一个基于移动Agent的面向Web服务的信任管理引擎设计与实现

潘 静[1,2], 徐 锋[1,2], 王 远[1,2], 张 林[1,2], 吕 建[1,2]
(1. 南京大学计算机软件新技术国家重点实验室, 江苏南京 210093; 2. 南京大学计算机软件研究所, 江苏南京 210093)

摘 要: 目前大部分已实现的信任管理系统主要存在如下问题: (1) 实现没有遵循Web服务规范, 难以应用于基于Web服务的应用系统安全保障; (2) 其核心部分信任管理引擎的运行效率较低, 往往无法满足高度变化的应用环境对安全决策及时性的要求. 为此, 我们遵行Web服务规范, 基于移动Agent技术, 使用java语言实现了一个高效的, 面向Web服务的信任管理引擎以较好地满足开放、动态环境中基于Web服务的应用系统的安全保障需求. 信任管理引擎的主要工作是截获用户向web服务提出的访问请求, 派出移动Agent进行信任链的发现以判断该访问请求是否合法. 其中信任链的发现部分采用信任链双向搜索算法实现, 以提高整个引擎的运行效率.

关键词: Web服务; 信任管理; 信任引擎; 信任链; 凭证; 移动agent
中图分类号: TP302.1 文献标识码: A 文章编号: 0372-2112 (2006) 12A-2571-04

The Design and Implementation of a Trust Management Engine Based on Mobile Agent

PAN Jing[1,2], XU Feng[1,2], WANG Yuan[1,2], ZHANG Lin[1,2], LÜ Jian[1,2]
(1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China;*
2. *Institute of Computer Software, Nanjing University, Nanjing, Jiangsu 210093, China*)

Abstract: Most existing trust management systems have two main problems, (1) hard to provide security guarantee for web resources since their implements do not comply with Web Service standards; (2) the chief trust engine runs at a low efficiency, can not fulfill demands of on time security policy in the all long changeable environment. We use java language to design and implement a trust engine oriented to Web Service based on mobile agent to satisfy security demands in the open and dynamic web environment. The main work of trust engine is to verify whether a web resource can be visited by some client. We use mobile agent to do bidirectional trust chain discovery to improve the efficiency of the trust engine.

Key words: web service; trust manager; trust engine; trust chain; credential; mobile agent

1 引言

随着Internet的高速发展与广泛应用, 使用标准化规范协议, 具有良好封装性和高度可集成能力的Web服务架构, 为创建基于Internet的应用系统, 如企业ERP系统、在线电子商务系统[1]等带来了新的机遇, 同时也给应用系统的安全保障带来了新的挑战. 一些传统安全技术和手段无法完全解决此类系统的安全问题[2]. 1996年, M. Blaze等人为解决开放网络的安全问题而提出的信任管理思想[3], 为开放网络中的安全问题提供了新方法, 其基本思想是承认开放系统中安全信息的不完整性, 系统的安全决策需要依靠可信任第三方提供附加的安全信息. 基于此思想已实现了多个信任管理系统, 如SPKI/SDSI[4]、RT[5]、dRBAC[6]等, 提出了多个信任链发现算法. SPKI/SDSI中给出了一个凭证收集和推导相分离的集中式信任链发现算法, 其基本思想是以收集到的凭证集所表达的授权关系为初始集进行闭包运算, 以推导出新的授权关系. 但实际应用系统中所需处理的凭证往往分布于网络的多个节点, 凭证收集与推导相分离的处理方式不可避免的因无目的的凭证收集而耗费大量的时间和空间. 为此, RTO[7]和dRBAC中提出了更具实际意义的目标驱动信任链发现算法, 根据凭证所表达的授权内容, 以授权主体和客体为节点, 主体到客体的授权关系为有向边, 构造一个有向图. 然而, 此类系统实现时大都没有遵循Web服务规范, 因此难以应用于基于Web服务的应用系统中. 另一方面, 信任管理引擎是信任管理系统的核心内容, 主要工作是依据分散于开放网络环境中各实体的信任信息形成一条证据链, 进行开放环境中的信任信息收集和验证问题, 其优劣直接影响到整个系统的运行效率. 一般信任管理引擎实现时采用远程方法调用形式进行信任链的查找, 存在以下问题: (1) 多个实体进行并行信任链查找时, 若其中之一已完成查找任务, 其它实体无法得知, 仍会将链长延伸下去; (2) 每个迁移至远程节点进行查找的实体都要一直与查找发起者保持网络连接, 网络负载消耗较大; (3) 网络中每个节点都需要部署相同的复杂信任链查找方法, 如查找策略解析, 搜索结果记录和迁移计划更新等等, 增加了系统的复杂

