基于 JAAS 的 JAVA 安全应用研究

Java Safety Application Research Based on JAAS

高　月　吕国斌　(武汉中国地质大学 计算机科学与技术系　430074)
梁本亮　(上海 同济大学 土木工程学院　200092)

摘要: JAAS 是对原有的 JAVA2 安全框架的一个重要补充。本文将 JAAS 出现前后,JAVA 的安全模式进行了比较。阐述了 JAAS 在 JAVA 安全编程中的应用。最后给出一个实例的编程思路和代码分析,并对采用 JAAS 的登录安全控制进行了演示。
关键词: JAAS　认证　授权　策略

1　引言

用户认证和访问控制是大多数 java 应用的重要安全尺度,特别是 J2EE 应用。Java Authentication Authorization Service(JAAS,Java 验证和授权 API)提供了灵活和可伸缩的机制来保证客户端或服务器端的 Java 程序。JAAS 是一个比较新的 Java API。在 J2SE 1.3 中,它是一个扩展包;在 J2SE 1.4 中变成了一个核心包。Java 早期的安全框架强调的是通过验证代码的来源和作者,保护用户避免受到下载下来的代码的攻击。JAAS 强调的是通过验证谁在运行代码以及他/她的权限来保护系统面受用户的攻击。它能够将一些标准的安全机制,例如 Solaris NIS(网络信息服务)、Windows NT、LDAP(轻量目录存取协议),Kerberos 等通过一种通用的,可配置的方式集成到系统中。

2　问题的提出

在 JAAS 出现以前,Java 的安全模型是为了满足跨平台的网络应用程序的需要而设计的。在 Java 早期版本中,Java 通常是作为远程代码被使用,例如 Applet。因此最初的安全模型把注意力放在通过验证代码的来源来保护用户上。

根据用户在代码中的可信度,Java 平台允许对计算资源(如磁盘文件和网络连接)进行细颗粒度的访问控制。Java 平台的大多数基本安全性特性都是为保护用户免受潜在的恶意代码破坏而设计的。例如,第三方证书支持的数字签名代码确保代码来源的身份。根据用户对代码来源的了解,它可以选择授予或拒绝对该代码的执行权。同样,用户可以根据给定代码来源的下载 URL 授予或拒绝访问权。

基于 Java 的系统上的访问控制是通过策略文件实现的,该文件包含的语句如下:

```
grant signedBy "gyue", codeBase "http://www.cug.edu.cn" {
        permission java.io.FilePermission "/tmp/abc", "read";
};
```

该语句允许由"gyue"签署并从 http://www.cug.edu.cn 装入的代码读取 /tmp/abc 目录。

JAAS 的出现反映了 Java 的演变。传统的服务器/客户端程序需要实现登录和存取控制,JAAS 通过对运行程序的用户的进行验证,从而达到保护系统的目的。JAAS 的认证和授权服务一起工作,提供了补充功能:它们防止敏感的 Java 应用程序代码遭到潜在的恶意用户破坏。

通过在应用程序和底层的验证和授权机制之间加入一个抽象层,JAAS 可以简化涉及到 Java Security 包的程序开发。抽象层独立于平台的特性使开发人员可以使用各种不同的安全机制,而且不用修改应用程序级的代码。和其他 Java Security API 相似,JAAS 通过一个可扩展的框架:服务提供者接口(Service Provider Interface,SPI)来保证程序独立于安全机制。服务提供者接口是由一组抽象类和接口组成的。应用程序级的代码主要处理 LoginContext。在 LoginContext 下是一组动态配置的 LoginModules。LoginModule 使用正确的安全机制进行验证。

3　JAAS 实例说明

本实例演示了 JAAS 登录,它从给出对话框读取用户输入的信息,通过密钥库中的信息进行验证。基于 jboss 实现并扩展了 SUN JAAS( )登录安全机制。

3.1　编程思路

身份验证涉及如何同用户交互以获取用户的信息(帐号、口令、指纹等)、如何根据用户输入的信息验证用户等。

用户信息使用回调处理器,本实例中使用 JAAS 提供的 com.sun.security.auth.callback 包中的 DialogCallbackHandler 类;验证用户信息使用登录模块,本实例中使用 JAAS 提供的

com. sun. security. auth. module 包中的 KeyStoreLoginModule 类。这些类都已经集成在 J2SDK1. 4 中。JAAS 使用可插入式验证模块(PAM)的结构,开发者可以不管具体的验证方式而使用标准的接口进行开发,使用何种验证技术以及如何进行验证可交给系统管理员通过登录配置文件进行配置,而不需要修改应用程序的代码。

登录配置文件是一个普通的文本文件,包含了使用哪些登录模块,类似如下格式:

```
simp {
  com. sun. security. auth. module. KeyStoreLoginModule required
  keyStoreURL = "file:C:/java/ch9/SimpLogin/mykeystore";
};
```

在运行 Java 程序时可以通过命令行选项指定使用该配置文件中指定的方法进行验证。其中"simp"是配置条目的名称,一个配置文件中可以有多个条目。com. sun. security. auth. module. KeyStoreLoginModule 是登录模块类的全名,这里使用 JAAS 提供的 KeyStoreLoginModule 类,它采用基于密钥库的验证机制,通过 keyStoreURL 指定密钥库的位置。只有当用户输入的别名、密码和密钥库中的相吻合,验证才通过。

