广西科学院学报
Journal of Guangxi Academy of Sciences
2006,22(4):246～249
Vol. 22,No. 4 November 2006

Java 安全体系结构研究
A Brief of Security Architecture of Java

蒋曹清
JIANG Cao-qing

(广西财经学院计算机与信息管理系,广西南宁 530003)
(Department of Computer and Information Management, Guangxi University of Financial and Economics, Nanning, Guangxi, 530003, China)

摘要:全面介绍 Java 安全体系结构的组成部分,并以一个实例来说明 Java 安全体系结构中各个组成部分为 Java 程序提供的安全性。

关键词:Java 安全 体系结构 Java 虚拟机
中图法分类号:TP311 **文献标识码**:A **文章编号**:1002-7378(2006)04-0246-04

Abstract: The security architecture and components of Java are discussed using an instance.

Key words: Java, security, architecture, Java virtual machine

随着 Intranet 和 Internet 中分布式应用的日益增多,计算机网络安全问题已经引起大家的高度关注,因此分布式应用必须解决安全问题。Java 是目前流行的网络编程语言,由于其平台无关性,目前在金融机构、电子商务软件、网络数据库应用和其它关键应用程序的部件中都得到了普遍应用,Java 已成为分布式应用编程的最佳工具[1]。随着越来越多复杂的分布式应用的成功实现,安全问题变得特别重要。本文全面介绍 Java 安全体系结构的组成部分,并以一个实例来说明 Java 安全体系结构中各组成部分为 Java 程序提供的安全性。

1 Java 体系结构

Java 采用了一个扩展的内置安全模型——"沙箱",解决由 Java 运行环境中可能存在的安全问题。它的核心思想是:本地环境中的代码能够访问系统中的关键资源(如文件系统等),而从远程下载的程序则只能访问沙箱模型内的有限资源。Java 1.0 提供了一个专用的运行 Java 程序的沙箱。沙箱内可做任何事情,但超出此边界就不能有任何操作。Java 1.1 采用的安全模型是信任模型,用户可以有选择地授权给远程代码,如果远程某地资源可以被信任,那么带有此地签名的代码将可以访问本地的系统资源,其它不被信任的代码仍受限在沙箱内。Java 2 采用了域管理模型,无论是本地代码或远程代码,都可以通过配置来设定可访问的资源,从而更好的支持企业级应用[2,3]。

Java 沙箱安全模型是 Java 安全体系核心部分,Java 沙箱安全模型包括 Java 虚拟机安全性和安全管理器及 Java API[4,5]。除此以外,Java 的安全体系(如图 1)还包含在 Java 语言中提供一系列安全工具及服务,以便于企业级应用的安全性的实施。

1.1 Java 虚拟机的安全性

Java 虚拟机为 Java 分布式应用提供了一个安全稳定的 Java 平台,使得 Java 分布式程序可以安全地运行。Java 虚拟机的安全性由 Java 语言自身、字节码校验器、类装载器具备的安全性来体现。

1.1.1 内置的安全性

Java 是一种基于语言的安全模式,是内置于 Java 虚拟机中的安全特性。这些安全性是通过编译器的编译来实现,即编译成功则说明达到了语言层安全性。语言层的安全性是 Java 安全最基本的要素,它包括以下几个方面。(1)安全的类型转换。Java 是一种强类型语言,也就是说变量在编译时和运行时的类型是一致的。这也意味着类型转变在编译和运行时都得检查是否合法,从而防止恶意代码逃避权限控制。在 Java 中,不能通过指针进行任意的类