收稿日期: 2006-10-15; 修回日期: 2006-12-29
基金项目: 国家重点基础研究发展规划(973计划)项目(No.2002CB312002); 国家高技术研究发展计划(863计划)课题(No.2006AA01Z159); 江苏省自然科学基金(No.BK2006712); 国家自然科学基金(No.60603034)

度.我们认为上述问题产生的主要原因在于引擎中将信任链的查找交给了每个固定在节点上的方法完成,缺乏独立的移动性.九十年代初出现并兴起的移动 Agent 技术[8],为此类问题的处理提供了一种适宜的解决方法.我们提出了一个基于移动 Agent 的信任管理引擎实现,引擎截获用户向 Web 服务提出的访问请求后,利用移动 Agent 实现信任链双向搜索算法,并依据发现的结果判断该访问请求是否合法并返回给相应的服务请求者.

文中第二部分描述了信任管理引擎的设计框架,其中包括凭证及其表达的授权形式以及双向信任链搜索算法,第三部分给出了一个信任引擎实现的简单例子,第四部分是与相关工作的比较,最后一部分对本文的工作进行总结和展望.

2 信任管理引擎设计框架

信任管理引擎的框架如图 1 所示.

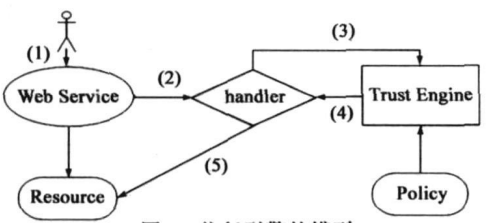

图 1 信任引擎的模型

图中(1)~(5)表明信任管理引擎的工作步骤:当用户调用某个 Web 服务资源时,配置于 Web Service 端的 handler 截取用户请求进行解析,将其转换为相应的待证明授权交给信任引擎,信任引擎通过查询本地策略库以及和其它信任引擎进行协作判断该用户请求是否满足安全设置,最后将是否允许访问的查询结果返回给 Web 资源所有者.本文的主要工作是设计一个信任链搜索算法进行凭证的查找,算法的优劣性决定了整个系统的工作效率,它与凭证的存储方式以及所表达的授权形式有关.

2.1 凭证及其表达的授权形式

凭证是对 Web 服务资源和访问者进行实体、角色和授权的定义.考虑到表达能力、简洁和可比性,本引擎中处理的凭证及其所表达的授权形式与 dRBAC 类似.dRBAC[6]是一种基于角色的信任管理系统,通过判断请求实体是否被授予访问某个资源所要求的角色来授权一次访问,其信任链发现的主要内容是回答"实体 P 是否具有与角色 R 关联的权限?".为简化问题,本文中未考虑 dRBAC 中提出的值属性概念,重要概念的具体含义如下:

• 实体(entity)用唯一的名字来标识,可以是资源的所有者 Web Service 或其访问者.

• 角色(role) 是 Web Service 定义并提供的一个或一类接口资源集合,形式为:服务名.角色名.若用户想访问一个受保护的资源,它必须能够充当该资源所对应角色的权限.

