移动 Agent 系统审计的设计与实现

谭 湘 顾毓清

(中国科学院软件研究所 北京 100080)

摘 要 移动 Agent 系统是当前研究的热点之一,然而其安全性问题却影响了它的广泛应用。审计是移动 Agent 系统安全的一个重要组成部分,对于监督系统的正常运行有着重要的作用,然而目前的移动 Agent 系统很少涉及该问题。本文针对 Aglet 系统设计并实现了一个审计子系统。该子系统支持 3 种审计策略,采用 XML 语言作为审计日志格式,能够灵活方便地完成审计功能。

关键词 移动 Agent 系统,安全性,审计,审计策略

The Design and Realization of Auditing in Mobile Agent System

TAN Xiang GU Yu-Qing

(Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract Mobile agent system is one of the focuses of recent research, but its security issues prevent it from widely applying. Auditing is an important part of the security in mobile agent systems, it is important for monitor the normal run of system. But the audit issue is not involved in the current mobile agent systems. The design and realization of an audit subsystem for Aglet system are presented. This subsystem supports three audit policy, uses XML as the audit file format, performs the audit flexibly and conveniently.

Keywords Mobile agent system, Security audit, Audit policy

移动 Agent 是一个能在异构网络中自主地从一台主机迁移到另一台主机,并可以与其他 Agent 或资源交互的程序,实际上它是 Agent 技术与分布式计算技术的结合体,它具有移动性和自主性等特点[1]。随着移动 Agent 技术向电子商务、分布式计算、信息搜索等领域的深入,安全性问题显得日益重要。审计是移动 Agent 系统安全的一个重要组成部分,对于监督系统的正常运行,保障安全策略的正确实施有着重要的意义,然而目前的移动 Agent 系统很少涉及该问题。本文首先介绍了移动 Agent 系统的审计问题,接着介绍了一个基于 Java 的移动 Agent 系统 Aglet。然后针对 Aglet 系统设计并实现了一个审计子系统,该审计子系统支持 3 种审计策略,采用 XML 语言作为审计日志格式,能够灵活方便地完成审计功能,并通过审计文件分析器能够对审计日志进行有效的事后分析与追踪。

1 移动 Agent 系统的审计问题

审计是移动 Agent 系统安全的一个重要方面,审计日志使使用者和 Agent 为它们的行为负责任,从而是建立 Agent 和 Agent 平台之间的信任的基础[2]。审计主要涉及到的问题包括:

(1)审计事件的定义。在指定移动 Agent 平台上的每个过程、使用者或 Agent 的行为都必须是可以被审计的。为了可审计性,每个过程、使用者或 Agent 都必须被唯一地标识,认证和审计。例如,必须要审计的内容包括:如文件的访问,平台系统属性的变化。

(2)审计事件包括的内容。一般来说,至少应该包括以下内容:使用者/Agent 名字、事件时间、事件类型、事件成功或失败。

(3)审计日志的保护问题。可审计性要求保存一个审计日志,它记录着发生的安全相关的事件和为此事件负责的 Agent 或过程。该审计日志必须予以保护,防止未授权的访问和修改。

(4)审计子系统的异常处理。当存储介质达到它的容量时,需要采取适当的措施防止审计日志的丢失。这些措施包括:当审计日志达到一定容量时能够警告系统管理员;依靠系统管理员完成作为他们正常管理职责的日常维护。当审计子系统由于某些错误而不可用时,系统也能够有相应的措施,如警告等。

2 Aglet 系统简介

Aglet 是由 IBM 公司开发的基于 Java 的移动 Agent 系统[3]。它的设计非常简洁,将每一个移动 Agent 视为一个可移动的 Java 对象。Agent——在该系统中被称为 Aglet,在置于不同的网络主机上的 Agent 服务器之间迁移。系统提供一个上下文环境(context)来管理 Aglet 的基本行为:如创建、复制、分派、召回、暂停、唤醒、清除等。

在 Aglet 系统中,Agent 迁移是绝对的,因为它需要为目标服务器指定本地相关的 URL。移动性是使用 Java 对象的序列化来实现的,线程级的执行状态不能被捕获。Aglet 和 Aglet 之间的通信可用消息传递的方式来传递消息。Aglet 并非让外界直接存取其信息,而是通过代理 proxy 对象提供相应的接口与外界通信。Agent 被代理 proxy 对象所保护,它提供了语言级保护和位置透明性。

