基于中间件的可定制信任管理框架

周明辉, 梅 宏, 焦文品
(北京大学信息科学技术学院软件研究所, 北京 100871)

摘 要: 针对 Internet 环境下广泛的信任危机, 人们着力于研究可信性支撑和度量技术. 传统模型和机制很少从信任本身看问题, 并且大多只涉及可信的单个方面, 对于有多种可信特性需求的系统来说, 很难提供一致的管理. 本文以 Internet 环境下的软件服务为研究实体, 从信任角度出发, 致力于为软件服务的可信性管理建立一致视图. 首先基于中间件建立了一个信任管理模型, 然后基于对可信特性的共性提炼, 构建了一个可定制、扩展性良好的信任管理框架, 并以可扩展为目标建立了一个可行的可信性度量模型. 最后基于构件运行支撑平台 PKUAS 进行了设计和实现, 并与相关工作进行了比较.

关键词: 可信计算; 信任支撑; 信任度量; 中间件
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2005) 05-0820-07

A Customizable Trust Management Framework Based on Middleware

ZHOU Ming-hui, MEI Hong, JIAO Wen-pin
(*Institute of Software, School of Electronics Engineering and Computer Science Peking University, Beijing* 100871, *China*)

Abstract: Being confronted with the trust crisis under the Internet environment, people have endeavored to develop technologies for supporting and measuring trust. Basically, traditional models and mechanisms seldom studied trust from the perspective of the trust itself and most of them can only deal with trust on a single aspect. Those traditional approaches cannot manage or support trust in consistent ways when systems are required to be in possession of multiple trust properties. Regarding software services in the Internet environment as the investigating entities, this paper is devoted to build uniform view for multiple trust properties. Firstly, a trust management model based on middleware is built. Secondly, based on the model, a trust management framework is built, which is customizable and extendable. Thirdly, by analyzing various trust properties deeply, a viable trust measuring model is setup. At last, the framework is implemented on the top of a middleware platform, i. e., the PKUAS. This paper also compares our research with the related work.

Key words: trust computing; trust support; trust metric; middleware

1 引言

随着信息技术的迅速发展, 信息技术基础设施从网络, 到操作系统, 到数据库, 到应用软件等, 在广泛的应用中暴露出越来越多的安全问题, 在天然具有开放性和动态性的 Internet 环境下彰显着信任危机. 同时, 因为 Internet 的巨大潜力, 社会各个领域的应用越来越多地基于 Intenet 实现和拓展, 人们越来越依赖于所身处的信息世界, 人们需要信任信息技术和服务. 这种矛盾的存在使得可信计算成为当前的一个研究焦点[1~4].

信任主要体现为一种关系, 即信任关系, 信任关系存在于信任实体(即信任者和被信任者) 之间, 信任者信任被信任者, 即"trustor trust trustee", 则被信任者被认为是可信的(trustworthy 或者 trusted). 为了方便描述, 本文将信任者实体记为 trustor, 被信任者实体(即可信实体) 记为 trustee. 如果 trustee 能够满足它所声明的或者 trustor 所期待的行为, 我们认为 trustee 可信. 因此, trustor 若要信任 trustee, 一方面 trustee 需要具有可信特性, 包括安全性*、可用性**、可靠性***、时效性**** 等(需要说明的是, 可信问题虽然主要是从安全而来, 但从信任角度看, 它理应具有更丰富的含义); 另一方面 trustor 需要对 trustee 所声称的可信特性进行度量, 从而决定是否信任 trustee, 进而

收稿日期: 2004-06-07; 修回日期: 2005-02-16
基金项目: 国家"973"重点基础研究发展规划项目(No. 2002CB312003); 国家自然科学基金(No. 60125206、60233010); "863"高技术研究发展计划项目(No. 2001AA113060)

* 系统的安全性主要体现为三方面: 一, 防止未授权用户访问数据(机密性); 二, 防止未授权用户操作或破坏数据(完整性); 三, 保证未授权用户不能使系统资源对于合法用户不可用(可用性).

