面向 XML 文档的细粒度强制访问控制模型*

李 斓+, 何永忠, 冯登国

(中国科学院 软件研究所 信息安全国家重点实验室,北京 100080)

A Fine-Grained Mandatory Access Control Model for XML Documents

LI Lan+, HE Yong-Zhong, FENG Deng-Guo

(State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62528254 ext 803, E-mail: lilan@is.iscas.ac.cn, http://www.is.iscas.ac.cn

Received 2003-09-17; Accepted 2003-11-11

Li L, He YZ, Feng DG. A fine-grained mandatory access control model for XML documents. *Journal of Software*, 2004,15(10):1528~1537.

http://www.jos.org.cn/1000-9825/15/1528.htm

Abstract: Information stored in XML documents should be protected by access control policy. Current access control models for XML documents are all based on DAC (discretionary access control) or RBAC (role-based access control). High security system uses MAC (mandatory access control) to secure information in system. XML document model is extended to include label information in this paper, and some rules that the extended model has to satisfy with are presented. Fine-grained MAC model for XML documents is described in detail by discussing four operations on XML documents. The fine-grained MAC model is based on XML schema, and its finest granularity of access control is element or attribute. The architecture and some mechanisms used to implement the fine-grained MAC model are discussed too.

Key words: access control; XML; mandatory access control; fine-grained; schema

摘 要: XML 文档存放的信息需要受到访问控制策略的保护.现有的一些面向 XML 文档的访问控制模型都是基于自主访问控制策略或基于角色的访问控制.高安全等级系统需要强制访问控制来保证系统内信息的安全.首先扩展了 XML 文档模型使其包含标签信息,并给出了扩展后的文档模型需要满足的规则.然后通过讨论 XML 文档上的 4 种操作,描述了面向 XML 文档的细粒度强制访问控制模型的详细内容.该模型基于 XML 模式技术,它的控制粒度可以达到文档中的元素或者属性.最后讨论了该模型的体系结构和一些实现机制.

关键词: 访问控制;XML;强制访问控制;细粒度;模式

中图法分类号: TP393 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60025205, 60273027 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802 (国家重点基础研究发展规划(973)); the National High-Tech Research and Development Plan of China under Grant No.2002AA141080 (国家高技术研究发展计划(863))

作者简介: **李斓**(1977－),男,江西新干人,博士生,主要研究领域为系统与网络安全;**何永忠**(1969－),男,博士生,主要研究领域为系统安全,密码学;**冯登国**(1965－),男,博士,研究员,博士生导师,主要研究领域为密码学,信息安全.

目前,XML[1](extensible markup language)技术的应用范围越来越广泛.随着 XML 文档中信息内容量的增长和敏感程度的加强,XML 安全也变得更加重要了.现在已经出现了很多 XML 安全方面的标准,例如,XML 签名,XACML 等.

为了保证 XML 文档中的信息不受到非授权的查看和修改,必须控制用户对 XML 文档的访问.XML 文档的结构是层次型的,它的基本成员是元素,通过元素的组织将许多相关的信息组合成一个文件.图 1(a)是一个记录公司雇员信息的 XML 文档实例——employee.xml.虽然一个 XML 文档包含的信息都是相互关联的,但是这些信息的敏感程度可能不相同,例如,employee.xml 中雇员的 salary 信息可能比较敏感,普通用户不能看到雇员的工资信息,但是能查看其他信息,而另一些高级用户却可以查看雇员的所有信息.这时我们必须为文档中的不同元素设置不同的安全属性,因此,我们需要细粒度的访问控制模型来控制用户对 XML 文档中信息的访问,这个模型不仅能控制用户访问整个 XML 文档,同时能精确到文档中的元素甚至是元素的属性.文献[2]讨论了一种对 XML 文档的细粒度访问控制系统,它实现了基于 DTD(document type definition)技术的自主访问控制策略.

DTD 和 XML 模式(XML schema)[3,4]是两种定义 XML 文档结构和内容的方法.DTD 出现的时间比较早,许多系统都在使用,但是与之相比,XML 模式有许多明显的优点.特别是,XML 模式文件本身也符合 XML 的语法,所以操作 XML 文档的机制同样可以用在模式文件上,于是可以采用统一的方式对 XML 模式和文档实例进行管理.因此,越来越多的系统都使用模式文件来定义 XML 文档的结构和内容.图 1(b)是 employee.xml 的 XML 模式文件 employee.xsd.

```
〈company〉
  〈empolyee name=“zhang”〉
    〈department〉manage〈/department〉
    〈office〉No.415〈/office〉
    〈phone〉52338215〈/phone〉
    〈salary〉10000〈/salary〉
  〈/empolyee〉
  〈empolyee name=“wang”〉
    〈department〉personnel〈/department〉
    〈office〉No.311〈/office〉
    〈phone〉52338327〈/phone〉
    〈salary〉7000〈/salary〉
  〈/empolyee〉
  〈empolyee name=“li”〉
    〈department〉sales〈/department〉
    〈office〉No.306〈/office〉
    〈phone〉52338364〈/phone〉
    〈salary〉8000〈/salary〉
  〈/empolyee〉
〈/company〉
```

