网络虚拟机器人——“软件人”安全策略与授权模型

王洪泊[1] 马忠贵[1,2] 曾广平[1] 涂序彦[1]
(北京科技大学信息工程学院 北京 100083)[1]
(中国科学院自动化研究所复杂系统与智能科学重点实验室 北京 100080)[2]

摘 要 在整个“软件人”安全体系结构中,授权是给特定的委托人授予特定资源访问权的一种访问控制机制,有必要对其内涵和表示形式进行深入探讨。扩展了大多数现有 Agent 平台所依赖的 Java 授权语义,总结出包括正向授权在内的5种授权机制,并且设计了适应“软件人”应用场景的新的权限类型。通过对比优选出适合“软件人”策略表示的方案,提出了一种具有高扩展性和灵活性的基于策略的“软件人”授权模型。

关键词 人工生命,分布式系统,“软件人”,安全策略,授权模型

Virtual Robot in Network——“SoftMan” Security and Authorization Design

WANG Hong-bo[1] MA Zhong-gui[1,2] ZENG Guang-ping[1] TU Xu-yan[1]
(School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China)[1]
(State Key Laboratory of Complex Systems and Intelligence Science, The Institute of Automation of the Chinese Academy of Sciences, Beijing 100080, China)[2]

Abstract Authorization model is a mechanism for those given consigners to visit and control their special resources in whole SoftMan security architecture, so it is necessary to go deep into its connotation and denotation. The paper extended the Java's authorization semantics which has been popularly used in many Agent platforms, and there are five kinds of authorization mechanisms including positive authorization. A new permission types and a policy presentation schema were designed which is exactly suitable for SoftMan scenarios. A kind of policy-based SoftMan authorization model with high scalability and flexibility was put forward.

Keywords Artificial life, Distributed system, SoftMan, Security strategy, Authorization model

1 引言

“软件人”是拟人的软件人工生命,是在特性、功能、行为、结构方面对人的模拟、延伸和扩展[1],是计算机网络时代的一项崭新而关键的技术[2],不仅可实现更灵活的分布式异步计算,而且有助于把网络计算推向智能化的新时代[3]。

授权[4]是把权限授予用户、程序或进程,是现代分布式系统的重要安全措施,它对软件实体在其宿主平台的各种行为(如文件系统访问、CPU 占用)进行控制,而且可以有效地防止恶意软件实体对系统资源的篡改和删除等破坏行为。基于 SoftMan 构建的系统必然也属于分布式系统的范畴,它具有 SoftMan 自主、学习和进化的行为特征,在当前开放式网络环境中具有天然的技术优势,可以完成诸如信息检索、网络管理等复杂任务,但同时也面临着严重的安全隐患,这对我们研究 SoftMan 的授权提出了新的挑战。

2 授权的语义扩展和形式化描述

现有大多数 Agent 平台(如 Aglet[5])在授权语义上仅支持正向授权,无法表达像“不允许访问某资源”、“除某资源外的资源都可访问”、“只允许满足限定性条件时访问资源”、“甲的权限可以委托给乙使用”等语义。因此,为了更好地支持 SoftMan 在开放网络环境中的应用,我们借鉴 Ponder 语言[6]的一些特性,从方向性和粒度上都对 Java 授权的语义进行扩展,并将其归纳为5种授权机制。

2.1 正向授权(Positive Authorization)

传统的授权机制仅支持正向授权[7]。这种授权典型地应用在 Java 策略文件中,给某个用户或代码显式地授予各种权限。当执行安全敏感的操作时,系统的访问控制器检查策略文件中所有授权项,以确定当前用户是否具有执行此操作的权限。如果发现一个相符的授权项,则授予用户权限执行此操作;反之,用户因无权执行此操作而中止。正向授权对于 SoftMan 来说是一种基本的授权方式。

2.2 负向授权(Negative Authorization)

正向授权只能隐含拒绝权限,没有定义相应的正向权限即表示无此权限,这样无法直接描述类似“不允许访问某资源”的语义,因此我们在系统中引入负向授权。正好与正向权

到稿日期:2008-03-20 本文受国家自然科学基金项目(60375038),中国科学院自动化研究所复杂系统与智能科学重点实验室开放课题(20060105),国家“十一五”科技支撑计划(2006BAJ04B07-2),北京市自然科学基金项目(4072018)资助。

王洪泊 讲师,博士,CCF 会员,主要研究方向为人工智能、人工生命,E-mail:foreverwhb@126.com;马忠贵 讲师,博士,主要研究方向为智能通信;曾广平 教授,博士生导师,主要研究方向为人工智能、计算机网络;涂序彦 教授,博士生导师,主要研究方向为人工智能、大系统控制、智能管理、人工生命。

