

基于 JAAS 的认证授权系统的设计与实现

盛 静

(长沙职业技术学院 信息系,湖南 长沙 410111)

摘 要:在传统的面向对象程序设计方法中使用 JAAS 会导致认证和授权代码与业务逻辑的实现代码纠缠在一起,不利于重用和维护。而基于面向方面的软件开发技术——AOP,利用方面封装现有的认证授权逻辑,可以保证业务逻辑和认证授权的有效分离。以现代软件的发展趋势以及传统基于 OOP 的开发技术为背景研究了 JAAS 认证和授权机制,分析了传统方法的优缺点。利用 AOP 设计了基于 JAAS 的认证授权系统,并通过原型系统的构造,验证了 AOP 技术实现认证授权系统的可行性和有效性。

关键词:面向方面软件开发;认证;授权;AspectJ;JAAS

中图分类号:TP311.5

文献标识码:A

文章编号:1673-629X(2009)07-0184-03

Design and Implementation of Authentication and Authorization System Based on JAAS

SHENG Jing

(Department of Information, Changsha Vocational and Technical College, Changsha 410111, China)

Abstract: In the traditional OOP method, using JAAS makes the codes of authentication and authorization tangle with the business logic which makes the whole system hard to be reused and maintained. While the new aspect-oriented programming (AOP) method uses aspects to encapsulate existing authentication and authorization codes which are effectively separated with business logic. Introduces the trend of current software and analyzes characteristics of JAAS mechanism based on OOP. Then it introduces the design of an authentication and authorization system based on AOP technology and proves its feasibility and validity with the construction of a prototype.

Key words: aspect-oriented programming; authentication; authorization; AspectJ; JAAS

0 引 言

认证和授权系统是企业信息系统的重要组成部分,在网络攻击越来越频繁的 Internet 环境中,越来越显得重要。当前认证授权的实施方式主要是采用基于面向对象的程序设计方法(Object Oriented Programming, OOP)。不同的业务逻辑使用认证和授权系统的时候,OOP 存在很多缺点,导致业务逻辑的核心实现代码混乱,难于维护与重用^[1]。而 AOP^[2~4](Aspect-Oriented Programming,面向方面的程序设计)技术的出现,可以有效地弥补 OOP 开发方式的缺点,它允许对认证授权这类横切关注点进行模块化。因此,研究基于 AOP 技术的认证和授权系统具有很好的应用价值。

1 AspectJ 对 AOP 的支持

AspectJ^[5,6]采用了源代码生成技术来实现 AOP。它提供了一套独有的基于 Java 平台的 AOP 语法,以及专有的 AspectJ 编译器。编译器在编译具有 AspectJ 语法的 Java 程序时,能够识别诸如 aspect, pointcut 等特殊关键字,然后利用静态织入的方式,修改需要被截取的方法所属类的源代码,把 advice 或者 introduce 的业务逻辑代码注入到正确的位置。利用 AspectJ 可以将业务逻辑的核心关注点完全独立出来,然后通过 AspectJ 语法,编写符合核心关注点要求的横切关注点代码,最后通过 AspectJ 编译器,将这两者在编译后期结合起来。采用这种静态织入技术,使得运用了 AOP 技术的系统在运行性能上未受到任何损失^[7]。

由于 AspectJ 是 Java 语言语法和语义的扩展,所以它提供了自己的一套处理方面的关键字。除了包含字段和方法之外,AspectJ 的方面声明还包含切点(pointcut)和通知(advice)成员。AspectJ 方面的声明类似于 Java 语言中的类声明,如图 1 所示。

收稿日期:2009-01-10;修回日期:2009-03-08

基金项目:湖南省自然科学基金项目(05JJ40132)

作者简介:盛 静(1980-),女,湖南长沙人,助理讲师,研究方向为软件工程。

```
public aspect authentication {
    public pointcut authenticationRequired(Account account):
    execution(public * Account. *(..)) && this(account);
    before(Account account): authenticationRequired(account) {
        authenticate(account);
    }
}
```

图 1 AspectJ 中声明方面示例

示例中的切点使用了修饰符和通配符模式来表达 Account 类中定义的“所有公共方法”。切点中对帐户 (Account) 的访问,由 pointcut 参数指定,advice 使用这个参数,而 pointcut 则用 this(account) 把它绑定,以捕获正在执行的方法所隶属的 Account 对象。

before 是一种类型的通知,advice 的主体与方法的主题相似。advice 可以包含认证代码,也可以调用其他方法,表示在切点 authenticationRequired 所指定的方法执行之前执行方法 authenticate(account)。

