SOA中基于属性的访问控制安全策略

文俊浩[1] 曾 骏[1] 张志宏[2]
(重庆大学软件学院 重庆 400030)[1] (中国建筑标准设计研究院 北京 100048)[2]

摘 要 SOA环境具有分布性、异构性和动态性的特点,传统的访问控制模型已经不能满足其需求。为解决SOA环境下的访问控制问题,提出了一种基于属性的访问控制模型(Attribute-based Access Control,ABAC)。模型以实体的属性作为评价的基本单位。通过对主体属性、资源属性以及环境属性的动态评估,结合访问控制策略来对用户的访问进行控制。并采用XACML和SAML两个规范对模型进行了实现。分析了框架中属性和访问控制策略的查询响应方法,以及访问授权的流程。分析结果表明,结合XACML和SAML标准实现的ABAC模型具有较好的安全性和移植性,适用于异构的SOA环境。

关键词 面向服务体系结构,基于属性,访问控制,SAML,XACML
中图法分类号 TP309 **文献标识码** A

Security Policy of Attribute-based Access Control in SOA

WEN Jun-hao[1] ZENG Jun[1] ZHANG Zhi-hong[2]
(College of Software Engineering, Chongqing University, Chongqing 400030, China)[1]
(China Institute of Building Standard Design & Research, Beijing 100048, China)[2]

Abstract In order to improve the security of SOA-based system, it is essential to take advantage of access control in SOA. However, the traditional access control models are unable to be used in heterogeneous SOA environment. To coordinate access control with heterogeneous environment, an Attribute-based access control(ABAC) model was proposed, which, takes the entities' attributes as the basic units of evaluation. According to pre-defined strategy, the model can provide a dynamic access control by evaluating the attributes of subject, resource and environment. The model was implemented by XACML and SAML. Analysis shows that the access control model based on XACML and SAML standard provides more flexibility and portability, therefore it can be dedicated to the distributed environment using SOA.

Keywords SOA, Attribute-based, Access control, SAML, XACML

1 引言

随着信息化的发展,SOA被越来越多地运用于大型企业中,以解决异构系统之间的通信及交互问题,从而实现了企业全局的业务过程自动化,并大大降低了应用系统的集成和维护难度[1]。但是SOA在提供了强大的集成解决方案的同时,其安全性也成为关注的焦点。其中,访问控制是对关键资源进行保护,是评价系统安全性的重要指标。

传统的基于角色的访问控制模型(Role-based Access Control,RBAC)[2,3]通过对用户分配角色,再对角色赋予相应的权限来达到访问控制的目的。但是在开放的SOA环境中,基于角色的访问控制存在一定的缺陷:(1)角色是静态的,随着用户和访问资源的增多,需要定义更多角色。大量的角色将会增加维护的难度和成本;(2)在多个角色访问同一资源的情况下,不能提供细粒度的访问控制;(3)在跨域的访问控制中,一个安全域角色和权限在另一个安全域内有可能会失效。本文针对SOA环境的分布性、异构性、动态性的特点,提出了一种基于属性的访问控制模型(Attribute-based Access Control,ABAC),并结合XACML和SAML两个规范对ABAC模型进行了实现。

2 SOA环境中访问控制问题分析

2.1 面向服务体系结构

面向服务的体系结构(Service-Oriented Architecture,SOA)是以服务为基本粒度的新型的软件架构。SOA中的服务通过企业服务总线(Enterprise Service Bus,ESB)进行连接。ESB是一种新型中间件,提供了网络中最基本的连接中枢,是SOA的核心部分。各个服务之间通过简单对象访问协议(Simple Object Access Protocol,SOAP)进行服务调用时的消息交互。

2.2 SOA环境中的访问控制特点

SOA开放环境中具有分布性、异构性和动态性的显著特点,所以对访问控制也有了更高的要求。

(1)分布性:面向服务环境中,资源分布在不同的安全域

到稿日期:2009-10-16 返修日期:2009-12-25 本文受基金项目"十一五"国家科技支撑计划重点项目(2007BAF23B03)资助。
文俊浩(1969—),男,博士,教授,主要研究方向为面向服务计算,E-mail:jhwen@cqu.edu.cn;曾 骏(1984—),男,硕士生,主要研究方向为面向服务计算;张志宏(1960—),男,博士,研究员。

中，通过 ESB 进行连接，并没有统一的管理中心。而一个服务要能很轻易地与属于不同安全域的其他服务进行对话，就要求访问控制能适应这种分布性，使请求者能够在服务间自由流动，系统能够自动地将请求者身份随着边界的不同而转换。