** 系统的可用性常常用"一段时间内不能提供服务的概率"来衡量, 不能提供服务的时间越长, 可用性越差.

*** 系统的可靠性常常用"一段时间内出现故障(次数) 的概率"来衡量, 出现故障的次数越多, 可靠性越差.

**** 系统的时效性是指计算结果的正确性不仅依赖于计算的逻辑结果, 而且还依赖于产生结果的时间.

确定是否采用 trustee 的服务等. 为具有可信特性, trustee 需要保障机制进行支撑和维护, 有了这些机制后, 可以更方便地进行度量. 如何保障和度量 trustee 具有某种可信特性, 在某些研究领域已取得了卓有成效的研究成果, 例如为保障安全的授权机制[5,6](包括从信任的角度研究授权问题的信任管理系统[7,8]), 为保障可靠和可用的容错机制[9,10], 保障网上交易可信度的信誉管理系统[12]等, 但从信任角度分析这些工作存在一定的局限性, 主要有:

(1) 这些特性和机制基本上都是从各自的角度考虑问题, 难以覆盖信任的内涵. 例如, 维护可靠性的容错机制如何让 trustee 可信? 如何让 trustor 信任 trustee? 此类问题少有提及; 或者, 它们只是针对单一的可信特性, 将信任等同于某一种特性和机制, 例如授权或容错, 即使像文献[7]中的信任管理系统, 也只针对授权;

(2) 这些特性彼此独立, 对于有多种特性需求的系统来说(例如, 有些使用者只关注系统的安全特征, 而忽略时效性; 有些使用者只根据可靠程度或可用程度来判定可信度; 而有些使用者却可能既要求可靠性, 又要求时效性, 而且需求的程度不一样)很难有一致的执行效果, 也缺乏统一的管理框架;

(3) 度量方法和机制比较欠缺, 主要是安全度量比较活跃, 例如国际通用安全度量准则(CC)和《中国计算机信息系统安全保护等级划分准则》等定义了作为度量信息技术产品和系统安全性的基础准则, 但是, 这种度量一般要求可信第三方存在, 并且很难适应于运行时刻可信特性值的动态更改.

有鉴于此, 本文从信任的角度, 为支撑和度量 trustee 的可信性建立统一的管理框架和计算模型, 从而为有不同可信特性需求的用户建立一个统一视图. 本文余下内容组织如下, 第 2 节界定了本文的研究范围, 建立了一个基于中间件的信任管理模型; 第 3 节深入分析现有可信特性管理机制, 从信任角度进行共性抽象, 构建一个可定制的信任管理框架, 并给出了一个可行的可信性度量公式; 第 4 节描述了信任管理框架在构件运行支撑平台 PKUAS 上的实现; 第 5 节总结全文, 并与相关工作进行比较.

2　基于中间件的信任管理模型

Internet 环境下, 计算机系统是否可信, 包括硬件、网络、操作系统、中间件、应用软件、信息系统使用者以及它们之间的交互的复杂系统等是否可信, 在这条链上的任何一个环节出现问题, 都会导致计算机系统的不可信. 为研究方便, 本文所关注的可信实体是 Internet 环境下开放、动态的软件服务(或实体).

软件服务的可信特征贯穿于整个生命周期. 以它在实际环境下的部署为分界线, 本文将其可信分成静态和动态两个维度. 静态可信性是指软件实体在运行之前所具有的可信特征, 例如可以加强它们本身的体系结构和代码设计, 使得软件质量提高, 因此软件服务会更加可靠. 动态可信性是指软件实体在运行时刻所体现出来的可信性, 例如通过交互所积累的信誉经验值等, 或者通过容错所表现出来的可靠性与可用性, 也具有动态的特征. 本文主要考虑软件服务的动态可信性.

软件服务是否可信表现为它们提供服务时是否具有安全性、可靠性、可用性、时效性等特征. 本文的目的是从信任的角度对它们进行共性提炼, 从而能够实施统一的支撑和度量. 所谓支撑, 就是提供某些机制, 使得软件服务采用了这些机制, 可以在运行时刻具有某种程度的可信特征(对于有可信需求的应用来说, 如果没有支撑机制的存在, 这部分工作需要应用开发者承担, 而支撑机制将其从应用逻辑剥离, 可以极大地减轻开发者负担), 并且这些机制需要对服务进行动态地监控和管理, 以维护或调整它们的可信性; 所谓度量, 就是利用支撑阶段所建立的管理机制, 获取软件服务跟可信度相关的某些特性值, 根据某些计算公式或算法计算其可信度, 从而衡量软件服务是否可信.

首先来看如何为 Internet 环境下的软件服务建立一个信任管理(包括信任支撑和度量)模型, 进而以此为基础逐步深入, 建立可定制、可扩展的一致的管理框架.

借鉴已有的模型和机制, 我们的信任管理模型遵循下述原则:

- 信任管理逻辑和应用逻辑相分离. 使得信任管理逻辑的加入不会增加应用复杂度, 尽可能减少应用开发者在可信性管理方面的开发工作.
- 机制和策略相分离. 使得信任管理机制非常灵活, 能够适应不同应用的需求, 但同时又能保证管理机制不变, 只是随策略改变而有不同的动作.