限相反的是，负向授权显式地定义拒绝授予某权限。当系统访问控制器发现试图进行的操作与策略文件中的一条负向授权项相符，则拒绝执行此操作。

负向授权能够提高授权效率。考虑这样一种情况，只有正向权限并且用户执行的操作在策略文件中没有定义，根据默认情况，系统将遍历授权项，最后拒绝执行操作。相反，假如在策略文件中明确定义了用户正在执行的操作的负向授权，那么系统找到此授权项后迅速拒绝操作，不需要完全遍历策略文件。

负向授权还能够增强系统的安全。假如系统管理员确定某用户没有访问文件系统的权限，那么一开始他就可以在安全策略中定义相应的负向权限，这就能有效地防止后来无意中赋予了此用户正向权限这种情况。如果认为负向权限的优先级高于正向权限，当正向权限与负向权限冲突时，系统忽略正向权限，所以此用户仍然没有访问文件系统的能力。由于采用了负向权限，系统是安全的。

正向授权和负向授权能够满足 SoftMan 对于授权类型的要求，但是要想在更细的粒度上对 SoftMan 的行为进行描述和控制，仅有这两个授权机制是不够的。下面我们分别引入例外、约束和委托等授权机制。

2.3 例外(Exception)

例外是对授权更精细的控制，它允许你在一个许可集内指定不符合授权条件的例外情况。例如，需要定义一个对某个目录的读权限，但其中有些文件是没有读权限的。在没有引入例外的情形下，要达到这样的目标，必须把目录内的这些文件移走。有了例外，这一工作就变得很简单，只需要定义一个 exception 就可以。

在形式化描述的层次上，例外应该位于正向授权或负向授权之内，是它们的子元素。例外本身的形式化定义与正向授权基本相同，包括 PermissionType, Target, Action 等项。但需要注意的是，在具体操作时，需要保证例外的子元素必须在包含它的授权的相应子元素的定义范围之内。

2.4 约束(Constraint)

与例外相同的是，约束可以更精确地定义权限。例外使得我们可以通过精确地控制 Target 来限制 SoftMan 的行为，而约束则使我们基于外部因素来决定授权。在 SoftMan 系统中，我们把外部因素限定于对时间的控制，即约束可以基于当前的系统时间决定授权的有效性。有了约束的支持，我们就可以允许某个 SoftMan 仅在某个具体时间段内访问系统资源。

2.5 委托(Delegation)

系统管理员使用委托可以把用户的某些权限授权给其他用户，从而不需要经常更新系统的安全策略文件。这是一个危险的授权行为，如果没有授权给正确的用户，有可能把此权限传递给任何人。因此，管理员在设置委托时应该谨慎。

讨论完这 5 种授权机制，现在给出正向授权和负向授权的完整形式化定义：

Permissions = {Subject?, (PositiveAuthorization | NegtiveAuthorization) * ,}

PositiveAuthorization|NegativeAuthorization=

{PermissionType, Target, Action, Exception, Constraint, Delegation}

Exception = {PermissionType, Target, Action, Constraint}

Constraint = {DateTime}

DataTime = {{And|Or|Not}?, {DataTime|

{Comparator, Year|Month|Dayofweek|Hour|Minute|Second}}

* }

Comparator = {==|! =|<|>|<=|>=}

Delegation = {Deleg, TimeLimit, Constraint}

其中：Subject — 被授权的用户或者组；

Target — 授权的目标，如文件；

Action — 在目标上执行的动作，如创建、删除等；

PermissionType — 权限类型；

Deleg — True 或 False，表示是否允许委托；

TimeLimit —以秒计，代表此权限委托给其他用户后多长时间内有效；

* — 此项可以不存在，或者存在一次或多次；

? — 此项可以不存在，或者只存在一次；

= — 使用右边项定义左边项；

| — 或。

3 权限类型设计

权限是对系统资源访问的抽象。通常情况下，它包括两部分：目标和操作。权限也可以理解为在目标上允许进行的操作集合。权限具有双重含义，不仅用于指定一个策略所允许访问的资源，同时表明在系统进行安全敏感操作前需要什么样的权限。前者是定义安全策略时需要考虑的问题，后者涉及到在何处实施安全策略。

系统引入 SoftMan 后，如图 1 所示，资源的概念得到拓展，不仅包括文件、网络连接、运行时等资源项，还应该包括 SoftMan、SoftMan 服务器、消息这 3 类新资源。SoftMan 携带代码和数据运行在 SoftMan 服务器上，SoftMan 服务器为 SoftMan 提供运行环境和管理设施，SoftMan 之间通过消息传递的方式通信。