JAAS 的登录机制使用 javax. security. auth. login 包中的 LoginContext 类实现独立于底层验证技术的登录。在创建 LoginContext 对象时通过其参数指定使用哪个回调处理器和用户进行交互,以及如何获得登录模块。执行其 login( )方法则自动执行登录模块的登录操作,由登录模块调用回调处理器向用户询问帐号、口令等相关信息,并进行验证。如果没有通过验证,将抛出 LoginException 异常。具体编程步骤如下:

(1) 创建和用户交互的回调处理器对象

```
DialogCallbackHandler handler = new DialogCallbackHandler( );
```

分析:不妨使用 com. sun. security. auth. callback 包中的 DialogCallbackHandler 类,它使用 Swing 对话窗口向用户询问与验证相关的问题。针对不同的验证方式,DialogCallbackHandler 会出现不同的窗口。如对本节的使用密钥库的验证方式,它会要求用户输入别名、密钥库密码和私钥密码。

除了 DialogCallbackHandler 类外,com. sun. security. auth. callback 包还提供了 TextCallbackHandler,以文本方式和用户交互。

(2) 创建 LoginContext 对象

```
LoginContext c = new LoginContext("simp",handler);
```

分析:LoginContext 类的构造器有两个参数,第一个是登录配置文件中的条目名称,JAAS 会读取登录配置文件时会找到该条目,并读取其中的登录模块,进而执行这些登录模块。第二个参数是上一步创建的回调处理器,以后登录模块会通过它和用户进行交互。

(3) 执行登录操作

```
c. login( );
```

分析:执行上一步得到的 LoginContext 对象的 login( )方法,该方法会自动执行登录模块的 login( )和 commit( )方法进行登录操作。而登录模块的 login( )方法会调用第 1 步的回调处理器读取用户输入的信息,并验证这些信息和数据库中的信息是否匹配。

该步骤若验证失败则抛出 LoginException 异常。

(4) 处理登录结果

```
Subject s = c. getSubject( );
System. out. println(s. getPrincipals( ));
```

分析:登录成功,则可以执行 LoginContext 对象的 getSubject( )方法获得代表登录者的主体,并可进一步执行其 getPrincipals( )方法获得其身份标志。如果登录不成功,则可以处理 LoginException 对象,可根据需要返回出错信息或重复登录。

3.2 代码与分析

本实例所使用的登录配置文件 simp. config 内容如下:

```
simp {
  com. sun. security. auth. module. KeyStoreLoginModule required
  keyStoreURL = "file:C:/java/ch9/SimpLogin/mykeystore";
};
```

使用 LoginContext 类实现登录的完整程序如下:

```
import com. sun. security. auth. callback. DialogCallbackHandler;
import javax. security. auth. *;
import javax. security. auth. login. *;
public class SimpLogin {
  public static void main(String[ ]args) throws Exception {
    //登录
    DialogCallbackHandler handler = new DialogCallbackHandler( );
    LoginContext c = new LoginContext("simp",handler);
    boolean pass;
    try {
```

```
            c.login();
            //登录成功
            pass = true;
        }
        catch (LoginException le) {
            //登录失败
            pass = false;
            System.err.println("Authentication failed:");
            System.err.println("  " + le.getMessage());
        }
        //显示登录结果
        if(! pass){
            System.out.println("Sorry");
        }
        else{
            System.out.println("Authentication succeeded!");
            Subject s = c.getSubject();
            System.out.println(s.getPrincipals());
        }
    }
}
```

3.3 运行程序

程序运行在 C:\java\ch9\SimpLogin 目录下,该目录中有编程者编写的 SimpLogin 程序。程序运行者使用 simp.config 登录配置文件,该密钥库拷贝在当前目录中,密码是 wshr.ut,有一个条目 mytest。

输入 java -Djava.security.auth.login.config = =simp.config SimpLogin

运行程序,其中命令行选项 -Djava.security.auth.login.config 指定登录配置文件的名称。程序运行将出现错误!

该窗口便是程序中创建 LoginContext 类时所传入的 com.sun.security.auth.callback.DialogCallbackHandler 类型的对象弹出的。在其中用户可以输入别名 mytest,密钥库密码 wshr.ut。正确输入信息后,单击"确定"按钮,DOS 窗口将显示:

欢迎光临寒舍!

可见,验证通过后程序可获取用户的身份。

如果输入的信息和密钥库中不匹配,如在密码一项中输入的不是"wshr.ut",则提示验证失败:

对不起,有错误发生,可能有以下原因:

(1) 您是一个无效的用户

(2) 你输入了错误的 ID 和密码

请检查后再试

该程序在 win2000 server 平台上,在 Jboss3.2.3 下调试通过,数据库采用 Oracle9i。

参考文献

1 Garben Thomas Smith Shery 等著,刘堃等译,J2EE 技术内幕,机械工业出版社,2002-06。

2 Java2 的安全性新特性及其应用,金胜昔、步俊杰、吉逸,全国第二届 Java 技术应用学术会议论文集[C]. [s.l.]:[s.n.],1999.

3 Sun Microsystems. JAAS Login Module Developer's guide [EB/OL]. www.sun.cum.2001.

4 Samar V, Lai C. Making Login Services Independent of Authentication Technologies [EB/OL]. Sun Microsystems, Inc. from