(a) employee.xml

```
〈? xml version=“1.0” encoding=“UTF-8”?〉
〈Schema name=“EmployeeSchema”
xmlns=“urn:schemas-microsoft-com:xml-data”
xmlns:dt=urn:schemas-microsoft-com:datatypes”〉
  〈AttributeType name=“name” dt:type=“string”/〉
  〈ElementType name=“department” content=“textonly”
      dt:type=“string”/〉
  〈ElementType name=“office” content=“textonly” dt:type=“string”/〉
  〈ElementType name=“phone” content=“textonly” dt:type=“string”/〉
  〈ElementType name=“salary” content=“textonly” dt:type=“string”/〉
  〈/ElementType name=“empolyee” content=“eltOnly”〉
      〈attribute type=“name”〉
      〈element type=“department” minOccus=“1” maxOccus=“1”〉
      〈element type=“office” minOccus=“1” maxOccus=“1”〉
      〈element type=“phone” minOccus=“1” maxOccus=“1”〉
      〈element type=“salary” minOccus=“1” maxOccus=“1”〉
  〈/ElementType〉
  〈/ElementType name=“company” content=“eltOnly”〉
      〈element type=“employee” minOccus=“0” maxOccus=“*”〉
  〈/ElementType〉
```

(b) employee.xsd

Fig.1 XML document and schema files

图 1 XML 文档与模式文件

自主访问控制(discretionary access control,简称 DAC)策略允许用户自主控制其他用户对其拥有对象的访问权限,系统的访问控制工作分散给所有的用户,让这些用户共同完成.但是,DAC 本身存在无法避免的缺陷,例如,它不能防止特洛伊木马的威胁.许多高安全等级的系统,如安全操作系统、安全数据库等,都使用强制访问控制(mandatory access control,简称 MAC)策略来控制用户的操作权限.MAC 让系统根据安全信息来管理用户访问对象的权限,用户无法自由地把其拥有对象的访问权限授予其他用户.MAC 的目的在于保证信息的流动始终处于系统的控制之下.

用 XML 文档存放的信息有良好的结构和可扩展性.如果高安全等级系统使用 XML 文档存放数据,必然要采用 MAC 来控制用户对 XML 文档的访问.一个 XML 文档可以存放许多相似或关联的信息,但是这些信息不一定具有相同的安全属性.因此我们不能只把整个 XML 文档作为 MAC 的客体,而是要把访问控制的粒度细化到文档中的元素甚至是属性.因此,本文修改了 XML 文档模型,使其包含客体的安全信息.在此基础上,讨论了一种面向 XML 文档的细粒度强制访问控制模型,这个模型基于 XML 模式技术.同时给出了实现模型的体系结构

和部分机制,并尽可能地使用了 XML 技术.

第 1 节介绍国内外的相关工作以及本文与它们的比较.第 2 节介绍比较常用的强制访问控制模型——BLP 模型.第 3 节给出增加了标签的 XML 文档模型,同时讨论确定 XML 文档中元素和属性标签的算法.第 4 节详细讨论面向 XML 文档的细粒度强制访问控制模型,并根据操作的类型进行分析.第 5 节给出模型实现的体系结构以及部分机制.最后是结论.

1 国内外相关工作

国内外对 XML 访问控制的研究也有很多.XACML 是 OASIS 的一个规范,用于编写和实施基于 XML 的访问控制策略.但是这个规范没有说明如何去实现控制 XML 文档访问权限的模型.Dimiani 和 Bertino 等人讨论了基于 DTD 的面向 XML 文档的访问控制[6~8].文献[2]中提出了一种细粒度的访问控制模型,这个模型定义了元素上的访问权限.但是,这些模型都是基于自主访问控制策略的.基于角色的访问控制(RBAC)是一种灵活而方便的访问控制模型,文献[9]给出了用于实现 RBAC 模型的 DTD 文件.对于 XML 文档仓库的基于 RBAC 的访问控制方法在文献[10]中给出.文献[11]提出了一种使用 XML 在分布式环境中管理安全策略的方法,但也只是讨论了如何用 Java 和 XML 技术来实现 RBAC.文献[12]讨论了 XML 的访问控制技术,并提出了一种临时授权模型为 XML 提供了精细的控制机制.文献[13]提出了一种多粒度访问控制系统,它是 XACML 规范基础上的扩展,并简略地描述了它的体系结构.

以上的讨论结果要么基于 DTD 技术,要么没有给出确定的实现技术.而目前 XML 模式技术正逐渐得到广泛应用,文献[14]提出了基于模式技术的面向 XML 文档的 RBAC 实现方法.虽然 RBAC 可以用来模拟 DAC 和 MAC[15],但是目前并没有文章直接讨论面向 XML 文档的强制访问控制模型.本文试图给出这样的一个模型,它基于 XML 模式技术.同时,我们的模型也是一种细粒度的访问控制模型,受控制的客体不仅包括文档和元素,甚至可能是某个元素的属性.

2 强制访问控制模型

Bell-La Padula 模型[16]是高安全等级系统中最常用的强制访问控制模型.我们把系统中的对象称为客体(object),用户称为主体(subject),$\lambda(o)$,$\lambda(s)$分别代表客体和主体的标签,标签包含两个元素,密级 C(class)和范围 G(category),因此,它是一个二元组(C,G),其中密级 C 是可以比较大小的序列,而范围 G 则是一个位的集合,这些集合之间可以有包含关系,也可以互不包含.标签之间的关系有两种:支配关系和不可比关系.一个级别 $\lambda_1(C_1,G_1)$ 支配另一个级别 $\lambda_2(C_2,G_2)$ 当且仅当 $C_1 \geq C_2$ 并且 $G_1 \supseteq G_2$,记为 $\lambda_1 \geq \lambda_2$ 或者 $\lambda_2 \leq \lambda_1$.如果两个级别不存在支配关系,则它们不可比.模型的规则如下:

- (简单安全特性)主体 s 能够读客体 o 必须满足$\lambda(o) \leq \lambda(s)$.
- (*特性)主体 s 能够写客体 o 必须满足$\lambda(o) \geq \lambda(s)$.