收稿日期:2006-07-03
作者简介:蒋曹清(1973-),男,讲师,主要从事计算机软件方面工作。

型转换,运行时系统对对象的处理要进行类型相容性检查,以防止不安全的转换。(2)不支持指针类型。Java 程序不能定义它们自己的内存指针或直接防问物理内存,这阻止了程序访问和修改安全系统的关键部分。类内方法和实例化的变量都是通过符号名来引用。程序员不能对指针进行算术运算,避免了蓄意或失误地滥用指针。(3)自动的垃圾收集。所有的数据结构都是对象,通过 new 为它们分配内存堆。Java 可对对象进行自动的管理并进行垃圾收集,有效地减少软件出现异常的概率,充分地利用系统资源,避免内存漏洞。(4)内存使用的安全性。在 Java 语言中,Java 编译器不决定程序代码的内存布局,而是由 Java 运行系统决定,并且 Java 运行系统所基于的操作系统不同,其内存布局也是有所不同的。这种在 Java 运行时才决定系统的内存布局的方式,就不会预先得知内存布局的信息,程序员也不可以随意更改内存单元的值,这就使利用内存布局来破坏系统成为不可能。(5)其它安全特性。若程序中有数组或字符串访问, Java 虚拟机就会检查访问是否越界,从而防止利用越界数组访问产生的安全问题。final 类和方法声明可以防止对已验证过的可信任代码的修改等。

1.1.2 类装载器

类装载器提供安全的第一道防线,起了很重要的作用。当一个分布式程序从网络上引入时,Java 虚拟机要调用类装载器。类装载器从网络上获取分布式程序的可执行代码,并为下载的代码创建命名空间。命名空间是由一个特定的类装载器装载的类名的集合。分布式程序为每个类装载器维持一个命名空间,这个命名空间里有由那个类装载器装载的所有类的名字。同一命名空间中的类可以互访。但是,由不同类装载器装载的类分别放在不同的命名空间里,这些类不能彼此访问。类装载器通过命名空间来隔离不同来源的类文件,有效地防止不安全代码访问、破坏安全代码。

类装载器通过分别使用不同的类装载器装载可靠的包和不可靠包来保护被信任的类库的边界。虽然通过赋给类成员受保护(或包访问)的访问权限,可以在同一个包中的类型间授予彼此访问的特殊权限,但这种特殊的权限只能授给在同一个包中运行时成员,而且它们必须是由同一个类装载器装载的。除了类装载器通过剔除装作被信任的不可靠类,来保护可信任类库的边界。另一种保护被信任的类库的方法是它只需通过简单地拒绝装载特定的禁止类型就可以了。

1.1.3 字节码校验器

字节码校验器在字节码执行之前,必须独立地检查每个方法的代码,此检查需要 4 次独立的扫描来完成。第一次检查类文件的格式。检查方法的正确定义、属性的长度是否合适、字节码的长度在合适的范围、常量池(Constant Pool)是否能够被分析。第 2 次检查类型数据的语义。检查那些不用分析字节码就可以验证对错的地方,主要是一些语法级的检查。它包括:final 类不能被继承或重载;每个类必须要有一个超类;常量池必须满足更严格的限制条件;常量池中关于属性和方法的引用必须要有合法的类名、属性名、方法名或者合适的签名。第 3 次用字节流分析法验证字节码的正确性。对指定的字节码程序任何给定点,不管这一点如何到达,必须做到:(1)栈的大小一致;(2)寄存器的存取要进行合适的类型检查;(3)属性域被修改成合适的类型;(4)所有的操作码要有合适的参数,或者在栈上,或者在寄存器中。第 4 次加载将要用到但是还没有用到的类的定义,并验证当前类是否允许引用新加载的类。这将导致对相应操作码的重写并加上快速标记,以便于以后加载该类时可以快速加载,从而提高运行速度。

通过字节码验证器验证后的代码满足:不伪造指针;不违反权限控制;忠实地访问对象; 以正确类型的正确参数来调用方法;没有栈溢出;没有非法的数据类型转换。

1.2 安全管理器及 Java API

一旦原始的类装载器或自定义的类装载器装入一个类并由验证器验证了它,Java 平台的第 3 种安全机制(即安全管理器)就开始运行。安全管理器负责说明一个安全策略以及执行这个安全策略。