在分布计算环境中, 中间件介于系统软件和应用软件之间, 为分布式应用屏蔽底层网络细节, 支持异构环境下应用的开发和互操作, 提供语言透明、位置透明以及可移植等特性, 极大地简化了分布式应用的开发, 因此, 越来越多的应用系统基于中间件技术开发. 鉴于中间件技术的广泛应用, 以及已有的较为成熟的中间件工作基础, 我们考虑基于中间件来构建可信管理系统. 目前中间件已经提供了良好的开发和运行框架, 如果以此为基础构建信任管理的统一框架, 不失为一个良好思路.

我们以中间件框架为基础, 构建一个信任管理框架(Trust Management Framework, 简称为 TMF), 用以管理软件实体的可信性. 如图 1 所示, TMF 基于中间件构建, 为在中间件之上开发的应用建立一个信任关系的管理环境.

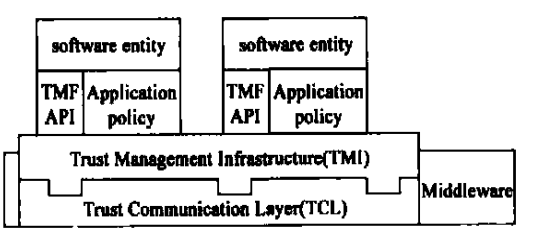

图 1 基于中间件的信任管理模型

TMF 分成两个部分: 信任管理基础设施(Trust Management Infrastructure, TMI)和信任管理通信层(Trust Communication Layer, TCL). TCL 主要为信任管理提供通信服务, 不仅提供一般的可靠通信, 而且在需要的时候还可以提供可信的(例如安全)通信能力等; TMI 包括对应用实体的可信性进行管理的各

种机制和工具,是 TMF 的主要部分.

TMF 对应用实体的可信性管理贯穿于它们的整个生命周期,包括设计、开发和运行.首先,TMF 提供工具辅助应用的设计和开发,为了能够满足运行时 TMI 的管理要求,应用本身需要有所约束,一般需要提供若干被管理接口(TMF API).接下来,应用通过策略(Application Policy)选择配置运行时所需的信任管理支撑,为了提供这种支撑机制,TMI 需要向外提供接口和策略,以灵活定制运行时管理框架.最后,当应用被部署到了管理环境中,则在运行时刻,TMI 和应用实体通过彼此提供的接口互相访问,实施管理,其中还可能需要 TCL 层的支持.

3 可定制的信任管理框架

基于上节的信任管理模型,下文将深入探讨 TMF 遵照怎样的原则进行构造.

3.1 信任管理框架

为了面向不同应用领域提供可信性支撑,能根据不同应用的可信性需求进行裁减的、基于构件的体系结构是最合适的选择.因此,TMF 主要采用构件化结构构造.TMF 将不同可信特性的支撑机制(例如保障安全性的访问控制机制、保障可靠性和可用性的容错机制,以及保障时效性的调度机制等)集成在一起,针对它们需要共同面对的问题提炼出一组统一的管理构件.

首先来看目前各种可信特性的支撑机制需要解决的主要问题:

第一,策略定制;面向应用开发者,支撑机制需要能够采集用户需求,从而定制应用行为.例如,为保障安全,用户可以通过输入像 JAVA2 中:

```
"grant Codebase" http://bar.com, Signedby "bar",
    Principal bar.Principal "duke" {
     permission java.io.FilePermission "/cdrom/duke/-",
"read";
    }"
```

的安全策略来定制安全需求,安全支撑机制会根据该策略来决定应用的安全行为.例如,为保障容错,用户可以通过配置应用服务的复制类型(ReplicationStyle)、副本数目(InitNumrReplica)、失效检测间隔(FaultMonitoringInterval)等容错策略来定制容错需求,容错支撑机制会根据这些配置来部署应用[10].

第二,策略部署;支撑机制根据用户定制的策略配置来部署应用.例如,安全支撑机制可以根据某些策略选择应用需要的认证模块(例如 $x.509$ 或者 PGP 或者用户名密码方式等);容错支撑机制根据用户定制的"副本数目(InitNumReplica)"策略确定为应用运行几个副本等.

第三,请求监控;在运行时刻,支撑机制需要动态监控应用服务面向客户请求的服务行为,并采用相应的管理动作.如安全,当有客户请求时,软件服务需要能够根据客户所持有的信任凭证对其进行认证和授权.而对容错,当有客户请求时,软件服务需要能够提供可用服务,不会因失效而导致请求失败.