简单安全特性保证用户“不能上读”,而*特性保证用户“不可下写”.这两个特性控制信息只能在同级之间或从低级向高级流动.因为 XML 文档中的信息都是结构良好的,如果低级用户要写高级信息,必须首先获取高级信息的结构.结构本身也是一种信息,为了保证低级用户无法获取任何高级信息,我们采用严格的 BLP 模型,用严格的*特性来代替原有的*特性.

- (严格的*特性)主体 s 能够写客体 o 必须满足$\lambda(o)=\lambda(s)$.

3 添加标签的 XML 文档模型

我们的强制访问控制模型面向 XML 文档,也就是说,XML 文档是模型的客体.但是,如果希望实现细粒度的访问控制,除了把整个 XML 文档作为客体以外,还要把 XML 文档的各级内部元素和元素的属性也作为模型的客体.强制访问控制模型中,每个客体都会有自己的标签.所以,不仅整个 XML 文档拥有标签,而且文档中任何子元素或者属性都可以拥有标签.一个 XML 模式文件可以用来定义一批结构相同、内容相近的 XML 文档,很多

情况下,这些文档的安全属性是相似的.为了方便管理,我们可以给 XML 模式文件及其元素定义标签,并将这些标签推广到符合该模式文件的所有 XML 文档实例中.因为 XML 模式文件本身也符合 XML 的语法,所以 XML 文档模型的定义既适用于普通的文档实例,也适用于 XML 模式文件.定义 1 给出了原始的 XML 文档模型的定义.

定义 1. XML 文档是一个七元组, $XDoc=(V_e,v_r,V_a,N_s,subelem,attrs,name)$,其中:

- V_e 是文档中所有元素的集合;
- v_r 是文档的根元素,因为 v_r 也是文档中的元素,所以 $v_r\in V_e$;
- V_a 是文档中所有属性的集合;
- N_s 是名字集合,既包括元素的名字,也包括属性的名字;
- $subelem$ 是一个二元关系, $subelem\subseteq V_e\times V_e$,如果 e_1 和 e_2 都是 V_e 中的成员,那么 $(e_1,e_2)\in subelem$ 表示 e_1 是 e_2 的子元素,或者从 e_2 中存在一个链接关联到 e_1;
- $attrs$ 是一个二元关系, $attrs\subseteq V_a\times V_e$,如果 a_1 是 V_a 中的成员, e_1 是 V_e 中的成员,那么 $(a_1,e_1)\in attrs$ 表示 a_1 是 e_1 的属性;
- $name$ 也是一个二元关系, $name\subseteq N_s\times(V_a\cup V_e)$,如果 n_1 是 N_s 中的成员, v_1 是 V_a 或者 V_e 中的成员,那么 $(n_1,v_1)\in name$ 表示 n_1 是 v_1 的名称.因为同一文档中不同的元素或者属性的名称可能相同,所以 N_s 中的同一个成员可能会映射到 V_a 或者 V_e 中的不同成员.

定义 1 给出的 XML 文档模型是利用 3 个二元关系将文档中的元素、属性以及它们的名字联系成一个整体.当然,必须在这些二元关系上增加一些限制,才能形成正确的 XML 文档.这里,我们不讨论这些限制的详细内容,只是说明 XML 文档都能被这个模型所描述.

如果在 XML 文档中添加标签,则需要在模型中增加一些组件.在 XML 文档中,我们对所有的元素和属性都添加标签.定义 2 是添加了标签的 XML 文档模型.

定义 2. 添加了标签的 XML 文档是一个九元组 $XDoc=(V_e,v_r,V_a,N_s,L_s,subelem,attrs,name,label)$,其中:

- $V_e,v_r,V_a,N_s,subelem,attrs,name$ 与定义 1 相同;
- L_s 是所有的标签集合;
- $label$ 是一个二元关系, $label\subseteq L_s\times(V_a\cup V_e)$,如果 l_1 是 L_s 中的成员, v_1 是 V_a 或者 V_e 中的成员,那么, $(l_1,v_1)\in label$ 表示 l_1 是 v_1 的标签,也可以表示为 $l_1=\lambda(v_1)$.

我们知道,XML 文档的结构是层次型的,根元素是它的最外层单元.除了根元素之外,其他的元素和属性都从属于文档中某个元素.而在获取元素或属性的信息时,必须获取外层元素的部分信息.例如,用 XPath[17]来表示元素的路径及属性时,该元素的所有直接或间接的外层元素都会出现在该路径中.既然在查询内层元素时要涉及外层元素,那么内层元素的安全级别不能比外层元素的低.因此,我们规定元素或者属性的标签必须支配其外层元素的标签.

规则 1. 在一个 $XDoc$ 中, $e_1\in V_e$, $e_2\in V_e$,如果有 $(e_1,e_2)\in subelem$,那么 e_1 的标签支配 e_2 的标签,即 $\lambda(e_1)\geq\lambda(e_2)$.