Java API 在进行一个可能不安全的操作前,总是检查安全管理器,故 Java API 不会在安全管理器建立的安全策略之下执行被禁止的操作。

运行 Java 应用程序时,缺省的做法是不运行任何安全管理器,但和运行 Java 应用程序不同,运行小应用程序时会立即装载一个安全管理器。安全管理器执行必要的权限检查工作的过程是:(1)如果程序代码需要调用需要相应权限才可执行的代码,代码就调用安全管理器检查权限;(2)安全管理器决定权限是否允许,这通常会与 java.security.AccessController 交互。如果没有权限,就会出现运行异常:java.lang.SecurityException,否则不会返回异常而正常执行。安全管理器会监督整

个执行线程以保证每个类都有相应的权限；(3)如果没有异常，说明有足够的权限，程序将继续运行。

尽管运行一个 Java 应用程序时，缺省的做法是不运行安全管理器，但是程序可以调用 System 类的 setSecurityManager()方法安装一个指定的安全管理器。一旦程序安装了一个安全管理器，除非这个安全管理器同意替换它自身，否则，其它安全管理器就不能安装。这一点很重要，否则一个有害的小应用程序就可以安装自己的安全管理器。因而，系统中可以有多个类装入器，但每个 Java 程序只接受一个安全管理器的管理。是否给予所有类相同的访问权限，取决于这个安全管理器的实现方法。

1.3 安全服务及工具

安全服务及工具扩展了 Java 的安全体系结构。随着安全服务和工具包的增多，Java 分布式应用的安全性越来越容易得到实现。

1.3.1 安全服务

常见的安全服务有：Code source 类、权限类 Permissions、安全策略 Policy 类和密码库 Keystore 类。

(1)CodeSource 类。由于 Java 代码可以从网络上的任何地方下载，因而代码的来源对维护其安全性至关重要。Java.security.CodeSource 类对象完整地描述了这些特征。CodeSource 包含了代码的来源(URL)以及含有公钥的数字授权证书，用来对获得的签名代码进行校验。没一个 Certificate 即是一个 Java.security.cert.Certificate 对象，每一个 URL 则由 java.net.URL 来代表。

(2)权限类 Permissions。Permissions 类是 Java 安全的核心，代表对不同的资源如文件、Socket 等的访问。一系列权限可以组成适当的安全策略。例如，下面的句子将使/temp 目录下的文件“abc”具有可读的权限：Perm = new java.to.FilePermission(“/temp/abc”,“read”)。

(3)安全策略 Policy 类。Java 2 允许用户对待特定的程序建立详细的安全策略。安全策略在系统中以 policy 对象来表示，该对象描述允许程序执行的与安全相关的操作。执行诸如：检查是否可以在某端口上创建与另一个远端系统建立 Socket 连接；检查是否可以删除、读取或写入一个本地文件；检查是否可以执行本地系统的一个程序；检查是否可以侦听某个网络端口上的连接请求；标识可以使用 System.getProperty()方法访问的系统属性等的方法在 java.lang.SecurityManager 类实现。这就使得与安全有关的任务的执行被限制在系统根据安全策略可以控制的范围内。

(4)密码库 Keystore 类。Keystore 是一个密码保护的数据库，它存放了私有密钥及相关的数字证书，用来对公钥进行验证。密码是在记录产生时选择好的，数据库中的每条记录可以拥有自己的密码。Keystore 中存放的数字证书(公钥)被认为是可信的。

1.3.2 安全工具

常见的安全工具有 Keytool，Jarsigner 和 Policytool。Keytool 用来管理 Keystore 数据库，它可以产生公钥私钥对，发送证书请求，导入证书，指定第三方公钥的可信性。Jarsigner 用来产生和验证 JAR 签名，当需要用私钥及相应的证书为文件签名时，Jarsigner 便会查找 Keystore 数据库；由于 Keystore 库和私钥都是密码保护的，因而只能知道密码的人才能访问到特定的私钥并进行签名。Policytool 命令生成和管理安装安全策略的配置文件，用户也可以根据需要更改系统安全策略文件或创建新的用户安全策略文件。

2 实例说明

以一个分布式程序为例在图 1 中的编译阶段，首先根据分析和设计的要求编写源程序，接着用 Java 编译器进行编译，这时根据 Java 内置的安全性检查程序，如果有错误或不安全的地方，编译器会报告错误；如果正确则生成二进制字节码文件。当要使用程序时，则通过网络从远端传输到本地机。