• 授权(delegation)授予某个实体具有某个给定角色的权限,其基本形式为:[Subject $\rightarrow$ Object] Issuer.其中 subject 主体代表资源的访问者,可以是实体或者角色;object 客体是资源的所有者,必须是一个角色;issuer 签发者必定是一个 Web Service 实体;箭头("$\rightarrow$")理解为一个 Web Service 授予服务调用者访问某一或全部资源的权限.

• 信任链(授权链, delegation chain)依据授权关系的传递性而存在的一个从某个主体到某个客体的授权序列.当 Web 服务请求者 req 想要调用提供者 pro 的接口 i 时,是否被允许就在于验证信任链 req $\rightarrow \cdots \rightarrow$ pro.r 是否成立,其中接口 $i \in$ RSpro.r.

• 证明(proof) 论证一个 Web 服务请求者是否具有访问某一资源的权限.

• 支撑证明(support proof)用于证明签发者具有授于某个非其所定义角色的权限.

• 授权形式(delegation form)根据授权主体、客体以及签发者之间的不同关系,授权分成三种基本的形式:

(1)自授权,签发者某个 Web Service 授权给访问者调用自身定义的接口资源的权利,例如:$[A \rightarrow B.r]B$、$[B.r_1 \rightarrow B.r_2]B$、$[A.r \rightarrow B.r]B$ 等.

(2)委托授权,签发者某个 Web Service 将其自身定义的客体角色委托给另一 Web Service 管理,后者有权将签发者定义的客体角色授权给其他实体.在授权表达中,采用客体角色上加一个撇号(')标记表示委托角色.例如:$[A \rightarrow B.r']B$、$[B.r_1 \rightarrow B.r_2']B$、$[A.r \rightarrow B.r']B$ 等.本文中不考虑 dRBAC 中定义的可传递的委托关系.

(3)第三方授权,签发者 Web Service 将其他 Web Service 定义的角色授权给另一实体,例如:$[A \rightarrow C.r]B$、$[A.r \rightarrow C.r]B$ 等.该授权的合法性需要由委托授权和自授权支持.

2.2 基于移动 agent 的信任链双向搜索算法

设待证明的授权为 $A \rightarrow B.r$,信任链搜索由一对安全验证 agent 协作完成.它们分别以已知主体和客体的匹配搜索作为自己的旅行任务,并行查找 Web 服务资源的期望角色集 E 和调用请求者的属性角色集 A.搜索过程中,认为所有凭证涉及到的实体处都有凭证存储,以保证在信任链发现过程中,能利用到所有存在的凭证.

以进行客体查询 E 的 agent 为例简述算法流程:agent 采用广度优先的搜索策略,优先依据本地存储的自授权凭证,尝试将待证明授权的客体角色(右侧)替换为与其客体角色相同的凭证所表达的授权主体(左侧).若替换为实体,则与待证明授权的主体进行比较,相同则证明成功;若替换为角色,则将该角色加入到结果集合 E 中,对 E 中每一个角色重复上述查找,对本地所有满足条件的凭证逐一进行替换,直到所有凭证均被使用过.这样通常能较大范围的获得满足条件的信任链,减少证明所需时间.agent 维护一个有限 task 组成的任务栈 Task Stack(TS),对于匹配过程中遇到的第三方授权情况,则将当前查找任务及其状态压栈,设支撑证明为当前任务,进行限定长度的深度优先搜索,将待证明授权的客体角色反复替换直至验证成功,并将该角色加入到 E 中;若超过限定深度则第三方验证失败.深度优先的搜索策略可以快速处理支撑证明.最后将之前的验证步出栈继续执行,直到验证成功或任务栈满使得验证失败.另一已知主体的验证 agent 同理.每完成一个节点的查找,一对 agent 将查找结果进行交运算,结果不

为空则验证成功, 算法结束, 否则继续查找, 大致流程如图 2 所示.