第四,运行监控;在运行时刻,支撑机制需要动态监控应用行为,从而确定相应的管理动作.如安全,支撑机制需要时刻侦测是否有入侵,然后才能针对入侵采取相应措施.而对容错,支撑机制需要时刻检测失效行为,若有失效,需要采取恢复行为等.

第五,运行管理;在运行时刻,支撑机制监控到应用服务的某些行为之后,需要采取某些管理动作.例如,安全机制针对入侵采取的恢复,容错机制针对失效采取的回滚等.

针对这些共性问题,TMF 需要加以解决.在此之外,从信任角度对各种可信特性进行管理还会引入一些新的问题:

(1)可信特性调整;因为 TMF 针对的是多种可信特性,不同特性之间可能存在矛盾和冲突,例如对时效性的要求可能会影响到可靠性和可用性.这样,一方面用户需要定制对不同可信特性的需求程度(例如,在某些条件下为实时牺牲可靠),另一方面,运行时刻 TMF 需要收集各种可信特性的动态信息,在不同可信特性之间进行权衡和度量,根据用户预定义的策略进行动态适应性调整.

(2)信任度量;TMF 需要能够根据运行时收集到的信任相关的数据,基于可信性计算公式计算出可信度.

可以看到,为了满足可信特性的动态调整以及信任度量,TMF 需要为用户提供权衡策略进行定制,还需要收集动态运行信息为调整和度量提供依据.我们认为,这里的策略定制可以与上文的策略定制融合在一起,都是面向应用开发者;信息收集可以对应到上文中的运行监控;调整和度量虽然可以对应上文中的运行管理,但为扩展性需要,可以彼此独立.

因此,综合考虑TMF 需要解决的问题,遵循模块化、可扩展、灵活性原则,我们为 TMF 提炼出一组管理构件:策略定制、策略部署、请求监控、运行监控、运行管理、特性调整、信任度量等.以用户定制的策略为驱动,TMF 中各个管理构件会实施相应的管理行为.针对不同的可信特性,每个管理构件的行为模式是一致的,但执行动作会不同,因此我们采用派发模式来表述各管理构件的结构,如图 2 所示,管理构件 CA 的不同的执行动作体现为可定制的细粒度管理构件 Cai($1 \leqslant i \leqslant n$).

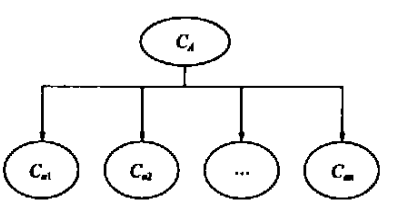

图 2 管理构件的结构

因此,TMF 中各管理构件可以针对不同的应用,通过应用选择的策略,运行时刻组装成应用定制的管理框架,从而提供相应的可信性管理.如图 3 所示:

- 信任实体通过中间件提供的基本通信机制或"信任管理通信服务"进行交互,信任者请求被信任者提供服务或响应信任者提出的访问请求.
- "策略定制"构件,主要面向应用开发者(或称应用),由开发者选择或定制应用想要被管理和维护的可信特性,并制定不同可信特性相关的可信度量化策略以及不同可信特性之间的权衡策略(参见 3.2 节).它接受应用定制之后,会根据策

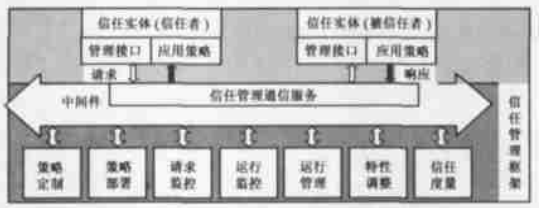

图 3 构件化的信任管理框架

略确定应用需要的部署行为,作为"策略部署"构件的输入.应用可以通过定制新的信任策略,即为新的可信特性定义特定的量化方式(公式),引入新的不同于上面曾提及过的各种可信特性,还可以将多种可信特性整合起来形成一个新的可信特性.

• "策略部署"构件,接受"策略定制"构件的输出作为输入,根据相应策略配置来部署应用.

• "请求监控"构件,运行时刻监控客户请求和应用服务的服务行为,执行管理动作.

• "运行监控"构件,运行时刻监控应用服务的一般行为,为维护应用服务的可信特性确定需要执行的相应动作;收集动态信息(例如跟时效性相关的执行时间,跟可用性相关的失效恢复时间等),为其它管理构件提供信息.

• "运行管理"构件,根据上者提供的信息执行相应动作.

• "特性调整"构件,根据"运行监控"构件提供的有关各种可信特性的动态信息,在不同可信特性之间进行权衡和度量,根据用户预定义的策略进行动态适应性调整.