规则 2. 在一个 $XDoc$ 中, $a_1\in V_a$, $e_1\in V_e$,如果有 $(a_1,e_1)\in attrs$,那么 a_1 的标签支配 e_1 的标签,即 $\lambda(a_1)\geq\lambda(e_1)$.

如果把 XML 文档看成是树型结构,那么根据这两个规则,从树根到树叶安全级别是由低到高发展的.根据标签支配关系的传递性,元素或者属性的标签应该支配所有直接或间接的外层元素的标签.

规则 3. 在一个 $XDoc$ 中, $e_1\in V_e$, $e_2\in V_e$,如果存在 $e_3,e_4,\ldots,e_n\in V_e$,使得 $(e_1,e_3)\in subelem\wedge(e_3,e_4)\in subelem\wedge\ldots\wedge(e_n,e_2)\in subelem$,那么 e_1 的标签支配 e_2 的标签,即 $\lambda(e_1)\geq\lambda(e_2)$.

规则 4. 在一个 $XDoc$ 中, $a_1\in V_a$, $e_1\in V_e$,如果存在 $e_2,e_3,\ldots,e_n\in V_e$,使得 $(a_1,e_2)\in attrs\wedge(e_2,e_3)\in subelem\wedge\ldots\wedge(e_n,e_1)\in subelem$,那么 a_1 的标签支配 e_1 的标签,即 $\lambda(a_1)\geq\lambda(e_1)$.

XML 文档的根元素是所有其他元素或者属性的直接或间接外层元素,因此,它的标签被所有其他元素或属性的标签所支配.而作为元素和属性的载体,XML 文档的标签也应该被其中所有元素和属性的标签所支配,所以,我们可以把根元素的标签作为 XML 文档的标签.

定义 3. XML 文档的标签就是它的根元素的标签, $\lambda(XDoc)=\lambda(v_r)$.

模式(schema)文件为有效的 XML 文档实例建立了内容与结构的约束.模式文件定义了文档实例的元素和属性,同时定义了元素以及属性之间的关系.例如,在图 1 的模式文件中,先定义了元素 *employee*,然后在定义元素 *company* 时引用了 *employee* 子元素.我们使用的模式文件利用元素 *ElementType* 来定义文档实例的元素,并且定义时通过元素 *element* 来引用已经定义好的其他元素.*ElementType* 约束元素的内容,*element* 约束元素的位置,两者共同完成了对 XML 文档实例中某个元素的定义.而在模式文件中 *ElementType* 和 *element* 都有自己的标签,这些标签也可以认为是它们定义的文档实例中元素的标签.为了保持一致性,XML 模式文件中定义和引用同一元素的 *ElementType* 和 *element* 的标签应该是一样的.

规则 5. XML 模式文件中定义和引用一个被定义的元素通过 *ElementType* 和 *element* 来实现,*ElementType* 的标签也是被定义元素的标签,假设该标签为 λ_1,那么所有引用该元素的 *element* 的标签都应该为 λ_1.

模式文件利用元素 *AttributeType* 来定义文档实例的属性,并且在定义元素的时候通过元素 *attribute* 来引用已经定义好的属性.同元素一样,定义和引用同一属性的 *AttributeType* 和 *attribute* 的标签也应该是一样的.

规则 6. XML 模式文件中定义和引用某个被定义的属性通过 *AttributeType* 或者 *attribute* 来实现,*AttributeType* 的标签也是被定义属性的标签,假设该标签为 λ_1,那么所有引用该属性的 *attribute* 的标签都应该为 λ_1.

有效的 XML 文档都符合某个 XML 模式文件,它们的元素和属性在模式文件中都有定义.符合同一个模式文件的 XML 文档中相同类型的元素结构相同,内容相近,所以大部分情况下它们的敏感程度也是相近的.因此,我们可以只在模式文件中定义它们的标签,然后将这些标签缺省地推广到所有符合该模式文件的 XML 文档中.如果图 1 中 company.xsd 定义元素 *employee* 的 *ElementType* 的标签是 λ_1,那么 compony.xml 中所有 *employee* 元素的缺省标签都为 λ_1.但是某些情况下,可能需要特殊对待某些文档或者文档中的某些元素,比如,compony.xml 中关于经理 *zhang* 的元素,因为它记录了管理人员的信息,敏感程度可能高于其他 *employee* 元素,这时,我们只能特别地为这个元素设置标签.如果我们为它单独设置的标签为 λ_2,那么根据强制访问控制模型中信息只能从低级流向高级的原则, λ_2 必须支配 λ_1.

规则 7. 如果 XML 文档中某个元素或者属性 v_1 的缺省标签为 λ_1,而管理员又特别地为 v_1 指定了标签 λ_2,那么必须满足 $\lambda_1 \le \lambda_2$.

我们在指定标签的时候,同样要保证文档中元素的标签是从外到内逐渐提高的.因此,我们在指定元素或者属性的标签时,必须保证指定的标签支配为它所有直接或间接的元素指定的标签.

规则 8. 如果管理员指定了 XML 文档中某个元素或者属性 v_1 的标签,记为 λ_1,而且也为 v_1 的某个直接或间接的外层元素 e_1 指定了标签,记为 λ_2,那么必须满足 $\lambda_1 \ge \lambda_2$.

但是根据规则 1 和规则 2,元素和属性的标签都支配其上层元素的标签,如果我们为 XML 文档中某个元素指定了标签,那么它的子元素和属性都必须支配指定后的标签,如果这些子元素和属性在定义的时候缺省标签比较低,那么我们需要提升这些子元素和属性的标签.根据上面的规则,我们采用算法 1 确定一个 XML 文档中元素或者属性的标签.