Aglet 安全模型定义了在认证过程中应予以区分的各种实体类别,包括 Aglet、context、Aglet 和 context 的制造者、所有者和管理者,其中 Aglet 是具有在其生命中唯一身份标识

的 Agent，context 代表 Aglet 的运行环境。

Aglet 将运行环境组织为域，所有的域成员必须遵守相同的由域管理员制定的安全策略。同一个域中的 Agent 系统相互信任。每个域中的 Agent 系统共享同一个密钥，该密钥由授权机构统一维护。域内的系统通过这个密钥，利用消息验证码(Message Authentication Code)来认证不同域中的 Agent 系统，如果认证通过，发起通信的用户 Agent 及其票据被一起发送，接收方通过此票据来判断它的权限。显然，如果密钥被泄漏，那么域内的所有 Agent 系统都会有被破坏的危险。

之所以选择 Aglet 作为研究的平台，是因为 Aglet 是一个开放源代码系统，可以较为方便地在它的基础上进行改进。

3 审计子系统的设计与实现

整个审计子系统的框架如图 1 所示。

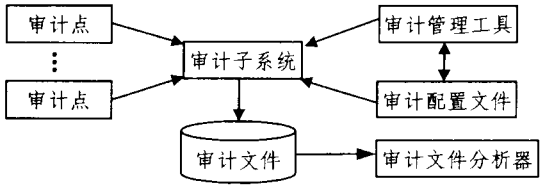

图 1 审计子系统结构图

3.1 审计策略

系统支持以下三种审计策略：

(1)事件生成策略：控制系统何时生成审计事件记录，以响应用户生成的或者系统生成的事件。事件生成策略的一般形式是：

if (动作与模式匹配)

then 系统必须生成新的审计事件

(2)事件处置策略：控制系统在生成审计事件后如何处理它们。事件处置策略的一般形式是：

if (审计事件匹配模式)

then 审计子系统必须对审计的事件进行记录或者发出警告

(3)连续性策略：如果审计子系统在需要时候不能运行，就由审计服务连续性策略控制系统的动作。审计服务连续性策略一般形式：

if (审计日志被填满或者审计服务不可用)

then 系统必须终止或者发出警告或者继续运行

3.2 审计事件

审计事件是系统审计的最基本单位。常见的审计事件有像 Create、Clone、Dispose、Dispatch、Retract 等基本事件，也有像一些与系统安全有关的操作，如系统属性操作、文件操作、socket 操作等。

为了配置管理的方便，系统根据审计事件自身的相关性，划分为若干审计事件类型，主要包括：

(1)Aglet 事件：Create、Clone、Dispose、Dispatch、Retract、Deactivate 等(如表 1 所示)

(2)文件操作事件：open、create、read、write 等

(3)系统属性操作事件：修改 java. home 属性等

(4)socket 操作事件：accept、connect、listen 等

在进行审计之前，必须要确立审计标准，即需要审计哪些事件。并不是所有的事件都要审计，可以只审计认为是重要的事件。审计管理员通过审计管理命令来配置审计配置文件，确定应该审计的事情。

表 1 Aglet 事件

Aglet 事件	名称	说明
Create	创建	新建一个 aglet 并赋予一个唯一的标识符
Clone	复制	复制一个 aglet 并生成新的唯一的标识符
Dispose	释放	aglet 终止执行并从当前环境中删除
Dispatch	分配	将 aglet 从一个环境分配到目标环境并在当前环境删除
Retract	收回	接受其它环境的请求，将 aglet 从当前环境“拉”到目标环境
Deactivate	暂停	暂时将 aglet 从当前环境删除并保存在辅存中
Activate	激活	aglet 从辅存中取出并恢复到原环境中
Messaging	发消息	aglet 发送消息给其它 aglet

当每次发生某个事件时，审计子系统会根据审计管理员所配置的审计策略来确定应该怎么反应。审计事件生成策略为系统审计提供了强大的控制，让其确定审计子系统何时生成审计事件。每个审计选项告诉审计子系统在发生某类系统事件时应该做些什么。每次发生系统事件时，审计子系统都将根据审计策略，看看是否有该类型事件的选项。如果没有该类型事件的选项，就不会产生审计事件。