2 JAAS 的认证和授权机制

JAAS 是从各种不同安全框架和技术发展而来的,JAAS 的可插拔功能基于 Unix 的 PAM(Pluggable Authentication Module)框架。同时,JAAS 还引用了两阶段提交协议中的事务行为和 J2SE 安全包中的安全配置概念。JAAS API 包含两个方面的安全内容,即认证和授权。认证是一种通过可靠的方式确定谁是 Java 代码执行者的方法;而授权是一种确保该人具有执行某种给定任务权限的方法。

1) 认证机制。

图 2 是一个典型的 JAAS 系统应用图。下面简单说明如何实现基于 JAAS 的应用程序认证步骤。

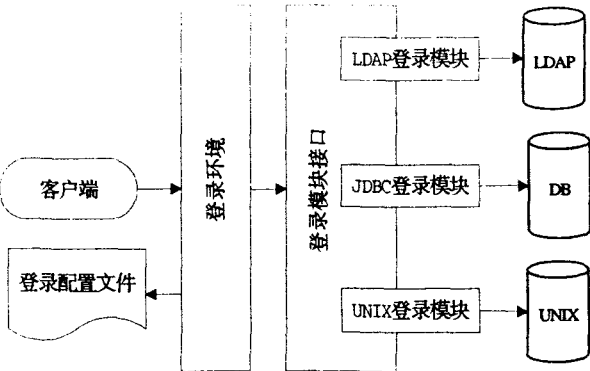


图 2 JAAS 系统应用图

首先,基于 JAAS 的验证的第一步是创建登录环境。应用程序客户端与登录环境对象进行交互,该对象会隐藏应用程序中的所有实现细节,并且只公开某些特定的方法^[8]。其次,登录环境对象引用配置文件,并查找要初始化的登录模块。登录模块对象必须实现

LoginModule 接口。登录模块可以实现不同的验证机制,如:LDAP、生物识别和 Kerberos。J2SE 平台上还提供了其他实现,诸如 NTLoginModule、JndiLoginModule 和 UnixLoginModule 等。

2) 授权机制。

授权是一个检测用户是否有权限使用系统内相应功能的过程。假设某银行系统要实现只有拥有管理权限的用户才能执行进行减免费用的操作。在授权之前必要的步骤是认证,之后要找到该用户所拥有的权限。最后,要验证用户当前执行的操作是否在他所具有的权力范围之内。举例来说,如果银行系统中用户拥有出纳的权限而没有管理权,那么费用减免的操作对该用户是不可用的。

使用 JAAS 实现授权功能的具体方法因不同系统对访问控制的需求不同而不同,典型的步骤:进行认证;创建一个动作对象;执行动作对象;检查访问。

3 用 AOP 实现基于 JAAS 的认证授权系统

基于 AspectJ 创建一个用于认证的基础方面,来实现系统的认证机制。要在具体的系统中实现认证,用户只需扩展这个基础方面,提供一组在切点中需要的操作。图 3 显示了基础方面认证功能模块。

```
public abstract aspect AbstractAuthAspect {
    private Subject _authenticatedSubject; //认证主体
    public abstract pointcut authOperations(); //需要认证操作的切入点
    before() : authOperations() { //认证通知
        if (_authenticatedSubject != null) { return; }
        try {
            authenticate();
        } catch (LoginException ex) {
            throw new AuthenticationException(ex);
        }
    }
    private void authenticate() throws LoginException { //认证逻辑
        LoginContext lc = new LoginContext("Sample", new TextCallbackHandler());
        lc.login();
        _authenticatedSubject = lc.getSubject();
    }
    public static class AuthenticationException extends RuntimeException { //认证异常
        public AuthenticationException(Exception cause) {
            super(cause);
        }
    }
}
```

图 3 只包含认证的抽象认证授权方面
基于图 3 的抽象认证方面,下面给出银行系统中

添加认证功能的示例,如图 4 所示,定义一个方面继承 AbstractAuthAspect,同时定义 authOperations() 切入点。这个例子中,在 Account 和 InterAccountTransferSystem 这两个类中定义捕获调用请求的切入点。

```
import auth. AbstractAuthAspect;
public aspect BankingAuthAspect extends AbstractAuthAspect {
    public pointcut authOperations()
        : execution(public * banking.Account. * (...))
        || execution(public * banking.InterAccountTransferSystem. * (...));
}
```

图 4 银行认证操作

同样使用 AOP 可以不对系统的核心进行任何修改就可以实现授权功能。下面将开发一个可重用的方面来使系统支持授权,而用户只需要扩展该方面实现子方面。而采用面向对象的程序设计,PrivilegedAction 的实现和调用都必须书写代码。采用上述方法,需要编写的代码大为减少,因为关注点被模块化到一个方面内。