(2)异构性：面向服务环境中，一套完整的业务流程可能由不同厂商的基于异构平台的多种服务组合而成。这要求访问控制具备平台无关性，能应用于每个异构的系统。

(3)动态性：服务的注册和发现以及服务间协作的建立都是动态完成的。要求访问控制能灵活地适应这种动态环境，并提供细粒度的访问控制。

3 基于属性的访问控制模型

3.1 ABAC 的基本思想

基于属性的访问控制模型，以访问控制的实体的属性作为最小粒度[4]，特别适合 SOA 的开放式环境，可为其提供细粒度的访问控制。属性作为访问控制中的基本单位，主要包括主体属性、资源属性、环境属性。

(1)主体是指请求对某种资源执行某些动作的请求者。主体属性的主要用户定义主体自身的身份和特性，包括：身份、角色、职位、年龄、IP 地址等。

(2)资源是指系统提供给请求者使用的数据、服务和系统组件。资源属性包括资源的身份、URL 地址、大小、类型等。

(3)环境是指访问发生时，可操作的、技术层面的环境或上下文。环境属性包括：当前时间、日期、网络的安全级别等。

3.2 ABAC 的定义

基于属性的访问控制模型的定义如下：

(1)设 S 表示主体，R 表示资源，E 表示环境。

(2)$SAk(k\in[1,K])$，$RA_m(m\in[1,M])$，$EA_n(n\in[1,N])$为主体 S、资源 R 和环境 E 的预定义属性。

(3)$ATTR(s)$，$ATTR(r)$，$ATTR(e)$ 分别表示主体 S、资源 R 和环境 E 的属性赋值关系。则有：

$ATTR(s)\subseteq SA_1\times SA_2\times\cdots\times SA_K$

$ATTR(r)\subseteq RA_1\times RA_2\times\cdots\times RA_M$

$ATTR(e)\subseteq EA_1\times EA_2\times\cdots\times EA_N$

例如对主体的单个属性进行赋值 $Age(s)=30$，$Role(s)=Employee$ 等。

(4)用一个基于 s,r,e 的布尔函数来判断主体 S 能否在环境 E 的条件下访问资源 R。

$Rule:can_access(s,r,e)\leftarrow f(ATTR(s),ATTR(r),ATTR(e))$函数根据 s,r,e 的属性值进行判断，如果返回 true 则允许访问；返回 false 则拒绝访问。

(5)一条策略可以由多条规则组成，每条规则都包含了主体 S、资源 R 及环境 E 的属性值。

3.3 ABAC 的基本结构

基于属性的访问控制模型的基本结构如图 1 所示。图中灰色部分为主要功能模块，其功能描述如下。

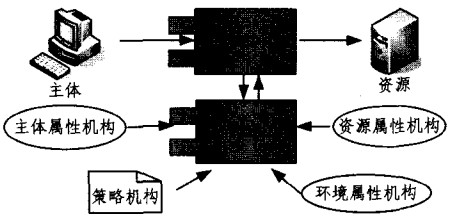

图 1 ABAC 的基本结构

(1)策略实施点(Policy Enforcement Point，PEP)是一个访问控制的实体，可拦截 SOAP 服务请求，并向 PDP 发送授权请求。然后根据授权响应结果，执行相应的动作，如允许用户请求和拒绝用户请求等。

(2)策略决策点(Policy Decision Point，PDP)用于收集主体、资源和环境的属性，并利用策略规则集来判断用户的访问请求是否满足要求，从而决定是许可还是拒绝，并将决策结果返回给 PEP。

(3)策略机构(Policy Authority，PA)用于编写策略和策略集，策略中主要定义了获得访问权所必须满足的属性要求。

(4)属性机构(Attribute Authority，AA)主要负责给 PDP 提供决策所需的各项属性，主要包括主体属性、资源属性和环境属性。

3.4 ABAC 和 RBAC 的比较

RBAC[9]是目前比较流行的一种访问控制模型，本节将通过一个示例，对 ABAC 和 RBAC 进行比较。

假设在一个系统中，用户 U 拥有年龄 Age(u)和类型 Tpye(u)两种属性，按年龄可以将用户分为成年用户 Adult 和儿童用户 Child；按类型可以分为高级用户 Premium 和一般用户 Regular。系统中的资源 R 也用两种属性，按付费与否 PayOrFree(r)可以分为付费资源 Paying 和免费资源 Free；按发布时间 ReleaseTime(r)又可以分为新资源 New 和旧资源 Old。假如有一条访问策略规定，只有成年的高级用户才能访问免费的新资源，则 ABAC 通过定义如下策略就可以进行访问控制：