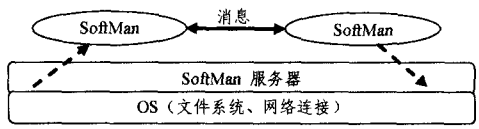

图 1 “软件人”系统

相应地，对这 3 类新资源的访问进行抽象，得到以下权限。

3.1 SoftManPermission

SoftManPermission 代表着访问一个 SoftMan 的权限。SoftManPermission 包括目标 SoftMan 的主体名称和对此 SoftMan 的操作名集合。主体名称即目标 SoftMan 用户的名称(如“yuchen”,“xiaolin”,“this”,或“anonymous”)，操作名称是控制 SoftMan 移动和生命周期的方法名，如表 1 所列。

表 1 SoftMan 移动和生命周期方法

方法	行为
Dispatch	发送 SoftMan 到目的地
Activate	从持久介质中唤醒 SoftMan 对象
deactivate	把 SoftMan 对象存储在持久介质中
Dispose	释放 SoftMan 对象
Clone	克隆 SoftMan 对象

3.2 SoftManServerPermission

SoftManServerPermission 指明一个 SoftMan 是否可以访

问服务器属性、启动或者关闭服务器、创建或者收回 SoftMan 等。SoftManServerPermission 包括两类：一类是访问服务器属性，此时目标为具有关键字的服务器属性，记为 property.〈key〉，操作为 read 或 write；另一类是 SoftMan 服务器管理方法，目标是此 SoftMan 的主体名称，操作是对 SoftMan 服务器的操作名集合。主体名称即 SoftMan 用户的名称，对应的操作名称可为以下方法："start"，"shutdown"，"retract"，"create"，如表 2 所列。

表 2　SoftMan 服务器管理方法

方法	行为
Start	启动 SoftMan 服务器
shutdown	关闭 SoftMan 服务器
Create	创建 SoftMan
Retract	从目的地收回 SoftMan

3.3　MessagePermission

MessagePermission 代表向一个 SoftMan 发送消息的权限，消息传递通常使用方法调用来实现，因此应该使用权限加以控制。MessagePermission 包括目标 SoftMan 的主体名称和消息名称。主体名称即目标 SoftMan 用户的名称。消息名称指已经定义的某类消息，如"sayHello"。我们采用消息映射的机制来实现消息名称和对象方法的关联。

4　基于 XML 的策略表示

策略是指针对具体安全问题给某些主体授权的资源访问权限集合[8]。以上分别给出了授权机制的形式化描述和针对 SoftMan 系统的新的权限类型设计，很自然地考虑到，到底应该使用何种方式实现策略的具体表示。基于 Java 的 Agent 移动平台大多数直接使用文本文件配置安全策略，本节提出了一种基于加密 XML 的策略表示形式，解决了文本文件无格式和保密性差的缺点，下面具体分析 3 种可能的解决方案。

4.1　文本文件

普通文本文件简单直接，能够清晰地描述安全策略，它只须我们针对授权机制的形式化描述定义语法。Java 的策略文件使用的就是这种方式。它的缺点是文本文件本身是无格式的，不适合描述像授权机制这样具有大量重复和嵌套结构的数据格式。另一个缺点是策略是明文形式，可以直接修改，这样很容易因人为因素造成系统的不正常运行，这不符合对策略保密的要求。当然，可以对文本文件加密。

4.2　关系数据库

关系数据库由二维表格组成，它在描述重复结构的数据方面具有天然优势，但不适合描述具有深层嵌套结构的数据。大多数关系数据库都提供了基于角色的访问控制，未经授权的用户无法读取表内容，这一点上数据库优于文件。在充当策略文件时，关系数据库最大的问题是需要安装相应的 RDBMS。对于相对轻量级的基于 SoftMan 的系统，这一要求无疑是苛刻的。再考虑到策略文件一般数据量偏小，采用数据库存储感觉有些大材小用。

4.3　XML 文件

XML 文件在数据存储、数据格式转换等方面应用广泛，现代大多数软件的配置文件均采用 XML 格式。XML 文件实际上也是文本文件，只不过它需要用户定义文件的语法，即 DTD(文档类型定义)，这给了用户很大的自由度，同时克服了文本文件不适合描述重复和嵌套结构数据格式的缺点。但是它仍然具有保密性差的问题。这一点我们可以采用加密的方式加以解决。