3.3 审计文件

审计文件是存放审计结果的 XML 格式文件，每次审计开始，都按照设定的路径和命名规则产生一个新的日志文件，文件名的形式为：

YearMonthDate＃＃＃HourMinuteSecond

其中 year 表示年份，Month 表示月份，Date 表示日期，＃＃＃表示序号，编号从 001 到 999，HourMinuteSecond 表示文件创建时候的时、分、秒，以保证文件不重名。

审计系统产生的日志是 XML 格式的[4]。之所以采用 XML 格式是为了审计子系统的通用性和处理方便而定的。与 Aglet 服务器的运行情况相关的审计记录是包含有一系列事件的 XML 文档。事件的结构被设计来容纳 Aglet 系统中相关的一系列行为。每个事件包含源、行为、结果。事件 Event 元素的一个实例如下所示：

```
〈event timestamp="10:20:23 2003/8/30" server="aglets.ios.cas.ac.cn"〉
    〈source〉
        〈server name="aglets.ios.cas.ac.cn"〉
        〈/server〉
    〈/source〉
    〈action type="agletAction Create"〉
        〈target〉
            〈server name="aglets.ios.cas.ac.cn"〉
            〈/server〉
        〈/target〉
    〈/action〉
    〈result statue="success" type="aglet"〉
        〈aglet Id="20030811a3g4h6"
            time="10:20:24 2003/8/30"
            class="test.mytes.testAglet"
            codebase="atp://aglets.ios.cas.ac.cn:4500/"〉
        〈/aglet〉
    〈/result〉
〈/event〉
```

元素定义说明如下：

source 元素被用来定义 action 的发起者，发起者是 server 或者 aglet。source 的身份用 name 属性来定义。

action 元素被用来定义 source 所完成的操作。该元素包含有 type 属性定义了行为的种类。对 aglet 系统来说，type 属性包含关键字 agletaction，跟随着激活操作的名字(如

(下转第 157 页)

比较。从图2可以看出，随着最小支持度的减少，FDIA算法的运行时间在不断减少，Apriori算法的时间在增加。

而且FDIA算法的斜率要比Apriori算法的斜率小很多。因为支持度越少，候选项目集越多，Apriori算法扫描数据库的次数也越多。而FDIA算法不需要扫描数据库，支持度变化对它的影响不大。这正是FDIA算法的优势所在。

总结 本文提出的FDIA频繁项挖掘方法，利用生物免疫机制的特点，结合免疫算法，从新的角度挖掘频繁项集。与人工免疫的结合，为频繁模式挖掘开辟了新的途径，新的方法。我们认为这是一个有前途的研究思路。由于该算法是研究上第一次把人工免疫与频繁模式结合，只是一个基本算法，很多地方需要改进。如把群体更新时，如何产生新抗体，使新的抗体尽量挖掘到最大的频繁项集。以及在基础上如何根据数据库总容量调节群体的规模。如何把事务项目集的一些约束联系起来等都是今后非常有用的研究方向。

参考文献

1 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large database. In: Peter Buneman, Sushil Ajodia eds. Proc of ACM SIGMOD Conf on Management of Data, New York: ACM Press, 1993. 207～216
2 Han Jiawei, Kamber Micheline. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001
3 S de Castro L N, Von Zuben F J. Artificial Immune Systems: Part IBasic Theory and Applications[R]: [TR-DCA01/99]. 1999
4 Chun J S, Kim M K, et al. Shape Optimization of Electromagnetic Devices Using Immune Algorithm. IEEE Trans on Magnetics, 1997, 33(2): 1876～1897
5 王晓峰，王天然，等. 一种自顶向下挖掘长频繁项的有效方法. 计算机研究与发展，2004，41(1)：148～155
6 Lin Dao-I, Kedem Z M. Princer-Search: A New Algorithm for Discovering the Maximum Frequent Set. In: Bertram Ludascher, Wolfgang May, eds. Proc of the 6th European Conf. on Extending database technology, Proc. Lecture Note in Computer 1377. Berlin: Springer 1998, 1998. 105～119

(上接第148页)

create)。涉及Java安全许可，type属性包含许可类型的类的名字(如：Java. net.)跟随着相关的许可(如open打开)。如果该行为需要额外的信息定义操作的对象，则在action元素中包含Target子元素。

result元素被用来定义操作的结果。该元素包含有status属性暗示行为是否成功。属性type定义了result标记所附加的数据类型。如果没有结果数据，那么result元素为空，type属性被设置成none。