Rule1：$(Age(u)=Adult)\wedge(Type(u)=Premium)$

Rule2：$(PayOrFree(r)=Free)\wedge(ReleaseTime(r)=New)$

Policy1：Rule1 ∧ Rule2

而在 RBAC 模型中，则需要根据用户的属性和属性取值定义 4 种角色，如表 1 所列。

表 1 RBAC 中的角色表

角色编号	角色
Role1	Adult premium user role
Role2	Adult regular user role
Role3	Child premium user role
Role4	Child regular user role

再根据资源的属性和属性取值定义 4 种权限，如表 2 所列。

表 2 RBAC 中的权限表

权限编号	权限
Permission1	Can access new free resource
Permission2	Can access old free resource
Permission3	Can access new paying resource
Permission4	Can access old paying resource

最后再对角色分配权限。如果存在 M 个主体属性和 N 个资源属性，Range()为某一个属性的所有取值的个数，那么需要定义的角色数 Num(R)和权限数 Num(P)为：

$$\mathrm{Num}(R)=\prod_{1}^{M}\mathrm{Range}(SA_m)$$

$$\mathrm{Num}(P)=\prod_{1}^{N}\mathrm{Range}(RA_n)$$

由此可见，当属性和属性的取值增加时，RBAC 模型需要定义的角色和权限的数量将呈几何倍数增长。而 ABAC 没

有角色这一中介，因此不需要定义大量的角色和权限。

另外 RBAC 没有考虑环境的因素，如果上例中的策略再增加一条规则：所有资源只能在 8：00 到 20：00 之间才能访问，则 ABAC 只需定义如下策略即可对访问时间进行限制。

Rule3：(*CurrentTime*(*e*)≥8:00) ∧ (*CurrentTime*(*e*)≤20:00)

Policy2：Rule1 ∧ Rule2 ∧ Rule3

4 基于属性的访问控制模型的实现

本文通过结合 XACML 和 SAML 这两项规范来实现 ABAC 的访问控制模型。可扩展的访问控制标记语言(eXtensible Access Control Markup Language，XACML)[5] 和安全性断言标记语言(Security Assertion Markup Language，SAML)[6] 都是由结构化信息标准促进组织(OASIS)定义的安全标准。它们都是完全基于 XML 的开放标准语言。其中 XACML 用于描述安全政策以及对网络服务、数字版权管理和企业安全应用信息进行访问的权限。而 SAML 用于交换有关主体的安全断言信息，为支持异构应用间的单点登录和身份管理提供了一套标准的方法。

4.1 ABAC 框架的实现

目前 XACML 和 SAML 都有丰富的开源工具，本文采用 sunxacml 和 OpenSAML 两个开源工具来实现。这两个开源工具都是由 Java 语言编写的一组类库，分别封装了 XACML 和 SAML 规范的相关方法和操作[7]。ABAC 安全策略的体系结构如图 2 所示。

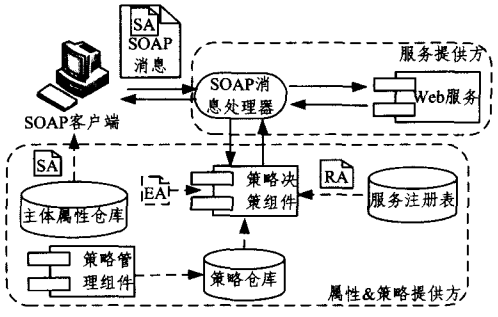

图 2 ABAC 安全策略的体系结构图

图中各个模块的功能如下：

(1)SOAP 消息处理器：充当策略实施点的作用，负责解析从客户端接收到的访问请求，抽取 SAML 断言中的主体属性，并将授权请求和主体属性发送给策略决策组件。在接收到策略决策组件的授权响应后，如果是访问许可则将访问请求发送给服务提供方；如果是访问拒绝则把拒绝信息返回给客户。

(2)策略决策组件：接收来自 SOAP 消息处理器的授权请求和主体属性，并从服务注册表中获取资源属性。然后从策略仓库中查找相关的策略。在获得策略和属性之后，策略决策组件用 SAML 授权决策断言来响应 SOAP 消息处理器。