综上所述，最终选择 XML 文件来描述安全策略，并且把授权机制的形式化描述作为策略文件的 DTD 来考虑。

举例来说，负向授权的 XML 语法如下：

```
〈NegativeAuthorization〉
  〈PermissionType〉〈/PermissionType〉
  〈Target〉〈/Target〉
  〈Action〉〈/Action〉
  〈Constraint〉〈/Constraint〉
  〈Exception〉〈/Exception〉
〈/ NegativeAuthorization 〉
```

5　基于策略的 SoftMan 授权模型

根据策略定义和策略实施分离的思想，我们建立如图 2 所示的 SoftMan 授权模型。模型中主要的数据结构为保护区、操作栈和策略映像。保护区封装了与代码源相对应的类特征，如代码源(包括 SoftMan 的 URL 和创建者签名证书)、主体即 SoftMan 执行者、权限集合引用。来自同一代码源的 SoftMan 处于相同的保护区。操作栈是当前嵌套调用操作的存储结构，每个操作(OP)属于类，每个类又属于某保护区，所以每个操作实际上与权限集合相关联。策略映像是策略库的内存表示，由主体、正向权限集和负向权限集等组成。

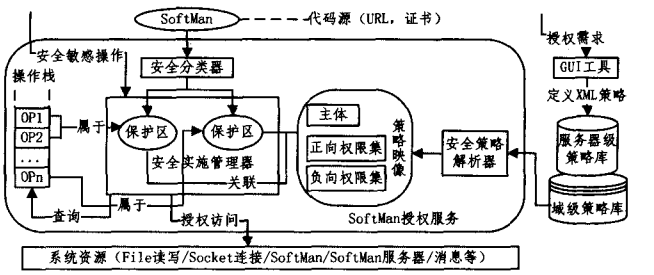

图 2　基于策略的 SoftMan 授权模型

模型的主要功能部件有安全分类器、安全策略解析器、安全实施管理器和 GUI 工具。当加载某个 SoftMan 时，安全分类器根据其代码源划分到不同的保护区，也就具有了不同的资源访问权。安全策略解析器读取 GUI 工具定义的 XML 格式的策略，转化为策略映像。当进行安全敏感操作时(如 SoftMan 迁移、读取 SoftMan 服务器属性)，启动安全实施管理器，安全实施管理器查询当前操作栈中所有操作对应的权限集是否都包括当前要进行操作的权限。若全都包括，则安全实施管理器授权访问相应的系统资源；若其中有一个不包括当前操作的权限，则拒绝资源访问。GUI 工具根据授权需求，定义符合 5 种授权机制形式化描述的域级策略库和服务器级策略库。此模型不仅具有良好的结构，而且具有很强的可用性和可扩展性。

5.1　策略的分级控制

策略实行两级控制方式，包括域级策略库和服务器级策略库。域是 SoftMan 服务器的管理组织，它定义域内全局的安全策略，实现基本的授权和访问控制。每个 SoftMan 服务器可以在域级安全策略的基础上进一步定义服务器级的安全

(下转第 286 页)

行。由于每次计算都是在上次更新数据的基础上迭代进行的，因此排行计算速度很快。

结束语 本文在"服务测试中介"模型基础上，提出一种基于贝叶斯投票算法的、多维度的服务评估技术，测试服务的各方在同一个协同测试的平台上共享和重用测试资产，系统根据协同测试数据实时地为同类服务进行动态排行，由于采用了带有用户信誉度和测试用例信誉度的加权平均的方法，设计思想具备用户交互和数据为核心的 Web2.0 的特点，服务排行的可信度会随着测试者、测试用例参与度和测试次数的增多而不断地提高。

参 考 文 献

[1] Bai X, Cao Z, Chen Y. Design of a Trustworthy Service Broker and Dependence-Based Progressive Group Testing. International Journal of Simulation and Process Modeling (IJSPM), 2006

[2] Tsai W T, Paul R, Cao Z, et al. Verification of Web Services Using an Enhanced UDDI Server// Proc. of IEEE WORDS. 2003: 131-138

[3] Tsai W T, Chen Y, Paul R, et al. Adaptive Testing, Oracle Generation, and Test Script Ranking for Web Services// 29th Annual International Computer Software and Applications Conference (COMPSAC). Edinburgh, Scotland, July 2005: 101-106

[4] Levis, Alexander H. Modeling and Measuring Effectiveness of C3 Systems. ADA173694, Sep. 1986

[5] 张燕生，白晓颖，蒋长征. 一种基于改进的贝叶斯投票算法的服务评估技术. 计算机科学，2008，35(4)

(上接第 277 页)

