，对登录系统的用户进行身份验证和授权。整个系统的安全性都由系统管理员通过配置来实现，应用的开发人员无需关心权限问题，把权限问题从表示层和业务层分离开来。在修改角色和对应权限时无需对应用程序级的代码进行修改，提高了系统的开发效率和代码的可重用性、可维护性。对大中型企业实施网上电子商务、办公自动化和构建企业信息系统的安全模块时，有很强的借鉴意义。

参 考 文 献

[1] Borland Company. Borland Enterprise Server TM developer guide[M]. America: [s. n.], 2004.
[2] 林天峰. 基于JAAS的Java安全编程[J]. 计算机应用与软件, 2003, 20(7): 86-88.
[3] ARUMUGAM P. J2EE form-based authentication[EB/OL]. http://www.onjava.com/pub/a/onjava/2002/06/12/form.html, 2005-04-11.
[4] 陆荣杰, 刘知贵, 黄晓芳. J2EE中基于容器管理的Web客户端安全验证[J]. 兵工自动化, 2005, 24(3): 47-48.
[5] BERGSTEN H. JSP设计[M]. 第3版. 北京: 中国电力出版社, 2004.
[6] 高正宪, 李中学. Web环境下基于角色的访问控制策略及实现[J]. 计算机工程, 2004, 30(8): 133-135.
[7] 刘孝保, 杜平安. J2EE模式下基于角色的访问控制的应用研究[J]. 计算机应用, 2006, 26(6): 1331-1333.
[8] 刘孝保, 杜平安. 基于角色的访问控制在多应用层CIMS中的应用[J]. 四川大学学报(工程科学版), 2007, 39(2): 140-144.
[9] 陈 阳, 佘 堃, 周明天. RBAC扩展J2EE/JAAS安全机制的设计与实现[J]. 计算机应用研究, 2005, 22(1): 114-116.
[10] 张方舟, 王东安, 李 生, 等. 采用J2EE安全机制支持RBAC模型的研究和实现[J]. 计算机工程, 2006, 32(13): 125-127.

编 辑 税 红