构件运行支撑平台反射体系的安全框架设计与实现

白 佳, 黄 罡, 刘 钊, 刘天成, 郑子瞻, 梅 宏
(北京大学信息科学技术学院软件研究所, 北京 100871)

摘 要: 反射式软件中间件改变了传统中间件纯粹的"黑盒复用"方式, 以观测和控制基于中间件的软件系统的运行状态和行为. 作为主流的中间件产品, 构件运行支撑平台有必要引入反射以适应动态开放的 Internet 环境. 但是, 在带来更大开放性的同时, 反射也给构件运行支撑平台带来安全隐患. 为此, 针对反射体系的特点, 本文逐层分析其中潜在的安全隐患, 制定了一种特定于反射体系的安全框架, 实现了四层安全访问控制机制. 该安全框架在反射式 J2EE 应用服务器 PKUAS(Peking University Application Server) 中得到实现, 并通过性能测试考察了安全框架对反射系统运行时刻性能的影响.

关键词: 反射式中间件; 构件运行支撑平台; 安全; J2EE
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2004) 12A-2207-04

An Access Control Framework for Reflective J2EE Application Server

BAI Jia, HUANG Gang, LIU Zhao, LIU Tian-cheng, ZHENG Zi-zhan, MEI Hong
(*Department of Computer Science and Technology, Peking University, Beijing* 100871, *China*)

Abstract: Reflective middleware opens up the implementation details of middleware platform and applications at runtime to improve the adaptability of traditional middleware. However, such openness brings new threats to security of middleware platform and applications. This paper studies how to protect a reflective J2EE application server with a set of access control mechanisms. At first, a computation model of reflective middleware is built up and illustrates that the access control of reflective middleware is far more complex and difficult to implement than that of traditional middleware. With the model, all potential access control points are identified while only some of the points require access control mechanisms. It reveals that the complexity and cost of the access control framework are mainly related to the concrete implementation of reflective mechanisms. At last, the framework is implemented mainly by reusing the access control mechanisms existing in traditional middleware and the performance is evaluated.

Key words: reflective middleware; access control; J2EE(Java 2 platform, enterprise edition)

1 引言

Internet 环境的开放、动态和多变的特性, 要求运行其上的软件系统具有开放的适应性[8]. 作为 Internet 环境下的主流基础设施, 软件中间件因其传统的"黑盒复用"思想, 屏蔽了底层运行环境及中间件自身的变化, 难以实现开放的适应性. 目前, 解决这一难题的主流途径是将反射性引入中间件的构造、使用与管理, 形成了中间件的研究热点之一——反射式中间件[1]. 反射式中间件是一种通过与自身状态和行为具有因果关联的系统自述来描述、推理和操纵自身的中间件, 而且, 描述、推理和操纵自身的方式与中间件描述、推理和操纵其问题域的方式类似[6]. 反射式中间件通过引入反射性来辅助开发适应性强的应用系统, 它的引入增强了对中间件平台的内部状态和行为的观测和调整能力, 提高了中间件平台的易用性.

如图 1 所示, 一个反射系统具有两个组成部分: 一部分对系统的问题域建模并对其进行推理和操纵, 以解决问题; 另一部分对系统自身建模并对其进行推理和操纵, 使系统适应某些变化. 而且, 两个部分采用同样的实现基理. 从实现角度看, 前一部分组成基层实体, 后一部分组成元层实体, 基层实体与元层实体之间具有因果关联(causal connection), 即, 基层实体状态或行为的任何变化均立即导致相应元层实体的变化, 反之亦然[5].