图 5 中的方面通过一个实现了 PrivilegedExceptionAction 接口的匿名类遍历需要授权的每一个方法调用。通过在实现了的 run() 方法中插入 proceed() 方法,解决了所有有着不同数目类型参数,不同类型返回值的操作的包装问题。

```
public abstract aspect AbstractAuthAspect {
    .....
    //获取授权的方法
    public abstract Permission getPermission(JoinPoint.
        StaticPart joinPointStaticPart);
    //创建和执行工人对象的环境通知
    Object around() : authOperations() && ! cflow-
        below(authOperations()) {
        try {
            return Subject. doAsPrivileged (-authenticatedSubject, new Privi-
                legedExceptionAction() {
                    public Object run() throws Exception {
                        return proceed();
                    }
                }, null);
        } catch (PrivilegedActionException ex) {
            throw new AuthorizationException(ex.getException());
        }
    }
    before() : authOperations() { //许可检测
        AccessController. checkPermission ( getPermission ( thisJoin-
            PointStaticPart));
    }
}
```

图 5 增加了授权的抽象认证授权方面

4 原型系统

通过一个简化的网上书店系统来测试基于 AOP 技术的认证授权系统。在网上书店系统中,将系统的认证连接点设置在每个页面显示欢迎信息的方法 Welcome 方法,将授权连接点设置在下载书籍操作的 getDownUrl 方法中。整个工作流程如图 6 所示。

系统运行取得了理想的结果,页面的授权访问功能执行良好,AOP 的技术优势得到了很好的体现。通过原型系统运行表明,用 AOP 实现认证授权系统是可行的,并且是优势明显的,它使得系统核心代码的独立性更好,为以后的移植和维护提供了更大的便利。而且由此所牺牲的程序的执行效率并不明显,可以认为对系统效率影响很小。

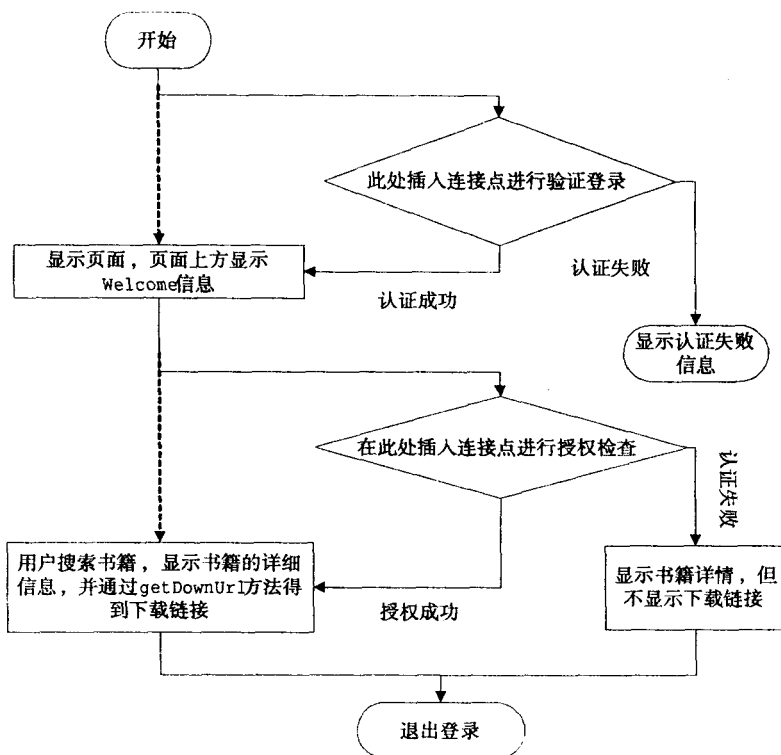


图 6 加入认证和授权的工作流程

5 结束语

介绍了信息系统中认证和授权的基本概念,以及基于 Java 技术的认证和授权的框架 JAAS。在基于传统面向对象的程序设计中,使用 JAAS 会导致认证和授权代码与业务逻辑的实现代码纠缠在一起,不利于重用和维护。基于面向方面的软件开发技术——AOP,利用方面封装现有的认证授权逻辑,从而导致业务逻辑和认证授权分离,提升了系统的重用性和可维护性。通过原型系统的实现,验证了 AOP 技术实现认证授权系统可行性,同时对程序的执行效率影响很小。

(下转第 190 页)

较大或者离开成员数 $L = 0$ 时,采用星形密钥图有较好的性能,有较小的更新开销。通过图 3 和图 4 可以得出相似的结论,就是当成员离开数 $L < N/4$ 或加入成员数 $J < N/2$ 时,密钥树比星形密钥图有好的性能,更新代价较低;其他的情况,星形密钥图有较好的性能,有低的更新代价。