算法 1. 获取 XML 文档中某个元素或者属性 v_1 的标签.

(1) v_1 是否有单独定义的标签,如果有,则该标签为 v_1 的标签.否则转入第(2)步;

(2) 从定义该文档的 XML 模式文件中获取 v_1 的缺省标签,记为 λ_1;

(3) 如果是根元素,那么 λ_1 就是 v_1 的标签.否则转入第(4)步;

(4) 获取 v_1 的外层元素的标签,记为 λ_2;

(5) 如果 $\lambda_1 \ge \lambda_2$,那么 λ_1 是 v_1 的标签;如果 $\lambda_1 \le \lambda_2$,那么 λ_2 是 v_1 的标签.否则转入第(6)步;

(6) λ_1 和 λ_2 是不可比的,那么把能同时支配 λ_1 和 λ_2 的最低标签作为 v_1 的标签,记为 $\mathrm{lub}(\lambda_1,\lambda_2)$.

根据算法 1,我们可以确定一个 XML 文档中任何一个元素或者属性的标签.这些元素和属性都将作为访问控制的客体.细粒度强制访问控制模型将会控制用户对所有这些客体的访问.

4 面向 XML 文档的细粒度强制访问控制模型

根据 BLP 模型的两条特性,用户在读写客体的时候必须满足一定的条件.然而,由于 XML 文档结构和内容的特殊性,用户对于它们的操作种类不仅仅限于读和写两种.但是,这些操作实质上都是由读和写构成的,所以我们可以把这些操作转换成读、写或者两者的综合,然后再根据 BLP 的特性进行控制.

4.1 XML文档上的操作类型

用户对 XML 文档的访问操作基本上可以分为 4 类,表 1 简单地介绍了这些操作.

读取操作可以查看文档中元素的内容及其属性值.在文档中创建一个元素则需要提供该元素的内容和属性值.更新操作不仅可以修改元素的内容,也能更改元素的属性值.删除一个元素则会把该元素的所有子元素和属性都删除.

Table 1 Operations on XML documents

表 1 对 XML 文档的访问操作

Access Method	Description
Read	Retrieve information from XML documents
Create	Create new element in XML documents
Update	Update information in XML documents
Delete	Delete element from XML documents

4.2 强制访问控制模型下对于XML文档的操作

在 BLP 模型的控制下,用户和 XML 文档中的内容都有自己的标签.当用户提交对 XML 文档的操作时,系统会获取用户和操作对象的标签,根据这些标签,强制访问控制模型可以判断用户如何执行这些操作.假设 $XDoc=(V_e,v_r,V_a,N_s,L_s,subelem,attrs,name,label)$ 是用户操作的 XML 文档,而 $XDoc'=(V'_e,v'_r,V'_a,N'_s,L'_s,subelem',attrs',name',lable')$ 则是操作后返回的文档.

4.2.1 读操作

根据 BLP 模型的简单安全特性,主体能读取客体的条件是主体的标签支配客体的标签.因此,如果一个用户要读取某个 XML 文档的内容,首先必须满足用户的标签支配 XML 文档的标签.然而,即使满足了这个条件,用户也不一定可以读取这个 XML 文档的所有元素,因为文档中元素的标签是逐渐升高的.如果用户的标签能支配某个元素 e_1 的标签,但不能支配 e_1 的某个子元素 e_2 的标签,那么用户看到的 e_1 中将不包含 e_2.同样,如果 e_1 的某个属性 a_1 的标签也不能被用户的标签所支配,那么用户看到的 e_1 也将没有属性 a_1.所以,用户看到的 XML 文档可能是原有文档的一部分.

假设 *XDoc* 是用户 *u* 想要访问的 XML 文档,根据强制访问控制,系统返回给用户的文档则是 *XDoc'*,其中:

- $v_r'=v_r$, $N_s'=N_s$, $L_s'=L_s$;
- $V_e'=\{e:V_e \mid \lambda(u)\geq\lambda(e)\}$, $V_a'=\{a:V_a \mid \lambda(u)\geq\lambda(a)\}$;
- $subelem'=\{(e_1,e_2) \mid e_1\in V_e' \wedge e_2\in V_e' \wedge (e_1,e_2)\in subelem\}$;
- $attrs'=\{(a_1,e_1) \mid a_1\in V_a' \wedge e_1\in V_e' \wedge (a_1,e_1)\in attrs\}$;
- $name'=\{(n_1,v_1) \mid n_1\in N_s' \wedge (v_1\in V_e' \vee v_1\in V_a') \wedge (n_1,v_1)\in name\}$;
- $label'=\{(l_1,v_1) \mid l_1\in L_s' \wedge (v_1\in V_e' \vee v_1\in V_a') \wedge (l_1,v_1)\in label\}$.

假设 company.xsd 定义 *salary* 的元素时缺省的标签是秘密级,而其他的元素都是普通级.秘密级比普通级高,那么如图 2 所示的是普通级的用户能看到的 company.xml 和 company.xsd.对于普通级用户来讲,*salary* 元素是不存在的.