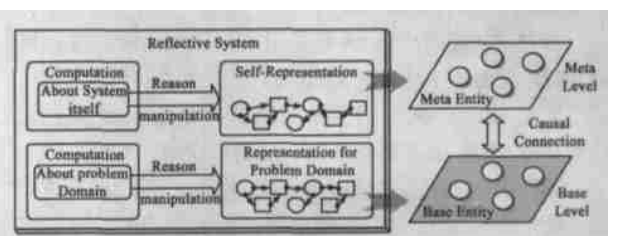

图 1 反射系统示意图

由上图可以看到, 反射体系提供了一种自身建模的机制, 通过元层实体与基层实体的反射关系, 在传统的"封闭式"黑盒体系上打开了若干窗口, 开放系统的内部细节, 大大提升了系统的开放程度. 但是, 开放和安全往往是矛盾的, 在越开放的应用系统中, 人们不得不越重视安全问题. 反射式中间件允

收稿日期: 2004-09-02; 修回日期: 2004-11-26
基金项目: 国家重点基础研究发展规划项目计划(973 计划) (No. 2002CB312003); 国家自然科学基金(No. 60233010, No. 60125206, No. 60303004); 国家高技术研究发展计划(863 计划) (No. 2004AA112070); IBM 大学合作项目

许用户观测和调整系统的状态和行为,这意味着不仅可以通过非正常渠道调用业务功能,还能通过恶意代码来修改系统使得业务用户在不知不觉中受损. 设想在某应用中嵌入恶意代码,该代码试图通过元层实体操作并替换系统某基层实体,以达到破坏的目的. 除非仔细研读每一个将要部署的应用的源码,否则无法确定一个应用是否存有恶意. 然而,让应用服务器管理人员通读所有应用的源码并不现实,因此,有必要针对反射的特点修补安全漏洞.

本文以反射式 J2EE 应用服务器 PKUAS[9] 为实验平台,深入全面地分析了反射体系存在的安全隐患,制定了相应的安全访问控制策略,通过集成现有的 Java 和 J2EE 安全技术,形成一种完备的反射体系安全框架,旨在解决如何在保留反射式中间件系统开放性、适应性等优点的同时,使反射式中间件系统可以像传统中间件系统那样时刻处于安全的状态,并给系统带来较小的性能损耗. 需要一提的是,现有中间件的安全体系并不是绝对安全的,本文只尝试将反射式中间件的安全提升到与传统中间件相当的级别,以确保反射体系的实用性,超越现有中间件安全级别并不是本文的研究范围.

文章余下部分组织如下:第二章概述反射体系安全框架;第三章介绍该安全框架在 PKUAS 中的实现及其性能影响;第四章介绍相关工作;最后总结全文并展望下一步的工作.

2 方法概述

不同于普通的 J2EE 应用服务器,反射式应用服务器由于其开放性带来了新的安全隐患. 以 PKUAS 为例,在一次实际的访问中,客户首先访问反射系统对外提供的接口 MEJB (Management EJB),该接口将请求转发给相应的元层实体,该元层实体进而访问基层实体的信息,并返回给客户. 由于反射为系统增添了元层以及反射访问接口,因此和元层实体及反射访问接口相关的任何访问,都需要加以控制才能达到预期目标. 在上述过程中,有四个层次的安全问题与此相关:客户对 MEJB 的访问权限及 MEJB 对指定元层实体的访问,元层实体之间的访问,元层实体对基层实体的访问,以及 Java 类对本地资源的访问,如图 2 所示:

客户访问控制:PKUAS 定义了用户访问反射系统的接口 MEJB,它是一个普通的无态会话 EJB,作为一个系统级应用部署在 PKUAS 中. 反射体系的用户只能通过 MEJB 访问反射体系,因此,首先必须在这一层进行控制.

元层实体之间的访问控制:元层实体之间的相互访问也可能给系统带来破坏. 如果中间件增加了新的服务或容器类型,需要提供相应的元层实体. 这意味着系统中的元层实体可能来自不同的厂商,因此,必须在元层实体之间进行访问控制.

元层实体访问基层实体的安全控制:为了防止元层实体对基层实体的恶意操作或误操作,一方面,需要禁止一个元层实体绕过另一个元层实体而直接反射后者的基层实体. 另一方面,需要考虑元层实体对自身直接反射的基层实体的访问权限.

本地资源访问控制:需要控制 Java 对象对本地资源的使用,如,禁止创建线程或控制线程状态,禁止建立网络连接等.