4 结束语

文中分析树形结构的密钥图和星形结构的密钥图独立更新方式带来的更新代价,重点仿真星形和树形结构批量更新的性能,从而得出一个大致结论:当成员离开数 $L < N/4$ 或加入成员数 $J < N/2$ 时,采用树形结构有较好的性能,而采用星形结构有较好的更新性能。该结论对实际的应用有一定指导意义,从而降低更新密钥的代价,提高密钥服务器的性能。

参考文献:

- [1] Chung Kei Wong, Gouda M, Lam S S. Secure group communications using key graphs[C]//Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication. NY, USA: ACM, 1998:68-79.
- [2] 许勇,陈恺.安全多播中基于成员行为的 LKH 方法[J].软件学报,2005,16(4):601-608.
- [3] 蔡延荣,王清贤,李梅林,等.安全多播密钥更新研究[J].

计算机技术与自动化,2003,22(3):110-112.

- [4] David A, Grew M, Sherman T. Key Establishment in Large Dynamic Groups Using One - Way Function Trees[M]//IEEE Transactions on Software Engineering. NJ, USA: [s. n.], 2003:444-458.
- [5] Naor D, Naor M, Lotspiech J. Revocation and Tracing Schemes for Stateless Receivers[C]//Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London, UK: Springer - Verlag, 2001:41-62.
- [6] Setia S, Zhu S, Jajodia S. A Scalable and Reliable Key Distribution Protocol for Multicast Group Rekeying[R]. Virginia, US: Department of Information and Software Engineering, George Mason University, 2002:1-14.
- [7] 李彦希,赵耀,林闯,等.基于单向函数树的高效分布式组密钥管理方案[J].清华大学学报:自然科学版,2005,45(10):1417-1420.
- [8] 杨焱林.基于 LKH 混合树的多播密钥更新方案[J].现代电子技术,2004,27:31-32.
- [9] Heydari M H, Morales L, Sudborough I H. Efficient Algorithms for Batch Re - keying Operations in Secure Multicast[C]//Proceedings of the 39th Hawaii International Conference on System Sciences. [s. l.]: [s. n.], 2006.
- [10] 屈劲,葛建华,蒋铭.安全多播密钥批更新算法研究[J].电子学报,2003,31(7):1047-1048.
- [11] 赵欣,吴敏强,陈道蓄,等.一个自适应的安全组通信密钥更新算法[J].电子学报,2003,31(5):656-658.

(上接第 183 页)

- [6] 李玮,侯整风. SSL 协议安全缺陷分析[J].计算机技术与发展,2006,16(12):224-226.
- [7] Dandash O, WU Xiaoping, Le Phu Dung. Wireless Internet Payment System Using Smart Cards[C]//Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05). Nevada: IEEE Computer Society, 2005.
- [8] WU Xiaoping, Dandash O, Le Phu Dung. The Design and

Implementation of a Smart phone Payment System based on Limited - used Key generation scheme[C]//Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06). [s. l.]: IEEE Computer Society, 2006.

- [9] 邢宝书,李刚,薛惠锋.一次一密加密系统设计与实现[J].计算机技术与发展,2007,17(3):150-152.

(上接第 186 页)

参考文献:

- [1] 盛津芳,王斌,陈松乔.方面化构件模型及其组装方法[J].计算机工程,2006,32(5):39-40.
- [2] Hilsdale E, Hugunin J. Advice weaving in aspect[C]//Proc. of the 3rd International Conference on Aspect - Oriented Software Development (AOSD 2004). Lancaster, UK: ACM Press, 2004:26-35.
- [3] 陈成,李行.基于 AOP 的 MDA 模型转换[J].计算机技术与发展,2008,18(7):87-90.
- [4] 古全友,王恩波,胥昌胜. AOP 技术在 J2EE 系统构建中的

应用[J].计算机技术与发展,2006,16(4):150-152.

- [5] 钱竹青,邹正武. Eclipse AspectJ——利用 Eclipse 和 AspectJ 进行面向方面程序设计[M].北京:清华大学出版社, 2006.
- [6] Nicholas L. Using AspectJ Enhancing Design Patterns[EB/OL]. 2005. <http://www-128.ibm.com/developerworks/>.
- [7] 王斌,周亮,谭云桥,等.基于类修改和反射的动态方面编织模型[J].计算机工程与应用,2008,44(7):82-85.
- [8] SUN. Introduction to JAAS Authorization[EB/OL]. 2002-11. <http://java.sun.com/j2se/1.4.2/docs/guide/>.