(3)策略管理组件：用于产生和管理策略，并将产生的策略存储到策略仓库中。如果策略发生变更，也可以对策略仓库中的策略进行修改。

4.2 ABAC 访问控制流程

ABAC 的访问控制流程如图 3 所示。

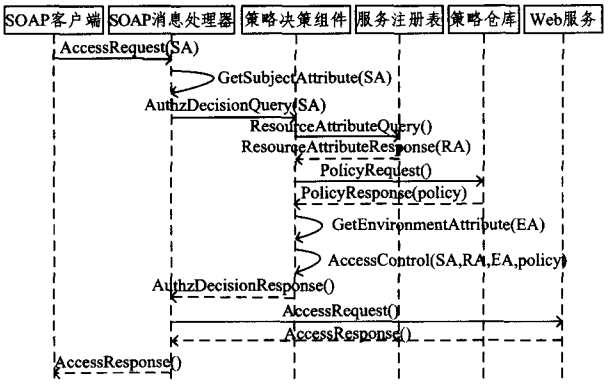

图 3 ABAC 访问控制时序图

需要补充说明的是，在客户端向 SOAP 处理器发送的 AccessRequset()中如果没有包含主体属性，则会重定向到身份认证界面。当用户输入正确的认证信息(如用户名和密码)后，再将包含主体属性的 SAML 断言封装到 SOAP 消息中。

如果策略决策组件返回的授权响应是拒绝访问，SOAP 消息处理器则不会向 Web 服务发送请求，而是将用户重定向到一个拒绝访问页面，告知用户该访问请求未被许可。

4.3 ABAC 访问控制策略的制定

ABAC 访问策略由 XACML 来定义，由于 XACML 完全基于 XML 标准，访问控制策略也完全由 XML 标签构成，其基本结构如下所示：

```
〈Policy PolicyId="…"〉
  〈Target〉
    〈Subjects〉
      〈Subject〉
        〈SubjectMatch MatchId="…"〉
      〈AttributeValue DataTyp="…"〉
        〈! -此处定义主体属性-〉
          〈/AttributeValue〉
      〈/SubjectMatch〉
       〈/Subject〉
    〈/Subjects〉
    〈Resources〉
      〈Resource〉
        〈ResourceMatch MatchId="…"〉
       〈AttributeValue DataTyp="…"〉
         〈! -此处定义资源体属性-〉
       〈/AttributeValue〉
        〈/ResourceMatch〉
      〈/Resource〉
    〈/Resources〉
    〈Action〉
      〈! -此处访问的行为-〉
    〈/Action〉
  〈/Target〉
  〈Rule RuleId="…"〉
    〈! -此处定义规则-〉
  〈/Rule〉
  〈Obligation〉
    〈! -此处定义职责-〉
  〈/Obligation〉
〈/Policy〉
```

访问策略主要由〈Policy〉元素来定义。〈Policy〉元素主要

由〈Target〉、〈Rule〉和〈Obligation〉元素构成，在策略决定点做出评估，以生成和访问决策[8]。由于可能发现多项策略对于访问决策来说都是可用的(并且由于单个策略可以包含多个 Rules)，因此可以使用组合算法(Combining Algorithms)来将多个结果协调成单个决策。标准组合算法被定义为 Deny-Overrides，Permit-Overrides，First Applicable 和 Only-One Applicable 等。〈Target〉元素用于将被请求的资源关联到一个可用的〈Policy〉上。它包含的请求〈Subject〉，〈Resource〉或〈Action〉必须符合〈Policy〉或〈Rule〉的条件，以便资源更适用。〈Target〉包括一个内置的方案，用于有效地索引/查找策略。〈Rules〉提供了在一个〈Policy〉中测试相关属性的条件。可以使用任何数量的〈Rule〉元素，每一个〈Rule〉元素都可以产生一个真或者假的结果。将这些结果进行综合便产生了对该〈Policy〉的单一决策，可以是 Permit，Deny，Indeterminate 或者 NotApplicable 决策。

结束语 由于 SOA 环境分布性、异构性、动态性的特点，基于角色的访问控制已经不能适应 SOA 环境下访问控制的需求。基于属性的访问控制以主体、资源、环境的属性为基本访问控制粒度，可以更加灵活地适应开放的 SOA 环境。结合 XACML 和 SAML 实现的 ABAC 框架，由于安全基于 XML 标准，更符合 SOA 异构环境的特点。但是大量的 XML 文档会增加网络资源的开销，从而增加访问的延迟时间。而且随着访问决策粒度的细化，SAML 断言中将会包含大量的属性信息，使这种延迟更加明显。所以如何控制访问决策粒度和找到最优化的访问控制策略，从而达到安全和性能的平衡，是未来需要研究的重点。

