安全管理器中的 SNMP 代理

张　明　余祥宣　詹建桥　（武汉华中理工大学 IBM 技术中心　430074）

摘要：Java 2 提出了一种可以根据安全策略实施配置，从而进行更细致的存取控制的安全模型。然而一旦安全策略配置不当，就会在运行中形成安全漏洞。本文在介绍 Java 安全模型的基础之上，分析了新的安全模型可能造成的管理复杂的问题，并提出了一种利用 SNMP 协议，在安全管理器中增加代理，实施整体安全控制的机制。通过实施这种机制，可以有效的保证安全策略的正确配置，降低安全漏洞出现的可能性。

关键词：Java　安全管理器　SNMP　代理

一、介绍

Java 语言从出现开始，就从以下两个方面提高系统的安全性：

（1）Java 语言本身的安全性，包括类型安全、代码检查以及语言的安全模型等；

（2）在 Java 语言中实现和提供一系列的安全工具和服务，以保证在其基础上开发的应用系统具有足够的安全性。

1. Java 最初的安全模型

Java 语言最初的安全模型被称为“沙箱模型”。其基本思想是对从开放网络上取得的不可信代码进行严格的安全控制，而本地运行的应用则被认为是可信的。这种模型严格而简单，在 JDK1.0.X 中实现。

在这种“沙箱模型”下，从网络上下载的小应用程序（Applet）无法存取本地的任何资源，这一方面虽然保证了防止网络上的恶意代码攻击，但另一方面也限制了 Java 语言在 Internet 和 Intranet 上的应用。

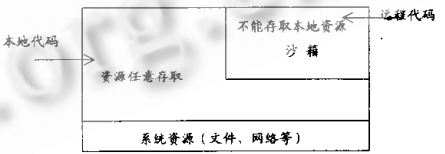

图 1　JDK1。0。X 安全模型

因此在 JDK1.1.X 中，就引入了“签名的 Applet”概念。这种签名的 Applet 一旦签名得到验证，就等同于本地应用程序，可以任意存取本地系统和网络上的资源。

在 Java 语言中是通过安全管理器(Security Manager)来保证上述安全模型实现。当某一操作需要存取系统资源时,安全管理器检查此代码,若允许存取,则继续执行;否则禁止存取并产生一个安全异常。对于上述的安全模型来讲,安全管理器的功能是非常简单的。

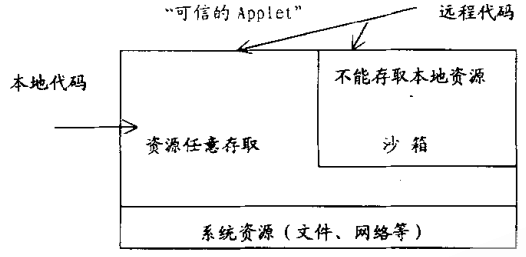

图 2　JDK1。1。X 安全模型

但上述模型都存在着一些缺陷,在某些情况下可能会产生安全漏洞。例如以下的例子:一个完成文件编辑功能的本地应用程序,用户可能会希望此程序可以读取本地的文件,但不能通过网络传递信息。这样就可以防止此应用程序窃取本地的文件通过网络将其传输到其他站点。但这种安全要求在上述模型中是无法实现的。因为一旦本地应用被认为是可信的,就可以存取任何系统资源。

因此就产生了新的安全需求:对应用的控制是可以配置的,即可以根据一些信息来对应用进行不同的控制。这样对于上述的例子就可以允许此编辑器的本地文件访问权而禁止其网络访问权。甚至可以进一步细化其文件访问权,使其被允许访问某个目录下的文件而禁止访问其他文件。而对另一个"绝对可信"(比方说自己开发的)的文件编辑器就可以开放所有的文件访问权和网络访问权。

根据这些安全需求,Java 语言又提出了新的安全模型,即 JDK1.2(又称为 Java 2)的安全模型。

2. Java 2 安全模型

Java 2 提出了一个更加完善的安全模型,这个模型中实现以下的安全策略:

(1) 对用户可以采用细致的鉴别与授权。由于现有系统对用户的身份鉴别方式复杂多样,从简单的口令鉴别到复杂的公钥鉴别甚至基于硬件的鉴别。Java 2 中对这些不同的鉴别方式都提供了一个统一的界面来实现这些不同的鉴别。

(2) 更加细致的存取控制。在 Java 2 中可以对系统的不同资源实施不同的控制,且控制部分可以通过一些方法(如继承和客户化 Security Manager 和 ClassLoader 类等)自行实现。这样就可以实现对资源的区别对待和处理。如允许访问文件而禁止访问网络。也可以允许访问某一部分文件而禁止访问另一部分文件。

(3) 安全策略的配置灵活。在 Java 2 中将策略定义为一组许可的集合。某段代码的执行就是在一组许可的集合下执行。许可的格式定义也便于扩展和顾客化。

Java 2 的安全结构中还包括其他一些安全措施,由于与本文无关,在此就不加以介绍了。

3. Java 2 安全策略配置的不足

Java2 在设计时希望能够通过编程和配置两种方式对安全策略进行配置。Java 2 中将策略定义为一组许可的集合,某段代码的执行就是在一组许可的集合下执行。如以下的三组许可就是关于文件读的许可、网络连接的许可和运行时退出的许可:

```
COM. MySoft File read path
COM. MySoft Net connect remoteIP:port
COM. MySoft Runtime allow system. exit
```