4.2.2 创建子元素操作

用户创建子元素就是在已存在的客体中创建新客体,根据 BLP 模型的严格的*特性,用户创建的子元素应该和用户有相同的标签.但是该子元素在定义的时候有缺省的标签,根据规则 7,元素的标签应该支配定义这个

元素时给定的缺省标签,因此用户的标签应该支配创建的子元素的缺省标签.同时,用户只能在可读的元素中创建子元素,根据读取操作的性质,用户的标签要支配所有可读元素的标签.所以,如果用户 u 想在元素 e_1 中创建子元素 e_2,应该满足两个条件:

- 用户 u 的标签支配元素 e_1 的当前标签;
- 用户 u 的标签支配元素 e_2 的缺省标签.

```
〈company〉
  〈empolyee name="zhang"〉
    〈department〉manage〈/department〉
    〈office〉No.415〈/office〉
    〈phone〉52338215〈/phone〉
  〈/empolyee〉
  〈empolyee name="wang"〉
    〈department〉personnel〈/department〉
    〈office〉No.311〈/office〉
    〈phone〉52338327〈/phone〉
  〈/empolyee〉
  〈empolyee name="li"〉
    〈department〉sales〈/department〉
    〈office〉No.306〈/office〉
    〈phone〉52338364〈/phone〉
  〈/empolyee〉
〈/company〉
```

(a) employee.xml

```
〈? xml version="1.0" encoding="UTF-8"?〉
〈Schema name="EmployeeSchema"
xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes"〉
  〈AttributeType name="name" dt:type="string"/〉
  〈ElementType name="department" content="textonly"
      dt:type="string"/〉
  〈ElementType name="office" content="textonly" dt:type="string"/〉
  〈ElementType name="phone" content="textonly" dt:type="string"/〉
  〈/ElementType name="empolyee" content="eltOnly"〉
      〈attribute type="name"〉
      〈element type="department" minOccus="1" maxOccus="1"〉
      〈element type="office" minOccus="1" maxOccus="1"〉
      〈element type="phone" minOccus="1" maxOccus="1"〉
  〈/ElementType〉
  〈/ElementType name="company" content="eltOnly"〉
      〈element type="employee" minOccus="0" maxOccus="*"〉
  〈/ElementType〉
```

(b) employee.xsd

Fig.2 Contents of document and schema files that unclassified users can read

图 2 普通级用户看到的文档及模式文件的内容

假设 *XDoc* 是操作之前的 XML 文档,则用户 u 在元素 e_1 中创建了子元素 e_2 之后的新文档为 *XDoc*′.假设 V_e'' 和 V_a'' 分别是 e_2 中所有的元素与属性集合, *subelem*″ , *attrs*″ , *name*″ 和 *label*″ 分别是 e_2 内部的二元关系. e_2 的标签就是 u 的标签.因为 e_2 中所有子元素和属性也是用户 u 创建的,所以它们的标签也都是 u 的标签.如果在定义的时候, e_2 中的某些子元素或属性的缺省标签高于用户 u 的标签,那么用户在创建 e_2 的时候 V_e'' 和 V_a'' 不包括这些高级的子元素和属性.于是 *XDoc*′ 满足:

- $v_r' = v_r$, $N_s' = N_s$, $L_s' = L_s$;
- $V_e' = V_e \cup V_e''$, $V_a' = V_a \cup V_a''$;
- $subelem' = subelem \cup \{(e_2, e_1)\} \cup subelem''$;
- $attrs' = attrs \cup attrs''$, $name' = name \cup name''$;
- $label' = label \cup label''$,其中如果 v 是 V_e'' 或 V_a'' 的元素,那么 $(\lambda(u), v) \in label''$.

4.2.3 更新元素操作

更新元素的操作可以改变元素的内容和属性值.在更新操作之前,用户必须能读取该元素的内容或者属性的值,然后才能改变它们.根据 BLP 模型,只有相同标签的用户才能读写客体.因此,更新操作应该满足以下条件:

- 如果用户 u 想更新某个元素 e_1 的内容,必须满足 $\lambda(u) = \lambda(e_1)$;
- 如果用户 u 想更新某个元素的属性 a_1 ,必须满足 $\lambda(u) = \lambda(a_1)$.

更新操作成功后,除了元素的内容或者属性的值发生了变化外,XML 文档的其他部分都没有发生改变.因此, *XDoc* 的结构在更新操作发生后没有任何变化.

4.2.4 删除元素操作

删除元素会把元素中所有的内容清除,包括它的属性和子元素.被删除元素如果比用户的级别还低,用户就改变了低级数据的内容,显而易见,这违反了 BLP 模型的特性,因此用户只能删除与其标签相同的元素.

但是,用户在删除元素的时候,这个元素可能包含高级的子元素或属性,这些高级子元素或属性对用户来讲是不可见的.如果不希望用户删除高级的信息,那么必然要拒绝用户的删除请求,这样就使得低级用户有可能获

取高级的信息.因为 XML 文档是层次结构的数据,除了根元素之外,其他元素都有外层元素.元素与其外层元素的联系是很紧密的.因此,当我们想要删除一个元素的时候,它的子元素失去了赖以存在的基础,即使是高等级的信息,也可以考虑将它们全部删除.

于是,用户能删除元素的条件是:

- 用户 u 的标签等于要删除元素 e_1 的标签.