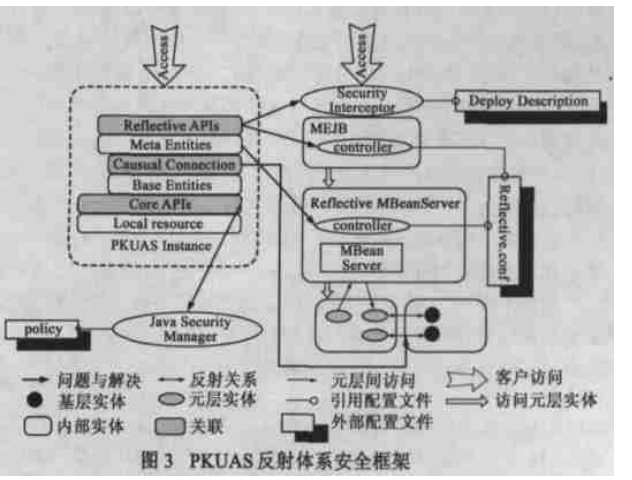

图 3 PKUAS 反射体系安全框架

3 实现和性能评测

PKUAS 是一个反射式构件运行支撑平台. 对于上面提出的四个层次的安全隐患,下面将依次给出在 PKUAS 中的解决方案及其实现. 最后,通过反射系统的性能测试,评估了安全框架对系统整体性能的影响.

3.1 PKUAS 概述

PKUAS(Peking University Application Server)是北京大学软件研究所自主研发、遵循 EJB2.0 规范的反射式构件运行支撑平台. PKUAS 提供了反射机制,并控制反射的内容、时机、及其正确性. 通过抽取一组基本功能形成一个微内核,PKUAS 将平台内部的其他功能封装在各个相对独立的构件内,允许用户根据领域特征定制与扩展这些系统构件,该微内核被实现为 JMX MBeanServer. 利用微内核的管理机制,PKUAS 方便地引入元层实体,同时做到了在不改变构件化平台结构的情况下形成元层实体与基层实体的和谐共处[6,9,10].

3.2 PKUAS 反射体系安全框架的实现

针对上面提出的问题,PKUAS 着重于在四个层次解决反射带来的安全问题. 整体设计框架如下图所示:

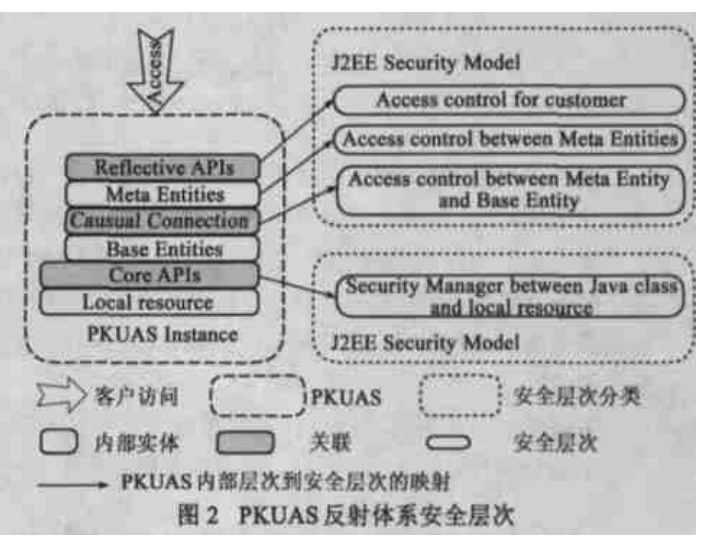

图 2 PKUAS 反射体系安全层次

从上图中可以看出一次反射请求在 PKUAS 内部的传递及处理方式. 客户请求首先发送给 PKUAS 反射体系对外的接口 MEJB, 此时直接利用 J2EE 安全访问控制机制以及在 MEJB 内部的 Controller 类来控制客户访问. 访问被许可后发送给相应元层实体, 该元层实体根据反射访问其对应的基层实体或通过 ReflectiveMBeanServer 类访问其他元层实体, 进而访问其基层实体. 下面进行详细分析并给出解决方案.