策略。这种逻辑分级结构有利于管理角色分工不同、各司其职，对属于不同级别的策略分别进行控制和管理。

5.2 策略定义和策略实施的分离

策略的实施是通过以安全实施管理器为中心，再辅以保护区、操作栈和策略映像 3 种数据结构的 SoftMan 授权服务完成的。我们可以通过外部 GUI 工具定义策略，动态更新策略，不必更改系统实现就可以实现对 SoftMan 行为的控制。这种分离的结构明显优于传统的策略定义和策略实施混合硬编码的解决方案，具有更强的可用性。

5.3 系统资源的高度抽象

系统内不仅定义具有普遍性的权限，如 File 读写、Socket 连接，而且抽象出特定于 SoftMan 系统的权限，SoftManPermission，SoftManServerPermission 和 MessagePermission。Soft-Man 扩展了资源的范围和类型，为系统的全面安全奠定了良好的基础。而且针对特定应用的需要，我们还可以扩展定义新的权限类型。

5.4 权限的不可传递性

在没有权限委托的默认情形下，权限是闭合的，即不可传递。如图 2 中的操作栈，操作 OP_n 调用 OP_{n-1}，OP_{n-1} 调用 $OP_{n-2}\cdots$，OP_2 调用 OP_1，按照调用次序依次进栈，OP_1 位于栈顶。设 $SoftMan_B$ 拥有比 $SoftMan_A$ 更高的权限，$SoftMan_A$ 的操作 OP_i 调用 $SoftMan_B$ 的操作 OP_{i-1}，那么 OP_i 仍然只被赋予了 $SoftMan_A$ 而非 $SoftMan_B$ 的权限。当安全实施管理器查询操作栈时自顶向下检查，遇到不具有权限的操作项，拒绝授予此权限。系统通过操作栈的这种数据结构可以有效地避免恶意代码的越权资源访问。

5.5 SoftMan 授权的分类管理

模型摒弃了直接将每个 SoftMan 和策略映像联系起来的方法，而是采用了更为灵活的设计，把具有相同特征(如代码源)的 SoftMan 分类到相应的保护区，每个保护区与一定的权限集合相关联，集中进行管理。

SoftMan 授权模型将策略定义和策略实施分离，这样我们就可以动态更新策略定义而不需要重构系统，增强系统的可扩展性和可用性。在策略的组织上，借鉴了网络中域的思想，对策略分层，把域的基本策略和特定于某 SoftMan 服务器的策略彻底分离，细化了策略的定义[9,10]。

结束语 SoftMan 自保护机制从系统架构入手，综合利用现有各种自保护技术，能够为各种应用提供合适的灵活的自保护安全服务，在一定程度上解决了如何选择移动代码保护方法的问题。

SoftMan 自保护机制从宏观上构建了一种安全框架并提出一些实现上的具体问题。如 SoftMan 提出安全申请前携带的安全需求规格说明的格式和描述语言、安全实施中心验证 SoftMan 申请合法性的依据和手段以及使用何种有效策略综合利用检测技术库、预防技术库和综合技术库中的保护技术等，都将成为下一步的研究重点。

参 考 文 献

[1] 曾广平，涂序彦. 软件人-网络世界中的虚拟人工生命[A]. 中国人工智能进展 2003[C]. 北京：北京邮电大学出版社，2003，12：677-682

[2] 曾广平，涂序彦. "软件人"的概念模型与构造特征[J]. 计算机科学，2005，32(5)：135-136，143

[3] 王洪泊，班晓娟，曾广平，等. 网络环境下虚拟机器人——"SoftMan"系统平台总体设计[J]. 计算机科学，2005，32(8)：152-154

[4] Bellavista G, et al. Security in Distributed Computing [M]. Prentice Hall, 1996: 21-27

[5] KarJoth G, Lange D B, Oshima M. A Security Model forAglets [J]. IEEE Internet Computing, 1997: 68-77

[6] Damianou N, et al. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, v2·0 [R]. Imperial College Research Report DoC 2000/1. 2000: 12-30

[7] Montanar R, et al. Flexible Security Policies for Mobile Agent Systems[J]. Microprocessors and Microsystems, 2001, 25(2): 93-99

[8] Wang Hongbo, Wang Zongijie, Zeng Guangping, et al. The Research on SoftMan Coordination and Its Application in Digital Gas Fields [R]// 2005 IEEE International Conference on Neural Networks and Brain. Beijing, China, Oct. 2005: 1429-1433

[9] 王洪泊，曾广平，涂序彦. 软件人系统研究及其应用[J]. 智能系统学报，2006，1(1)：24-28

[10] 曾广平，涂序彦，王洪泊. "软件人"研究及应用[M]. 科学出版社，2007