删除操作成功后,被删除元素以及它的属性和子元素全部被删除,假设用户删除了 *XDoc* 中的非根元素 e_1, V_e'' 和 V_a'' 分别是元素 e_1 中所有的元素与属性集合, *subelem*″ , *attrs*″ , *name*″ 和 *label*″ 分别是 e_1 内部的二元关系, e_2 是 e_1 的外层元素,那么删除操作成功后的 *XDoc*′ 满足:

- $v_r' = v_r$, $N_s' = N_s$, $L_s' = L_s$;
- $V_e' = V_e - V_e''$, $V_a' = V_a - V_a''$;
- $subelem' = subelem - (\{(e_1, e_2)\} \bigcup subelem'')$;
- $attrs' = attrs - attrs''$, $name' = name - name''$, $label' = label - label''$.

如果用户要删除的元素是 XML 文档的根元素,那么实际上删除的是整个 XML 文档.根据删除操作的条件,XML 文档只能被标签相同的用户删除.

5 XML 文档的强制访问控制模型的实现机制

强制访问控制模型根据安全标签来控制主体对客体的访问,因此需要在系统中保存用户与 XML 文档的标签.如果把这些安全信息也存放在 XML 文档中,那么我们就可以采用 XML 技术来实现强制访问控制.我们把 XML 文档中元素与属性的标签存放在另外的 XML 文档中,这些文档叫安全信息文件,这样有利于扩展现有的面向 XML 文档的系统.对于现有的已经组织好的 XML 文档,只要为这些文档建立安全信息文件,而不需要修改现有文档的内容和结构,就可以把它们纳入强制访问控制模型的管理之下.图 3 是实现 XML 文档的强制访问控制模型的体系结构.

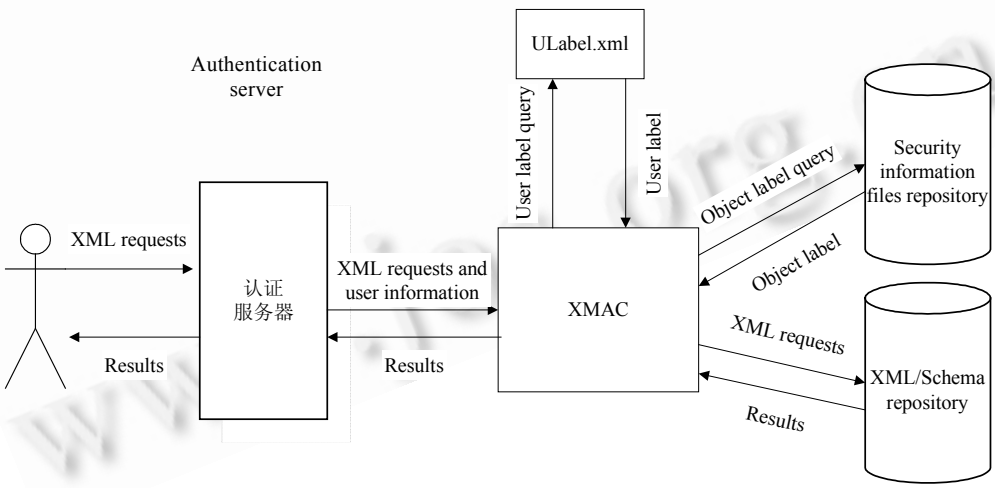

Fig.3 Architecture to implement MAC model for XML documents

图 3 实现 XML 文档强制访问控制模型的体系结构

图 3 中的体系结构使用 XML 文档存放主体与客体的安全信息,ULabel.xml 存放用户的标签信息,而客体的标签则存放在安全信息文件仓库中.安全信息文件仓库中包含许多 XML 文档,每个文档都对应 XML/模式文件仓库中的某个文档.XML/模式文件仓库中的每个模式文件在安全信息文件仓库中都有一个文件与之对应,其中保存着该模式文件中各个元素的标签,这些标签也就是它们定义的元素标签.普通的 XML 文档则不一定存在安全信息文件,只有在管理员为这个文档中的某些元素特别地指定了标签之后,系统才会生成一个文件用来保存

这些标签,而那些没有特殊定义标签的元素在这些文件中是不必记录的.XMAC 是体系结构中最核心的模块,它负责在强制访问控制策略下处理用户的请求.根据存放用户标签的 ULabel.xml 文件和记录 XML 文档元素标签的安全信息文件仓库,XMAC 将把符合强制访问控制策略的操作结果返回给用户.

用户把 XML 请求送给 XMAC 处理之前,必须先通过认证.认证成功后,认证服务器会把 XML 请求与用户的身份一起发送给 XMAC.XML 请求包括了用户想要访问的客体名称以及操作类型,这些客体可能是 XML 文档本身,也可能是文档的某个元素或者属性.XMAC 根据用户的身份从 ULabel.xml 文件中查找到用户的标签,然后根据客体的名称从安全信息文件仓库中获取客体的标签,算法 1 可以帮助 XMAC 确定某个元素或者属性的标签.获得了主体与客体的标签之后,XMAC 根据强制访问控制策略和操作的类型正确地处理 XML 请求.处理完成后,XMAC 将结果返回给认证服务器,认证服务器再将结果返回给用户.这个过程需要在各个组件中传输许多消息,例如,XML 访问请求、操作结果等.如果这些消息符合 XML 语法,那么我们也可以采用 XML 技术来存放和传输这些消息.