代码在开始执行时,代码以某种方式获取某一组许可的集合,安全管理器也获取了同样的一组许可集合。当代码要执行某一项涉及安全的操作时,安全管理器会根据当前许可的集合来允许或禁止此操作的执行。这样,安全管理器就是通过一组许可的集合来控制代码的执行,即根据一组安全策略(或称为安全配置信息)来保证代码的安全执行。

当系统中如上的安全配置信息的数量增加和/或需要进行配置的 Java 安全管理器数量增加时,Java 安全管理器就存在着安全配置信息管理复杂的问题。这些问题主要体现在以下方面:

(1) 安全配置信息数量增加时,增加这些配置的开销增加,尤其在目前只能通过程序或手工逐个设置的情况下更是如此。当网络中需要配置的 Java 安全管理器数量大到一定程度时,增加这些安全配置信息的速度甚至比不上因为需求变化而导致的安全配置信息变化速度。这样就导致了整个系统安全策略的无法正确实施。

(2) 系统管理员很难得知所有虚拟机安全管理器的安全配置信息,从而导致无法对全局安全配置信息的有效、安全管理。

(3) 当配置信息大量增加时,一般用户无法对其进行正确有效的判断,而安全配置信息的不合理组合可能会导致安全漏洞的产生。

二、安全管理器中的代理技术

通过分析我们发现，这种因配置信息增加导致系统管理复杂甚至实际不可用的情况，同计算机网络建设中，因为结点数量增加而导致网络管理复杂的情况是非常类似的。在网络管理中，已经形成了 SNMP、CMIP 等网络管理协议，其中 SNMP 协议一直得到发展与广泛应用。因此为了解决安全管理器的配置管理问题，我们借鉴网络管理中配置管理的思想，对不同 Java 安全管理器中的安全策略进行集中配置管理。

简单网络管理协议(SNMP)作为一种用于网络管理的协议，其提供的配置管理功能是比较完善的。我们将其引入 Java 安全管理器的安全配置信息管理，即在 Java 安全管理器中增加实现 SNMP 协议的部分，使得通过 SNMP 协议可以对 Java 安全管理器中的安全策略进行管理。一种可行的方案如下：

在每个需要进行管理的安全管理器中增加一个 SNMP 的代理(Agent)和一个 SNMP 管理信息库(MIB)，在管理员系统管理平台上增加一个安全配置信息管理模块(Security Policy Manage Module)。其中：

(1) 每个安全管理器的管理信息库中存放相应 Java 安全管理器的安全配置信息；

(2) 每个安全管理器的代理负责检索、更新相应 Java 安全管理器的安全配置信息。代理同管理模块进行通信，执行管理模块的命令(收集配置信息、更新配置信息等)；

(3) 安全配置信息管理模块负责收集、管理、更新每个 Java 安全管理器的安全配置信息，由系统管理员或安全员直接进行操作。

在对安全管理器增加了这样的代理之后，其基本结构图如下：

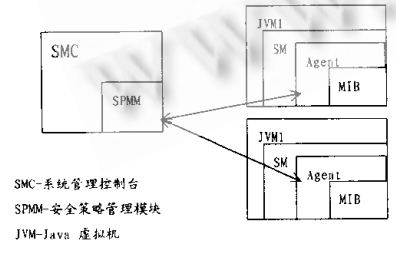

图 3 基于代理的安全配置信息管理

在这种结构下，为了保证安全配置信息的安全性，可以采用以下方法解决：

(1) 将管理信息库中的安全配置信息进行加密保存；

(2) SPMM 与代理之间互相认证，这可以采用公钥或私钥的方法实现；

(3) SNMP(v2 和 v3)支持不同的分级安全机制，这些机制可以按照安全需求和应用进行配置。这里为保证安全，需要选择“完全保密”类别。同时对其使用的密码算法还可以进行替换以增强算法上的安全性。

采用这种基于 SNMP 代理的安全管理器结构，可以充分解决因配置信息数量增加导致的问题：

(1) 系统管理员通过 SPMM 可以向代理发出查询命令，可以迅速得知某个或全部安全管理器中的安全配置信息，从而可以检查安全管理器中的安全策略是否符合实际需要和是否存在安全漏洞；

(2) 当系统的安全策略需要改变时，系统管理员同样可以通过 SPMM 向代理发出更新的命令，可以一次修改某个、部分甚至全部 JVM 的安全策略。这样既保证系统安全策略的更新及时，又可以保证由系统管理员来判断安全策略的完备性，避免因策略定义不当而造成的安全漏洞。

三、结论

Java 2 中提供了更细致的存取控制、可配置的安全策略和良好扩展性的存取控制结构。其中安全策略的可配置一方面增加了 Java 语言的灵活性，增加了安全控制的灵活度，另一方面也增加了 Java 虚拟机中安全管理器的配置复杂度。本文通过提出一种基于 SNMP 代理技术的方法，对多安全管理器的安全策略配置进行有效方便的管理，保证了安全策略的有效实施。

参考文献

[1] J. Gosling, B. Joy and G. Steele, The Java Language Speci - fication, Addison - Wesley, Menlo Park, Calif., 1996.

[2] T. Lindholm and F. Yellin, The Java Virtual Machine Speci - fication, Addison - Wesley, Menlo Park, Calif., 1997.

[3] L. Gong, “Java Security: Present and Near Future”, IEEE Micro, Vol. 17, No. 3, May/June1997.

(来稿时间：1999 年 5 月)