• "信任度量"构件,根据"运行监控"构件提供的与可信度量相关的动态信息,根据度量公式动态计算可信度.

• "信任管理通信服务"为信任实体提供与信任管理相关的通信服务,例如支撑容错的组通信协议或支撑安全的 SSL 协议.

例如,应用开发者可以通过"策略定制"构件选择配置软件服务的安全性和可靠性等,则"策略部署"构件会负责初始化该服务并配置它所需要的各管理构件.各管理构件在运行时刻各司其职,进行监控管理,主要包括为维护可信性执行的管理动作以及动态的可信度计算等.同时,若应用开发者在"策略定制"中预定义了可信特性权衡策略,"特性调整"构件会根据动态监控到的软件服务的状态,做出适应性调整.当然,许多细节问题是实现相关的,本文只是从信任的角度给出管理模型和框架.

这样,TMF 将对各种可信特性和信任关系的支撑机制集成到了一个框架中,应用开发者可以通过策略定制可信特性的支撑,TMF 根据应用的定制策略动态配置,实施应用特定的可信性管理.TMF 的主要优点是:(1)将不同可信特性的管理提炼出共性,建立一组统一的管理构件,能够为应用提供可配置、可组装的灵活的信任管理;(2)可信特性调整构件的引入为应用提供了在不同可信特性之间权衡的机制,使得应用系统能够在外部环境变化的情况下根据预定义策略进行适应性调整;(3)因为 TMF 采用构件框架,提供良好的可扩展性,使得其它可信特性的维护能够方便地集成进来,而无需对原有机制进行太多修改.

3.2 可信性度量

从上节可以看到,TMF 为应用服务提供了灵活的可信性管理机制,但其中有两个关键问题需要解决:一是如何度量应用服务的可信性,二是如何制定可信策略.在 TMF 中,可信策略主要有三种,一种是与每种可信特性相关的策略,基本参照现有机制的做法,例如,安全中的访问控制策略;一种是可信度量化策略,这个与可信性度量相关;一种是可信特性权衡策略,这将是我们下一步工作的重点.本节主要讨论可信性度量.

每种信任特性的量化都是很困难的,更何况将它们综合在一起来考虑.本节试图给出一个可行的度量公式,但目的并不在于想提供完美的解决方案,而只是在可行的基础上留下可扩展空间.因此,下文在量化信任度时,首先将考虑如何为不同的可信特性建立一个可行的度量公式,然后再进行综合.

事实上,各种可信特性都可能因为覆盖面太广、标准太多而使得量化非常困难.例如安全性的计算模型各有说法.又例如时效性,虽然如定义所说,它依赖于产生结果的时间,但不同应用对时间的需求是不一样的,可能表现为下述情况的一种:

• "在 k 时刻(前)必须完成任务 a";
• "任务 a 必须在 n 个单位时间内完成";
• "在单位时间内,任务 a 必须(或至少)被执行 q 次";
• "任务 a 必须在任务 b 之前完成".

在现实应用中,用户可能只对某些可信特性的某些方面感兴趣,因此没有必要为每一类可信特性定义一个统一的量化公式.为此,在量化各种可信特性时,可以根据不同的应用系统及用户期望,定制特定的量化公式.例如,可以将安全性定义成软件服务被攻破的概率,在一段时间内服务被攻破的次数越少,安全性越高.设被攻破次数为 n,则安全性可以量化为

$$\text{Security}(S) = 1/n \tag{1}$$

又例如,如果用户更关心获得结果的快慢,那么可以将时效性定义成软件服务执行请求的及时性,执行时间越短,则认为它时效性越好.因此,设执行时间为 t,可以将其量化为

$$\text{Timeliness}(T) = 1/t \tag{2}$$

对于可靠性和可用性,文献[11]给出了一种较为经典的计算模型.可靠性指软件服务在一个完整的时间间隔之内正常服务的概率:

$$\text{Reliability}(R) = \text{MTBF}/(1+\text{MTBF}) \tag{3}$$

可用性指软件服务某个时刻可用的概率:

$$\text{Availability}(A) = \text{MTBF}/(\text{MTBF}+\text{MTTR}) \tag{4}$$

其中 MTBF 为软件服务的平均失效间隔,MTTR(Mean Time To Repair)为平均恢复时间.

另外,为了避免因为可信特性(例如安全性和时效性)所涉及的面太广而无法定义统一的量化公式,可以将复杂的可信特性分解成子可信特性,然后给不同的子可信特性定义相对独立的量化公式.譬如,对于时效性,如果用户既关心软件系统的响应时间,也关心系统返回结果的快慢,那么我们可以

分别定义两个不同的量化公式来量化这两种子可信特性. 通过引入和定义子可信特性, 可以简化可信特性的量化和管理. 在下文定义综合可信度时, 我们甚至可以将子可信特性看成是相互独立的, 从而为实现可组装的可信特性管理提供了可能性.