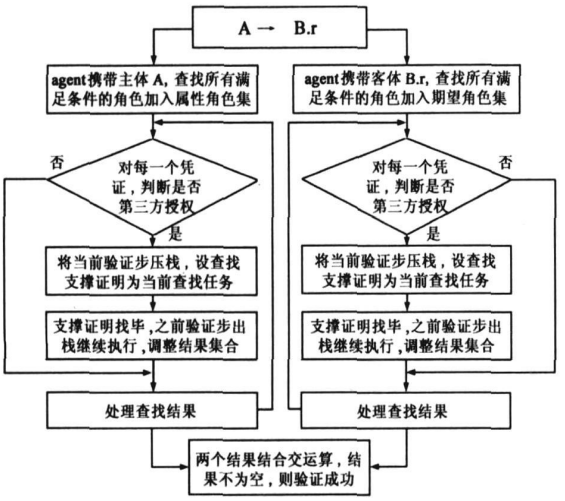

图 2　基于移动 agent的双向搜索算法流程图

对于算法的描述如下: 一个长度为 L 的任务栈 TS 中, 每个task 中包含一个待证明的授权 r 及其主体 subject、客体 object 和签发者 issuer, 期望角色集 objSet, 属性角色集 subSet, 迁移旅行计划 FIFO 队列 Itinerary 和最大旅行步 n, 以及所要执行的查找方法. 引入四个临时变量: D 为本地凭证集所表达的授权集合, p 表示本地授权集中的一个授权, resultSet 为结果集合, M 表示验证过程中遇到或推导出的委托授权集合.

ProveDelegation Algorithm(r)

(1)　objQuery(r) AND subQuery(r); //一对 agent 并行进行期望角色集和属性角色集的信任链搜索

(2)　resultSet ← objSet ∧ subSet;　　//将两者搜索结果取交集

(3)　If resultSet ≠ ∅ Then

(4)　　succeed

(5)　else

(6)　　fail

(7)　End If

其中, objQuery() 函数是从待证明授权的客体出发, 与本地授权集里的每一个授权进行比较, 比较过程由一个递归的函数 objProve() 实现, 算法如下. subQuery() 函数同理.

objQuery Algorithm (r)

(1)　If objSet= ∅ Then

(2)　　objSet ← r.object

(3)　End If

(4)　For all object ∈ objSet Do

(5)　　objProve(object)

(6)　End For

算法中提到的 objProve() 函数算法流程如图 3 所示. subProve() 函数同理. 处理支撑证明信任链搜索的函数如图所示为 supportProve().

为保证 agent 的可控性, 对于每一个支撑证明, 我们只派出一个单向的 agent 进行查找, 出现多个节点需要进行远程迁移时, 将派出克隆的子 agent 并行以减少支撑证明的查找时间.

3　信任引擎实现与运行实例

我们利用"基于 Web Service 的移动 agent 系统"在 Sun Application Server 基础平台以及 NetBeans 开发环境中, 用 java 语言实现上述信任管理引擎. 考虑到系统中所采用的通信机制, 我们把引擎中的信任链查找部分设计成由两个移动 agent 和一个静止 agent 共同完成. 实现的大致思路是: 应用程序接到一个查找任务, 便在 host 节点上注册一个守护 agent 负责管理, 一对外派的 agent 分别按照各自的旅行计划迁移至各节点进行查找. 在外的 agent 可以根据查找结果动态改变下一步的迁移计划和查找任务, 并向驻守在 host 的 agent 发送结果. 后者将一对移动 agent 的查找结果进行比对, 决定是要消亡还是让其继续查找任务.