3.4 审计点

在每个系统调用处(前面所述的Aglet事件)设置审计点，Aglet的源代码开放性使得我们可以修改其代码来捕获安全相关的事件，记录完整的信息关于Aglet的身份、请求的操作，请求的参数、操作的目标、请求的结果。安全相关操作的捕获可以通过增强Aglet系统的原语来达到。而且，Aglet系统所实现的安全管理器也可以被扩展来记录成功和失败的操作的参数[5]。

此外，Aglet系统所实现的安全管理器被修改来记录有关系统级别的操作的详细信息。例如，文件操作(open、write等)、socket操作(accept等)，系统属性操作(如修改Java. home属性等)。

为了记录这些审计事件，有关Aglet的信息必须从相应的Aglet的线程中获取。因此我们给AgletThread类增加新的方法getAuditInfo，该方法返回Aglet的AgletInfo结构，包含有Aglet身份的各种信息。

3.5 审计文件分析器

审计日志的分析是审计的一项重要功能。由于审计文件格式是XML格式的，我们采用采用JDOM技术来分析审计文件。JDOM的处理方式有些类似于DOM，但它主要是用SAX实现的[6]。因此它既有DOM方式处理的方便性，同时又不必担心处理速度和内存的问题，这在日志文件比较大的时候就显得更为重要。

分析器可以自由选择要分析的日志文件，也可以根据日期、审计事件、审计级别等进行查询。能够根据日志文件生成各种需求的结果。

3.6 异常处理

当审计系统出现错误或者审计数据填满时候，必须根据审计配置文件中所设定的审计连续性策略来进行异常处理。

(1)*审计数据填满* 审计数据可能会增长得很快，必须要限制审计文件所占用的文件系统空间，审计管理员可以配置文件系统的预留空间(如8%)，在写磁盘时候进行检查，如果低于预留值，则视为异常情况，需要紧急处理。目前系统支持两种简单的处理：(1)关闭审计子系统。系统继续运行，但是发出警告提示：审计日志被填满。(2)关闭整个系统。

(2)*审计系统出现错误* 审计系统运行过程中出错的可能会有很多，如文件创建或者读写操作失败等，这些错误往往很难得到满意的恢复，从而导致审计子系统不可用。当出现这种异常情况的时候，需要紧急处理。目前系统支持两种简单的处理：(1)关闭审计子系统。系统继续运行，但是发出警告提示：系统审计子系统不可用。(2)关闭整个系统。

4 审计子系统的安全保护

审计是移动Agent系统的重要组成部分，它监督着系统各种安全特性的实施。如果审计系统自身的安全性不够，审计数据的可靠性无法得到保障，那么就无法提供准确的事后分析和追踪。在审计子系统的设计与实现中，我们考虑了Aglet系统的安全特性，保证了审计的自身安全。

(1)程序和代码的安全。审计数据的采集记录部分与Aglet平台相关，审计进程的执行不受外界影响，审计管理员配置程序和审计数据分析程序严格遵守强访问控制MAC和最小特权控制PAC，保证只有审计管理员才能有权利使用这些程序。

(2)审计配置文件和审计文件的安全。审计配置文件和审计文件都必须施加MAC控制，确保只有审计员才能查看和修改审计配置。此外，对审计文件进行加密，保证泄漏出去别人也无法看到。

结束语 本文针对Aglet系统设计并实现了一个审计子系统，审计采用XML格式的审计记录，能够灵活方便地完成审计功能，并通过审计文件分析器能够对审计日志进行有效的事后分析与追踪。这为以后的进一步统计分析和入侵检测等的开发打下了良好的基础。通过实验比较，与原Aglet系统相比效率下降少于5%。目前的移动Agent系统众多，如何将审计子系统扩展到其它的移动Agent系统是一项十分有意义的工作，也是我们下一步要进行的工作。

参考文献

1 Jansen W, Karygiannis T. NIST Special Piblications 800-19: Mobile Agent Security. National Institute of Standards and Technology: [Tech Rep: MD208999]. 1999
2 Jansen W. Countermeasures for mobile agent Security. Computer Communications, 2000, 23(10): 1667～1677
3 Karjoth G, Lange D B, Oshima M. A Security Model For Aglets. IEEE Internet Computing, Aug. 1997. 68～77
4 Wold Widre Web Consortium(W3C) Technical Reports and Publication. http://www.w3.org
5 Lange D, Oshima M. Programming and deploying java mobile agents with aglets. Massachusetts, USA: Addison-Wesley press, 1998
6 http://jdom.org/downloads/index.html