此外, 考虑到动态环境下可能的用户反馈信誉度量, 我们希望为软件服务建立一个动态信誉度. 这方面也有工作可以参考. 比较常见的场景是 P2P 电子社团中点(peer)经常要与不认识的或者不熟悉的点进行交互, 在没有可信第三方的情况下需要承担交互风险, 针对这种不确定问题, 有许多工作在深入研究. 文献[12]基于点与点交易的数目(包括交易总数和满意数目等)给出了一种度量点与点之间的信誉度的定量计算方法, 如公式(5)所示.

$$T(u,t)=\frac{\sum_{v\in P,v\neq u} S(u,v,t)*Cr(v,t)}{\sum_{v\in P,v\neq u} I(u,v,t)} \tag{5}$$

其中:

$T(u,t)$ Peer 网络 P 中, (点 u 之外的) 其它点关于交易 t 对点 u 的信任度量

$S(u,v,t)$ 点 u 与 v 之间关于交易 t 进行交互的满意数目

$Cr(v,t)$ 平衡因子, 为弥补点 v 可能的欺骗反馈

$I(u,v,t)$ 点 u 与 v 之间关于交易 t 的交互总数

交互信誉度, 参考公式(5), 本文为软件服务建立一个动态信誉度, 将它规约成一种可在静态信任权值的基础上, 根据以前交互经验的反馈信息(即信誉)动态地发生变化的一种量度:

$$Reputation(RT)=T_s+T_d \tag{6}$$

其中 T_s 可以是用户根据自己的其它经验值对软件服务所设置的初始信任权值, T_d 遵照公式(5), 只是其中的"交易 t"可以认为是用户或其它实体(总之是 trustor)对该软件服务的一次服务请求. $S(u,v,t)$ 是指 trustor 在请求被执行后反馈的对服务质量的满意次数, $I(u,v,t)$ 是指发生的服务请求总数.

以上文给出的各种可信特性的计算公式为基础, 我们可以定义一个用来计算相对于多种可信特性的综合信任度的综合信任函数, 即软件服务 U 在运行时刻 t、相对于多种可信特性的综合可信度可以计算如下:

$$STW(u,t)=\sum_{c\in TrustChars} w_c\times TW_c(u,t) \tag{7}$$

其中:

w_c 是特定可信特性(C)在综合可信度中的权重; $TW_c(u,t)$ 是相对于可信特性(C), U 的可信度. 计算同公式(6); TrustChars 是信任特征集, 实体所关心并维护的信任特征.

在信任管理框架中, 我们允许用户根据实际情况定制或调整不同可信特性的权重. 并且, 不同可信特性的权重应满足如下条件.

$\sum_{c\in TrustChars} w_c=1$, 即所有权重之和为 1.

不可否认, 上述计算公式都存在一定的局限性, 因为它们只是针对定义中的某一点进行量化, 没有考虑到各种可能的用户需求. 有鉴于此, 我们认为, 可以让用户自己提出度量公式(或者我们也可以进一步提出更完善的计算公式), 但需要遵照一定的原则.

这个原则就是可信度计算公式中的变量需要是 TMF 所能监控到的数据. 参见 3.1 节,"运行监控构件"会收集应用服务的动态信息, 为"信任度量构件"提供依据. 后者遵照公式进行计算, 前者需要提供公式中的变量信息. 显然, TMF 需要预先知晓计算公式中的变量值才能在运行时刻动态获取, 如果是预定义的计算公式, 则不存在问题(例如对于公式(7), TMF 需要获取的动态信息包括: 软件服务被攻破的次数, 执行时间, 失效间隔失间, 失效恢复时间, 被其它实体请求服务的次数, 服务反馈的满意次数等), 问题在于如果是用户自定义的度量公式, 并且如果这些公式的变量值不在预定义公式的变量范围内, TMF 如何能够为它们提取相应的变量信息. 这个问题的解决方案就在于需要为可信度计算定义出可能的相关变量值, 这个也是共性的凝练过程, 需要对每种可能影响可信特性的因素进行深入分析. 这个并非本文工作重点, 将在下一步工作中给出.

4 基于构件运行支撑平台 PKUAS 的信任管理框架的实现

针对上文给出的信任管理框架和可信性度量公式, 我们基于北京大学软件研究所自主开发的构件运行支撑平台 PKUAS[13] 设计和实现一个原型系统, 系统实现结构如图 4 所示.