3.2.1 客户访问控制　客户访问可以通过 PKUAS 的安全截取器进行控制. 但由于 MEJB 的方法粒度较粗, 安全截取器只能对 MEJB 的某个方法(如 getAttribute)进行访问控制, 而不能针对具体的元层实体提供不同的访问权限, 因此该方法并不完善. 更加合理的方式是在 MEJB 的业务逻辑中加入控制机制, 使得 MEJB 可以为不同的元层实体提供不同级别的访问控制. 如可以针对某一个元层实体指定是否可以对它进行查询属性的操作. 为达到该粒度上的控制, 在 MEJB 中嵌入了一个 Controller 类, 该类实现了类似于安全截取器的功能, 在 MEJB 接收到客户请求后, 该类截获请求, 然后根据请求内容, 查询配置文件 Reflective. conf, 以判断是否可以访问. 根据 Reflective. conf 的描述, Controller 决定是否把请求转发给相应的元层实体的方法并返回; 或者抛出访问异常.

下面给出 Reflective. conf 的一个例子, 说明如何定义一个具体 Meta Entity 方法级别上的访问控制.

```
〈CONFIG〉
〈method-permission〉
〈ATTRIBUTE NAME="role-name" VALUE="admin"/〉
〈ATTRIBUTE NAME="bean-name" VALUE="AppName.EntityTest"/〉
〈ATTRIBUTE NAME="method-name" VALUE="getAttribute"/〉
〈/method-permission〉
〈/CONFIG〉
```

这里 Reflective. conf 制定了三个必要元素: 角色, Meta Entity 名和方法名, 只有在 Reflective. conf 中定义过的访问, 才可以通过检查, 否则都会抛出访问异常.

3.2.2 元层实体之间的访问控制　JMX1.2 规范增加了 MBean 之间的访问控制. 基于 Java 的安全管理器, MBeanServer 可以对 MBeanServer 提供的方法以及 MBean 定义的方法进行访问控制. 但是, Java 安全管理器的开启将使 Java 应用的性能至少下降 13% ~ 14%[2], 因此, 本文通过在 MBeanServer 外包裹一层 ReflectiveMBeanServer 类来实现访问控制, 以替代 JMX 缺省的安全访问控制机制.

PKUAS 的类装载机制决定了一个元层实体不能直接访问另一个元层实体, 而需要通过 JMX 提供的 MBeanServer 进行调用. 通过在 MBeanServer 外包裹一层 ReflectiveMBeanServer, 就可以在请求转发至 MBeanServer 前进行权限判断. 当元层实体访问另一个元层实体时, 首先将请求以及自身角色信息传给 ReflectiveMBeanServer, ReflectiveMBeanServer 中内嵌的 Controller 类读取配置文件 Reflective. conf 判断该角色是否有权限进行本次访问, 根据判断的结果将请求转发给 JMX 的 MBeanServer 或者抛出访问异常. 值得一提的是, 客户访问控制和元层实体之间的访问控制两部分复用同一个权限配置文件与 Controller 类, 简化了反射体系安全框架的使用.

3.2.3 元层实体访问基层实体的安全控制由于元层实体往往由基层实体提供者开发, 因而不需要考虑直接反射基层实体的元层实体是否执行恶意操作. 但是, 如果一个元层实体能够绕过另一个元层实体而直接访问那个元层实体反射的基层实体, 安全依然无法得到保证.

PKUAS 的类装载机制很好的解决了这个问题, 它保证只有服务、容器系统或容器的元层实体才能使用其基层实体的类装载器[9], 而一个对象无法直接调用被其他类装载器装载的对象, 因此, 只有通过 MBeanServer, 一个元层实体才能访问另一个元层实体, 进而反射其基层实体. 这样保证了元层实体对基层实体的访问安全.