用一个待证明的授权[$A \rightarrow F.r$] F 需要跨越多个分散管理域的简单例子, 来体现 agent 之间协作完成双向搜索的优势. 信任管理引擎根据已有的授权构造一个验证某个待证明授权存在的证明序列(包括其支撑证明) , 可表示为如图 4 所示:

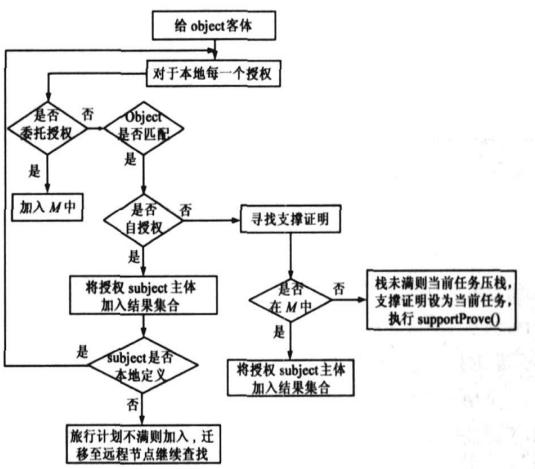

图 3　objProve()递归函数流程图

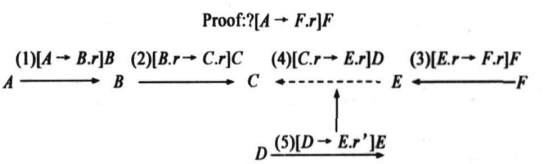

图 4　信任链发现的例子

图中(1)~(5)为五个已知授权, 双向查找分别从待证明的授权两端开始. 其中授权序列(1),(2)是总主体 A 开始查找的一条信任链, 证明实体 A 具有角色 $C.r$ 的权限, 授权序列(3),(4)从客体 $F.r$ 开始查找, 证明角色 $C.r$ 具有 $F.r$ 的权限. 授权(4)是一个由实体 D 签发的第三方授权, 其合法的前提是实体 D 具有管理属于实体 E 的角色 $E.r$ 的权限. 授权(5)即是委托授权的证明. 这样(1)~(4)便组成了[$A \rightarrow F.r$] F 的证明授权序列.

系统中, 外出完成信任链发现的一对移动 agent 为 subQuery 和 objQuery, 它们产生于同一个 host, 由其上的 hostAgent 负责跟据 agent 的名字来判断其执行的任务并对反馈结果进

行相应计算. 信任管理引擎的信任链发现通过调用系统提供的 Web Service 接口函数, 实现流程结构如图 5 所示:

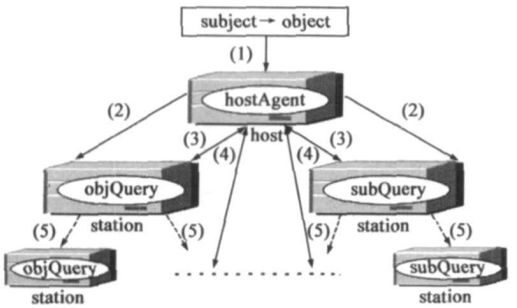

图 5 算法执行的流程结构图

(1) 应用程序调用接口函数 registerManager( ), 向设定的 host 注册 hostAgent.

(2) 一对 agent 分别调用 startMogent( ) 按照各自的初始旅行计划 Itinerary 在各节点间移动.

(3) 到达目的地后先调用 sendAddress( ) 汇报自己的所在站点的信息并等待反馈.

(4) 根据反馈, 若需要运行查找任务, 则执行后调用 sendMessage( ) 返回结果并等待指示.

(5) 需继续迁移的 agent 根据旅行计划前往相应站点, 重复上述(3), (4)步骤, 直到证明成功或任务栈 TS 满验证失败. 最后 hostAgent 将待证明授权成功与否的证明结果通过 getResults( ) 返回给待证明授权发起者.

4 实验结果与分析

与现存的其它信任管理引擎实现相比, 基于移动 Agent 的信任管理引擎实现的主要优势在于充分利用了移动 Agent 的移动性、自主性和协作性, 能够较好地应对凭证分布的特点, 提高引擎的处理效率.