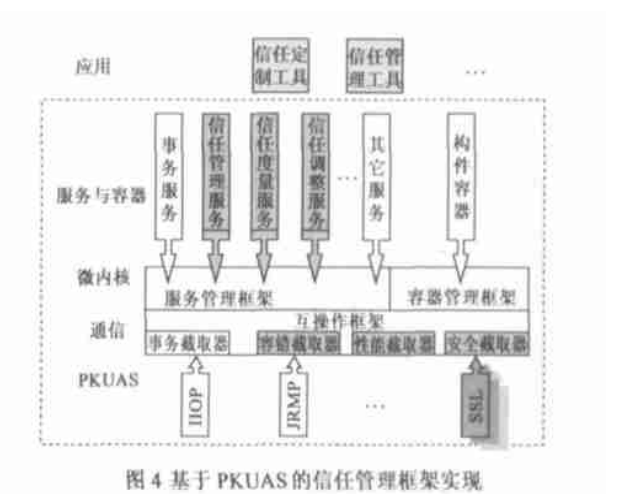

图 4 基于 PKUAS 的信任管理框架实现

PKUAS 遵循 J2EE 规范实现, 采用微内核结构, 服务(例如事务服务)、协议(例如 IIOP 协议)皆可插拔. 它的开放性为我们提供了一个良好的基础平台. 图中所示阴影部分是信任管理框架实现内容, 它们实现图 3 中各类管理构件的功能. 其中, 信任定制工具实现"策略定制"构件, 信任管理服务实现"动态管理"构件, 信任调整服务实现"特性调整"构件, 截取器实现"请求监控"构件等; 此外,"信任管理通信服务"实现为 PKUAS 中的不同协议等,"动态监控"构件功能需要由截取器和管理服务共同完成(因为信息采集需要请求时的服务信息和一般运行时的状态信息).

该管理框架为应用提供的管理步骤如下:

(1) 开发者完成应用逻辑开发后, 可以通过信任定制工具定制信任策略(例如根据需要选择可靠性和安全性), 信任定制工具会为应用生成一些关于信任管理的描述信息而发布;

(2) 应用部署时, PKUAS 能通过描述文件感知信任管理的需要, 从而为应用定制信任管理机制(该部分即为策略部署机制), 如为安全启动安全截取器, 启动相应的管理服务等;

(3) 运行时刻, PKUAS 中的信任管理机制进行监控管理: 截取器截取客户请求, 实施相应管理动作, 例如, 为安全调用相应的认证模块, 并根据安全策略进行授权; 信任管理服务监控应用的运行情况, 例如, 检测应用服务是否失效, 从而确定是否调用相应模块进行处理; 截取器和信任管理服务共同配合以获取动态运行信息, 例如, 通过截取器监控请求执行时间, 通过信任管理服务监控应用失效恢复时间, 从而获得相应的值来计算应用的可信度(参见 3.2 节), 并将值提供给信任管理工具;

(4) 使用者和管理员都可以通过查询信任管理工具而获得可信度参考值, 从而决定是否采用相应的应用服务或者是否调配某些系统资源.

可以看到, 信任管理框架能够有效地为应用提供信任管理. 首先, 应用开发者无需开发纷繁复杂的信任管理逻辑, 利用信任定制工具, 应用可以在部署时获得某种程度的可信性支撑, 例如, 为安全加入访问控制, 或者为容错加入日志恢复; 其次, 信任管理框架(包括信任管理服务和信任管理截取器等) 会在运行时刻对应用服务进行监控, 提取它的一些特征值以计算可信度, 作为使用者和管理员的度量依据; 此外, 管理机制还会根据环境变化做一些适应性调整, 例如若应用在初始定制时说明对时效性的要求大于可用性, 管理机制可以在系统负担过重时暂停对服务失效的监控.

5 结束语

可信计算中的关键技术问题是信任支撑和信任度量, 虽然学术界和产业界针对各种可信特性的研究历史都比较久远, 但很少从信任整体分析问题, 也基本上都是独立视图, 很难有一致的执行效果, 很难满足有多种可信特性需求的用户. 为此, 可信计算的重点和难点在于如何综合及权衡可信特性各个方面, 以达到系统的可信需求. 针对这些问题, 许多研究工作正致力于此. 例如文献[14] 和[15] 想要将经验值引入到授权机制中, 以动态确定和调整信任级别. 文献[12] 提到其未来工作想要将信任管理与入侵检测进行结合. 文献[16] 想要以构件的可信属性为基础, 预测构件组装后的体系结构的可信属性. 文献[17] 以中间件为基础, 通过组通信服务将高可用和容错等综合在一起.