3.2.4 本地资源访问控制　Java 的安全管理器在这方面做了许多工作, 可以直接使用. 开启 Java 安全管理器后将自动控制 Java 对象对本地资源的使用, 包括磁盘读写以及网络资源的占用, 如创建线程以及建立网络连接等. 但是使用安全管理器需要管理员提供访问权限的 policy 文件, 为了使 PKUAS 以及部署在上面的应用能够正常工作, 该 policy 文件必须包含详尽的磁盘读写、网络资源占用等权限描述, 工作量较大. 同时, 由于开启安全管理器将对系统性能产生较大负面影响, 因此 PKUAS 缺省地关闭安全管理器.

3.3 性能评测

无疑, 安全框架将对 PKUAS 带来较大的性能影响, 下面通过 PKUAS 的管理工具进行考察. 测试平台是 PIV 2.8GHz, 512M SDRAM 的 PC, 操作系统是 Windows XP.

安全框架所提的四个层次中, PKUAS 的安全截取器在正常业务过程中始终监控客户请求, 元层实体对基层实体的访问控制依赖于 PKUAS 的类装载机制, 这两层安全访问控制均不为反射体系带来额外开销, 无需评测其性能影响. 另外, 如前所述, 本地资源访问控制需要开启 Java 安全管理器, 性能损耗过大, 实际应用较难, 因此也不进行测试. 综上, 本文的测试仅覆盖"元层间访问控制"、"客户对 MEJB 访问的控制"以及两者同时开启的情况.

本测试以获取所有已部署的应用名称为例, 在测试中 PKUAS 上部署三个应用, 并在 Reflective. conf 中加入 100 条安全许可(其中获取应用名称的许可在最后一条). 在 10 次管理页面的刷新过程中, 获取已部署应用的时间耗费的平均值如图 4 所示:

图 4　性能测试结果

从以上结果可以看出, 开启安全检查将对 PKUAS 性能带来一定影响. 其额外的时间开销主要在于初始化安全框架时装载配置文件到内存, 以及每次检查权限时在内存中查询访问许可, 其中后者对性能的影响远大于前者. 因此, 该时间开

销受制于配置文件大小以及该许可在配置文件中的前后位置. 在本测试用例中, 一次初始化工作以及遍历内存的时间开销约 25 毫秒左右.

4 相关工作

早在 1990 年召开的第一届反射体系国际研讨会上就已经达成共识[4]:“反射引入了一种新的安全威胁, 如果不进行合理控制, 将带来严重后果.” 但是, 直至今日, 人们的关注点仍然集中在反射体系本身, 而反射体系的安全考虑得较少.

Guarana[7] 是 Java 语言的反射扩展, 修改了 JVM 以支持在 Java 对象创建时为其分配唯一的组装器, 进而由该组装器建立元层对象与 Java 对象之间的因果关联. 由于组装器是基层对象的唯一访问点, 因此可以决定元层对象反射基层对象的权力. 这与 PKUAS 反射体系安全框架中元层实体与基层实体之间的访问控制类似, 因为一个基层实体只能有一个可直接反射该基层实体的元层实体, 其他元层实体必须间接通过该元层实体才能访问基层实体, 此时的负责直接反射基层实体的元层实体就是 Guarana 中的组装器.

Caromel 等人[2] 利用 Java 语言的安全管理器控制元层实体对基层实体的访问, 这与 PKUAS 反射体系安全框架中本地资源访问控制类似. 如前所述, 由于安全管理器作用于所有基层实体, 这意味着对基层实体的所有访问都要经过安全管理器的检查. 采用安全管理器之后的应用性能降低 13% ~ 4%. 这严重影响了基层实体之间的正常业务计算, 因而实用价值值得商榷. PKUAS 缺省禁止使用安全管理器. JBoss[3] 利用 JMX 实现了与反射类似的管理框架, 但仅仅考虑了客户对这个管理框架的访问控制, 并没有考虑管理框架内部的安全漏洞. 特别地, JBoss 允许用户定义并实现自己的元对象并插装入管理框架, 这意味着恶意代码可以避开 JBoss 的访问控制机制进行任意的破坏.