6 结 论

XML 访问控制决定用户如何访问 XML 文档中存放的信息.访问控制策略有许多类型,高安全等级系统使用强制访问控制策略保证由系统来控制用户对信息的访问.当高安全等级系统使用 XML 存放数据时,XML 文档必须置于强制访问控制的管理之下.本文提出了一种面向 XML 文档的强制访问控制模型,控制用户对高安全等级系统中 XML 文档的访问.本文首先给出了包含标签信息的扩展 XML 文档模型,并给出了扩展文档模型需要满足的 8 条规则,这些规则保证了 XML 文档中元素和属性标签的合理性.在此基础上,通过讨论针对 XML 文档的 4 种操作,描述了细粒度强制访问控制模型的详细内容.该模型基于 XML 模式技术,它的控制粒度可以达到文档中的元素或者属性.最后讨论了该模型的体系结构和一些实现机制.为了使访问控制模块能良好地结合到面向 XML 的系统中,本文提出的体系结构尽可能地使用 XML 技术来实现强制访问控制模型.面向 XML 文档的强制访问控制模型还有许多方面有待研究,诸如完整性——有效的 XML 文档对任何级别的用户来说都是有效的,无论用户看到的是文档的全部内容还是部分内容;多实例特性——不同级别的用户看见的同一元素或属性的内容不同等.在今后的工作中,我们将对这些特性做进一步的研究.

References:

[1] World Wide Web Consortium (W3C), Extensible Markup Language (XML) 1.0. 2000. http://www.w3.org/TR/REC-xml

[2] Damiani E, Vimercati SDC, Paraboschi S, Samarati P. A fine-grained access control system for XML documents. ACM TISSEC, 2002,5(2):169~202.

[3] World Wide Web Consortium (W3C), XML Schema Part 0: Primer. 2001. http://www.w3.org/TR/xmlschema-0

[4] World Wide Web Consortium (W3C), XML Schema Part 1: Structures. 2001. http://www.w3.org/TR/xmlschema-1

[5] World Wide Web Consortium (W3C), XML Schema Part 2: Datatypes. 2001. http://www.w3.org/TR/xmlschema-2

[6] Damiani E, Vimercati SDC, Paraboschi S, Samarati P. Design and implementation of access control processor for XML documents. Computer Network, 2000,33(1-6):59~75.

[7] Bertino E, Castano S, Ferrari E, Mesiti M. Specifying and enforcing access control policies for XML document sources. World Wide Web, 2000,3(3):139~151.

[8] Beritino E, Castano S, Ferrai E. Securing XML documents with Author-x. IEEE Internet Computing, 2001,5(3):21~31.

[9] Chandramouli R. Application of XML tools for enterprise-wide RBAC implementation tasks. In: SIGSAC ed. Proc. of the 5th ACM Workshop on Role-Based Access Control, Berlin: ACM Press, 2000. 11~18.

[10] Hitchens M, Varadharajan V. RBAC for XML document stores. In: Qing SH, Okamoto T, Zhou JY, eds. Proc. of the Int'l Conf. on Information and Communications Security. Lecture Notes in Computer Science 2229, Springer-Verlag, 2001. 131~143.

[11] Vuong NN, Smith G, Deng Y. Managing security policies in a distributed environment using eXtensible markup language (XML). In: SIGAPP ed. Proc. of the 2001 ACM symposium on Applied computing. Las Vegas: ACM Press, 2001. 405~411.

[12] Fu HY, Li H, Wang YM. Research on XML access control technologies. Net Security Technologies and Applications, 2002,9: 17~20 (in Chinese with English abstract).

[13] Zhao X, Huai JP. XML-Based multi-granularity access control system. Computer Engineering and Applications, 2002,38(21): 155~159 (in Chinese with English abstract).

[14] Zhang XW, Park J, Sandhu R. Schema based XML security: RBAC approach. Technical Report, IFIP WG 11.3, 2003.

[15] Osborn S, Sandhu R, Munawer Q. Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM TISSEC, 2000,3(2):85~106.

[16] Bell DE, LaPadula LJ. Bell-LaPadula model for secure computer systems. Technical Report, 1976.

[17] World Wide Web Consortium (W3C), XML Path Language (XPath). 2002. http://www.w3.org/TR/xpath20

附中文参考文献:

[12] 傅海英,李晖,王育民.XML 访问控制技术研究.网络安全技术与应用,2002,9:17~20.

[13] 赵遐,怀进鹏.基于 XML 的多粒度访问控制系统.计算机工程与应用,2002,38(21):155~159.

△△△

2005 年智能计算及其应用国际研讨会

会 议 通 知

为了适应我国高技术计划“智能计算的理论与应用研究”的需要,推动我国深入开展智能计算的研究,加强国际交流与合作,在国家宇航学会、教育部等资助下,中国地质大学将举办一次高水平的智能计算国际研讨会(International Symposium on Intelligence Computation,简称 ISICA 2005),会议定于 2005 年 4 月 4 日~6 日在中国地质大学举行.会议将邀请演化计算方面各主要分支的国际知名专家来参加此次会议,还将出版论文集并送检索机构检索.欢迎有兴趣者参加此次研讨会.

一、会议的主要议题

神经网络	影像库的自动分析	航天器中的演化硬件
智能 GIS	多目标优化	数字和组合优化
模糊系统	动态优化	机器学习
人工免疫系统	生物信息学	蚁群方法
粒子群方法	演化数据挖掘	遗传算法、遗传程序设计、基因表达式程序设计等在科学数据分析中的应用

二、联系方式

联系地址: 中国地质大学计算机科学与技术系

邮政编码: 430074

联 系 人: 蔡之华 李晖 李振华

联系电话: 027-67883713

电子信箱: jsjb@cug.edu.cn; zhcai@cug.edu.cn