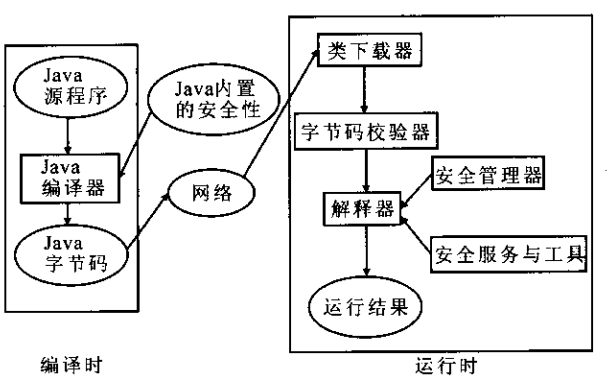

图 1 Java 安全体系结构

在图 1 中运行阶段的类下载时，检查从哪里下载文件，是否有权限进到本机系统，根据类名建立相应的命名空间。接着进行字节码校验，此时要分析下载的字节码是否合乎规则。如果字节码的格式不合要求，则拒绝执行。完全符合规则的字节码才允许执行。如果字节码校验时没有错误，则 Java 解释器马

上执行字节码程序,这时候,安全管理器始终监测所执行的没一步操作,检查其合法性,安全服务和工具按预先的设置实现其安全功能。

3 结束语

在 Java 安全体系结构中,Java 虚拟机内置了许多安全特性,在语言层确保 Java 应用程序的安全性,而且在字节码运行前通过字节码校验器和类装载器保证代码的安全性,运行时,实现预先定制的 Java 安全服务和工具的功能,并通过安全管理器始终监控 Java 程序的运行。Java 安全系统结构的各个组成部分都能为 Java 程序提供安全性。

参考文献:

[1] TRENT J,JOCHEN L,VSEVOLOD P,et al. Security architecture for component-based operating systems [M]. ACM SIGOPS European Workshop,1998:222-228.

[2] BILL VENNERS. 深入 Java 虚拟机[M]. 曹晓刚,蒋靖,译. 第 2 版. 北京:机械工业出版社,2003:25-72.

[3] SUN Microsystems Inc. The Java virtual machine specification [EB/OL]. [2006-06-10] 2nd Edition. http://java.sun.com/docs/books/vmspec/.

[4] LI GONG. JavaTM 2 Platform Security Architecture [EB/OL]. [2006-06-10]. http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security-spec.doc.html.

[5] 辛运帏,饶一梅,张钧. Java 程序设计[M]. 北京:清华大学出版社,2004:3-6.

(责任编辑:邓大玉)

(上接第 245 页)

3 结束语

本文选择 OWL 描述本体,并用它实现对 Web 服务的标准 UDDI 的扩充。该方法用本体来描述 Web 服务的语义信息,使得 Web 服务中包含了适当的语义信息,从而服务发现就可以利用服务中的语义信息,这将大大提高 Web 服务发现的查全率和查准率。下一步我们将开发更丰富的本体来描述操作、前提和结果的功能;研究服务发现算法以便更充分地利用 WSDL 和 UDDI 中的语义,其中将重点研究根据匹配相似度排列查询结果的最佳算法。

参考文献:

[1] BERNERS-LEE T,HENDLER J,LASSILA O. The Semantic Web[J]. Scientific American,2001,284(5):34-43.

[2] BORST W N. Construction of engineering ontologies for knowledge sharing and reuse[D]. PhD thesis,Tweente University,1997.

[3] 柴晓路. tModel 的用途及结构详解[EB/OL]. 2001 [2006-07-15]. http://www-128.ibm.com/developerworks/cn/webservices/ws-tmodel/part/index.shtml.

[4] PAOLUCCI MASSIMO,TAKAHIRO KAWAMURA,TERRY R PAYNE,et al. Importing the Semantic Web in UDDI: Proceedings of Web Services,Ebusiness and Semantic Web Workshop[C]. London:Springer-Verlag,2002:225-236.

(责任编辑:凌汉恩 邓大玉)