简言之, 与上述工作相比, PKUAS 对整个系统的反射框架进行了全面完整的访问控制.

5 结束语

针对开放、动态、多变的 Internet 环境, 反射式构件运行支撑平台为用户提供了一种运行时观测、控制整个系统的手段, 在增强系统开放性的同时, 也带来了新的安全隐患. 因此, 必须认真考虑反射体系带来安全漏洞, 并尝试解决之.

本文从反射体系的特点出发, 分析了反射式构件运行支撑平台的四个层次上的安全隐患, 依次探讨了每一个安全隐患可能的解决思路, 并在一个反射式构件运行支撑平台——PKUAS 上进行了实现, 最后对比了安全机制带来的性能影响, 证明了安全机制的实用性.

但是, 仅通过角色来控制访问权限并不是完备的做法, 如果一个应用伪造自身角色, 现在的安全框架是不能识别的. 因此需要更加有效的身份认证机制, 并在客户请求中传递该身份标识, 达到逐层安全控制的目的. 同时, 为提高性能, 安全模块应能够快速定位安全许可, 这可以通过对许可进行分类实现. 安全模块首先定位相应的大类, 再逐一察看该类别下的安全许可, 而不必察看所有许可条目. 在安全许可条目较多的情况下, 这能够显著地减少性能损耗. 以上述性能测试为例, 察看 100 条安全许可花费的时间约 23 毫秒, 若通过分类把必须察看的条目减少到 20 条, 则三种开启安全机制情况下的时间消耗将依次减少约 16、16 和 32 毫秒.

参考文献:

[1] Blair G S, G Coulson, P Robin, M Papathomas. An architecture for next generation middleware[A]. Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware' 98)[C]. Lake District, UK, Editors: Davies N, Raymond, K, Seitz J, Springer-Verlag, 1998. 191- 206.

[2] Caromel D, J Vayssiere. A security framework for reflective Java applications[J]. Software Practice and Experience, 2003. 33. 821- 846.

[3] Fleury M, Reverbel F. The JBoss extensible server[J]. Proceedings of IFIP/ACM Middleware 2003, 2003, LNCS 2672, 344- 373.

[4] Ibrahim M H. Report of the first workshop on reflection and metalevel architectures in object-oriented programming[A]. OOPSLA/ECOOP' 90[C]. Ottawa, Canada, October 1990.

[5] Maes P. Concepts and experiments in computational reflection[A], Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA' 87)[C]. Orlando, FL USA, October 1987. 147- 155.

[6] Mei H, G Huang. PKUAS: An architecture-based reflective component operating platform[A]. invited paper, 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS)[C]. Suzhou, China, 26- 28 May 2004.

[7] Oliva A, L E Buzato. The design and implementation of guaraná[A]. 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS' 99)[C/OL]. http://www.ic.unicamp.br/~oliva/guarana/docs/desimpl.ps.gz.

[8] Oreizy P, M M Gorlick, R N Taylor, et al. An architecture-based approach to self-adaptive software[J]. IEEE Intelligent Systems, May/June 1999. 54- 62.

[9] 黄罡, 梅宏, 杨芙清. 基于反射式软件中间件的运行时软件体系结构[J]. 中国科学, E 辑, 技术科学, 2004, 34(2): 121- 138.

[10] 黄罡, 王千祥, 曹东刚, 梅宏. PKUAS: 一种面向领域的构件运行支撑平台, 电子学报, 2002, 30(12A): 1938- 1942.
HUANG Gang, et al. PKUAS: A domain-oriented component operating platform[J]. ACTA ELECTRONICA SINICA, 2002, 30(12A): 1938- 1942.

作者简介:

白 佳 男, 硕士生, 主要研究领域为软件构件和分布计算技术.

黄 罡 男, 博士, 讲师, 主要研究领域为软件工程、软件构件和分布计算技术. E-mail: huanggang@sei.pku.edu.cn.