本文以信任支撑和度量技术为研究焦点, 着重从信任角度为用户建立一个统一视图, 与已有的信任管理方法和机制相比, 我们的特色主要表现在:

- 将面向不同可信特性的管理机制进行共性提炼, 建立信任管理框架, 为应用提供可配置、可组装的灵活的信任管理;
- 考虑到不同可信特性之间的权衡问题, 引入权衡策略定制和信任特性动态调整机制, 使得应用在外部环境变化时能根据用户预定义策略进行适应性调整;
- 基于 J2EE 平台设计和实现一个支撑度量管理框架, 对可信应用进行统一的支撑和度量, 可扩展性良好(方便其它可信特性的加入).

下一步将主要考虑两个问题. 第一个问题是如何为应用开发者定制信任策略提供决策依据, 目前的做法是根据对可信特性的认识为 TMF 制定出信任策略, 然后提供工具让应用开发者进行选择. 这样做的问题在于开发者需要了解信任策略细节, 而不只是一个可信程度的选取. 如果能够将信任策略与可信度关联起来, 将进一步减轻开发者负担. 第二个问题是我们目前对信任策略的提炼还比较有限(包括对影响可信性的因素的提炼也很有限, 参见 3.2 节), 还不能较大程度地涵盖应用对可信的需求, 更无法映射为 TMF 实际的支撑行为, 因此, 如何找到有效的办法对信任策略进行描述和求精, 也是一个关键问题.

参考文献:

[1] NSF. CISE-Trusted Computing[Z]. www. nsf. gov, March 20, 2003.

[2] Microsoft. Trustworthy Computing[R]. Microsoft White Paper, www. microsoft. com, 2002.

[3] Bill Gates. Trustworthy Computing[R]. www. microsoft. com, Jan. 17, 2002.

[4] Trusted Computing Group. What Is TCG? [R]. http://www. trustedcomputinggroup. org/, April, 2003.

[5] B Lampson, M Abadi, M Burrows, E Wobber. Authentication in distributed systems: Theory and practice[J]. ACM Transactions on Computer Systems, 1992, 10(4): 265- 310.

[6] P Samarati, S C Vimercati. Access control: Policies, models, and mechranisms[A]. FOSAD 2000[C]. LNCS 2171, 2001. 137- 196.

[7] M Blaze, J Feigenbaum, J Lacy. Decentralized trust management[A]. Proceedings of 17th Symposium on Security and Privacy[C]. Oakland, IEEE Society Press. 1996, 164- 173.

[8] Tyrone W A Grandison. Trust Management for Internet Applications [D]. Department of Computing, Imperial College, London, 2003.

[9] Dhiraj K. Pradham. Fault-Tolerant Computer System Design[M]. Prentice Hall PTR, June 1996.

[10] Priya Narasimhan. Transparent Fault Tolerance for CORBA[D]. University of California, Santa Barbara, December 1999.

[11] Jeffrey D. Nyman. http://www. globaltester. com/sp5/stability. html [Z].

[12] L Xiong, L Liu. A reputation-based trust model for peer-to-peer ecommerce communities[A]. IEEE Conference on Electronic Commerce (CEC'03)[C]. Newport Beach, USA, June, 2003.

[13] 黄罡, 王千祥, 曹东刚, 梅宏. PKUAS: 一种面向领域的构件运行支撑平台[J]. 电子学报, 2002: 30(12A): 39- 43.

[14] 徐峰. 开放协同软件环境中信任管理研究[D]. 南京: 南京大学计算机软件研究所, 2003.

[15] T Grandison, M Sloman. A survey of trust in internet applications[J]. IEEE Communications Surveys and Tutorials, 2000, 4(4): 2- 16.

[16] H Schmidt. Trusted components-towards automated assembly with predictable properties[A]. Proceedings of the 4th ICSE Workshop on Component Based Software Eng[C]. Canada, 2001.

[17] S Mishra. A middleware for constructing highly available, fault tolerant, and attack tolerant services[A]. the Proceedings of the 17th ISCA International Conference on Computers and Their Applications (CATA 2002)[C]. San Francisco, CA, April 2002.

作者简介:

周明辉 女, 1974 年生, 博士后, 主要研究方向为分布计算, 软件工程和可信计算. E-mail: zhmh@sei.pku.edu.cn.

梅 宏 男, 1963 年生, 教授, 博士生导师, 主要研究方向为软件工程, 软件复用和软件构件技术, 分布对象技术等.

焦文品 男, 1969 年生, 博士, 副教授, 主要研究方向为软件工程和自主计算.