早期的遵循 SPKI/SDSI 规范的信任管理引擎在实现时采用了凭证收集和推导相分离的集中式信任链发现算法. 这种方式很难处理凭证分布于网络中的多个节点的情况, 另外, 凭证收集过程的无目的性导致耗费大量的时间和空间. 而利用移动 Agent 在网络节点间的移动来完成凭证的收集与推导, 可以充分发挥移动 Agent 的移动性和自主性, 依靠已收集的凭证信息进行局部推导, 并依靠这些局部的结论调整凭证收集的方向, 既可有效减少凭证集的规模, 又可有效节约信任链发现的时间.

另一类以 dRBAC 和 RT 为代表的基于角色的信任管理系统, 在实现过程中采用了借助搜索标记的目标制导的双向信任链搜索算法, 一次搜索过程通常是各信任管理引擎之间的层层嵌套的远程方法调用过程. 其方法的主要问题在于各子搜索过程之间缺乏必要的交互, 导致重复搜索, 以及增加网络负载. 而利用移动 Agent 之间的通信机制, 可以较方便地实现多个 Agent 在一次信任链搜索过程中进行分工与合作. 如可以使用两个 Agent 同时从信任链的不同方向进行凭证收集, 并适时交换信息. 当某个移动 Agent 已经找到相应的信任链, 则可通知其他 Agent 结束. 移动 Agent 之间的协作性可避免出现重复的和无意义的搜索过程, 从而加快信任链发现的速度, 提高信任管理引擎的效率.

5 结论

本文给出了一个信任管理引擎的设计与实现, 其策略解析部分采用类似 dRBAC 的凭证及授权定义形式, 信任链发现部分采用基于移动 agent 的双向搜索算法. 结合移动 agent 的性能优势实现的该引擎结构较为简单, 具有管理方便、查找效率较高、灵活和通用的特点. 虽然本文设计与实现的信任管理引擎仅能处理类似 dRBAC 的凭证, 但其中的信任链发现算法具有一定的通用性, 经少量改造可用于处理 RT0 等包含其它授权形式的凭证.

参考文献:

[1] Xu Feng, Lü Jian, Tao Xianping, et al. A Mobile agent based electronic market space[J]. Journal of Nanjing University (Natural Sciences), 2002, 38(2): 131- 138.

[2] 徐锋, 吕建. Web 安全中的信任管理研究与进展[J]. 软件学报, 2002, 13(11): 2057- 2064.

[3] M Blaze, J Feigenbaum, J Lacy, et al. Decentralized trust management[A]. 17th Symposium on Security and Privacy[C]. Oakland: IEEE, 1996. 164- 173.

[4] D Clarke, J Elien, C Ellison, M Fredette, A Morcos, R L Rivest. Certificate chain discovery in SPKI/SDSI[J]. Journal of Computer Security, 2001. 9(4): 285- 322.

[5] Ninghui Li, John C Mitchell. RT: A role-based trust management framework[A]. 3th DARPA Information Survivability Conference and Exposition (DISCEX III)[C]. Los Alamitos: IEEE, 2003. 201- 212.

[6] Eric Freudenthal, Tracy Pesin, Lawrence Port, Edward Keenan, Vijay Karamcheti. dRBAC: Distributed role-based access control for dynamic coalition environments[A]. 22nd International Conference on Distributed Computing Systems (ICDCS' 02)[C]. Vienna: IEEE, 2002. 294- 306.

[7] Ninghui Li, William H. Winsborough, John C. Mitchell. Distributed credential chain discovery in trust management[J]. Journal of Computer Security, 2003. 11(1): 35- 86.

[8] 陶先平, 吕建, 张冠群, 李新, 董桓. 一种新的移动 agent 结构化迁移机制的设计和实现[J]. 软件学报, 2000, 11(7): 919- 922.

作者简介:

潘 静 女, 1983 年生, 硕士研究生, 主要研究方向: 系统安全.
E-mail: panjing@ics.nju.edu.cn