参 考 文 献

[1] 邢少敏，周伯生. SOA 研究进展[J]. 计算机科学，2008，35(9)：13-20

[2] Sandhu R S，Coyne E J，Feinstein H L，et al. Role-based access control models[J]. Computer，1996，29(2)，38-47

[3] 尹刚，王怀民，滕猛. 基于角色的访问控制[J]. 计算机科学，2002，29(3)：69-71

[4] Yuan E，Tong J. Attributed based access control (ABAC) for Web services[C]//IEEE International Conference on Web Services(ICWS'05). 2005

[5] OASIS. eXtensible Access Control Markup Language(XACML) v2. 0[S]. OASIS Standard. http://docs. oasis-open. org/xacml/2. 0/，2005-5

[6] OASIS. Security Assertion Markup Language (SAML) v2. 0 [S]. OASIS Standard. http://docs. oasis-open. org/security/saml/v2. 0/，2005-5

[7] Steel C，Nagappan R，Lai R. 安全模式[M]. 陈秋萍，罗邓，袁国忠，等译. 北京：机械工业出版社，2006：204-233

[8] 李晓峰，冯登国，徐震. 基于扩展 XACML 的策略管理[J]. 通信学报，2007(01)：103-110

[9] 罗海，安世全. 网格访问控制及对 RBAC 模型扩展的研究[J]. 重庆邮电大学学报：自然科学版，2008，20(6)：714-718

(上接第 116 页)

结束语 本文描述了分布式交换机内部通信系统的设计和实现，并重点论述了基于 FEC 的可靠组播算法的实现。提出了一种自适应的 FEC 动态算法并计算了 RDCP 协议的性能。通过试验，验证了 RDCP 协议的有效性和可扩展性。

参 考 文 献

[1] Yang L，et al. Forwarding and Control Element Separation (ForCES) Framework[S]. RFC3746. 2004

[2] Floyd S，et al. A Reliable Multicast Framework For Light-Weight Sessions and Application Level Framing[J]. IEEE/ACM Transactions on Networking，1997，5(6)：784-803

[3] Adamson B，et al. Negative-acknowledgment(NACK)-Oriented Reliable Multicast(NORM) Protocol[S]. RFC 3940. 2004

[4] Rizzo L. Effective erasure codes for reliable computer communication protocols[J]. Computer communication Review，1997，27 (2)：24-36

[5] McAuley A J. Reliable Broadband Communication Using a Burst Erasure Correcting Code[C]//Proc. ACM SIGCOMM 90. 1990：287-306

[6] Huitema C. The case for packet level FEC[C]//Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks. 1996：110-120

(上接第 130 页)

[3] 王映辉，张世琨，刘瑜. 基于可达矩阵的软件体系结构演化波及效应分析[J]. 软件学报，2004，15(8)：1107-1115

[4] Magee J，Kramer J. Dynamic structure in software architectures [C]// Proceedings of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering. 1996：3-14

[5] Luckham D. Rapide：A language and toolset for simulation of distributed systems by partial orderings of events[C]//Proceedings of the DIMACS Workshop on Partial Order Methods in Verification. 1997：329-357

[6] Allen R. A formal approach to software architectures[D]. Pittsburgh：Carnegie Mellon University，1997

[7] Kacem M H，Miladi M N，Jmaiel M，et al. Towards an UML profile for the description of software architecture[C]// The Conference on Component-Oriented Enterprise Applications. 2005：25-39

[8] 马晓星，曹春，余萍. 基于图文法的动态软件体系结构支撑环境[J]. 软件学报，2008，19(8)：1881-1892

[9] Metayer D L. Describing software architecture styles using graph grammars[J]. IEEE Trans. on Software Engineering，1998，24(7)：521-533

[10] Kong J，Zhang K，Dong J，et al. A generative style-driven framework for software architecture design[C]// Proc. of the 29th Annual IEEE/NASA Software Engineering Workshop. 2005：173-182

[11] Loyall J，Kaplan S，Goering S. Abstraction and composition in D-specifications of concurrent systems[C]// Proc. 6th Int. Workshop Software Specification and Design. 1991：52-59

[12] Bucchiarone A，Galeotti J P. Dynamic software architectures verification using DynAlloy[C]//Proceedings of GT-VMT. 2008