A Flexible Mandatory Access Control Policy for XML Databases[†]

Hong Zhu
Huazhong University of Science and Technology,
Wuhan, Hubei 430074, P.R.China
+86-27-87544400
zhuhong@public.wh.hb.cn

Renchao Jin*
Huazhong University of Science and Technology,
Wuhan, Hubei 430074, P.R.China
+86-27-87792212
jrc@hust.edu.cn

Kevin Lü
Brunel University,
BBS, Room 76, Tin Building, Brunel University, Uxbridge, UK UB8 3PH
+44-01895-265254
Kevin.Lu@Brunel.ac.uk

ABSTRACT
A flexible mandatory access control policy (MAC) for XML databases is presented in this paper. The label type and label access policy can be defined according to the requirements of applications. In order to preserve the integrity of data in XML databases, a constraint between a read access rule and a write access rule in label access policy is introduced. Rules for label assignment and propagation are proposed to alleviate the workload of label assignment. Also, a solution for resolving conflicts of label assignments is proposed. At last, operations for implementation of the MAC policy in a XML database are illustrated.

Categories and Subject Descriptors
H.2.0 **[Database Management]**:Security, integrity and protection

General Terms
Management, Security

Keywords
Database Security, XML database, Mandatory access control

1. INTRODUCTION
XML is widely used in a variety of applications, and has become a standard for describing and exchanging data across the Internet. As more and more XML documents are stored in XML databases, the security of XML databases has become an important issue. Access control is one of the methods used to guarantee the security of XML databases. Access control models for XML databases can be divided into two types: the discretionary access control model (DAC) [2] and the mandatory access control model (MAC) [4, 6]. In the DAC models, a subject can discretionarily control privileges of other subjects accessing an object in a XML database but attacks from a Trojan horse cannot be resisted because of inherent flaws.

In the MAC model based on BLP [1], every object is assigned a label which specifies the security privilege of the object, and every user is assigned a label which specifies what objects he/she can access. The label in [1] is a binary-tuple $L=<l, c>$ consisting of a classification l and a category c. $L(s)$ and $L(o)$ denote the labels for a subject and an object respectively, and the label $L(o) \leq L(s)$ if and only if $L(o).l \leq L(s).l$ and $L(o).c \subseteq L(s).c$. We call $L(o) \leq L(s)$ as $L(s)$ dominates $L(o)$. In [3], when an object is accessed, the label of the subject is compared with the label of the object by the *Simple security property* and **-property*. The MAC security of a system based on the BLP model is sufficient, but the rule for label comparing in BLP model is too rigorous for some cases. The larger the set of category for an object is, the fewer the users that can access it. In some applications [7, 8], the rule for label comparing is not as rigorous as this**.** On the contrary, the requirement is: the larger the set of category, the more users that can access it. Moreover, in these applications the structure of the label may be different from the structure of the label in the BLP model. In order to meet the needs of these applications, [5, 7, 8] have enhanced the flexibility of the MAC mechanism in relational databases. The existing MAC models [4, 6] for XML databases are all based on the BLP model. How to make the flexibility of MAC available to XML databases has not been reported in the literature. In order to enhance the security of XML databases for all-purpose uses, XML databases should provide a flexible MAC. This paper focuses on this problem**.**

The contributions of this paper are:

(1) A flexible MAC policy for XML databases is proposed. The label structure and label access policy can be defined according to the requirements of the applications.

(2) A constraint between the read access rules and write access rules in label access policy has been introduced. The rules for label assignment and propagation are addressed to alleviate the workload of label assignments. The problem for conflicts of label assignments has been solved.

The rest of this paper is organized as follows: Section 2 describes the basic concepts. Section 3 describes the flexible MAC policy for XML databases in detail. Section 4 summarizes the conclusions. Because of the space limit, we do not illustrate our implementation of the policy in a XML database.

†The work of this paper is supported by the National Hi-Tech Research and Development Program (863) of China. Grant No. 2006AA01Z430.

*To whom correspondence should be addressed.

INFOSCALE 2007, June 6-8, Suzhou, China
Copyright © 2007 ICST 978-1-59593-757-5
DOI 10.4108/infoscale.2007.947

2. Basic Concepts

The structure of the label in our MAC policy is that: each label is specified by a label type, and the label type specifies what components the label type consists of. Every component in a label is specified by a label component type. Normally, the structure of the label is specified by the administrator; it can be the same or not the same as the structure of the label in the BLP model.

Definition 1. Label component type. The label component type *LC* is a set of elements $LC = \{c_1, c_2, \ldots, c_m\}$, $m \geq 1$ and it also specifies whether the set is ordered or unordered. If *LC* is an ordered set, then $c_1 \leq c_2 \leq \ldots \leq c_m$ and we call it an ordered label component type. Otherwise, we call it an unordered label component type.

Definition 2. Label component. The label component *lc* is an instance of a label component type. If *LC* is an ordered label component type then $lc \in LC$, otherwise $lc \sqsubseteq LC$.

Example 1. We can define two label component types *secret* and *Dept*. The *secret=*LC_1*={unclassified, secret, Top-secret}* is an ordered label component type with *unclassified ≤ secret ≤ Top-secret*. Any element in LC_1 is a label component as the classification of the label in the BLP model. *Dept=*LC_2*={Technique, HumanResource, Financial}* is an unordered label component type. It denotes the department names of a company. Any subset of LC_2 is a label component as the category of the label in the BLP model. We can define label component types in XML format files.

Definition 3. Label type. A label type is an n-tuple $LT=<LC_1, \ldots, LC_n>$, which is composed of n label component types. Here $n \geq 1$, LC_i is a defined label component type. In order to prevent semantic confusion of the labels for subjects and objects, we specify that there is at most one ordered label component in a label type. If a label type has an ordered label component type, then we specify that LC_1 must be the ordered label component type.

Definition 4. Label. A label $LA = <lc_1, \ldots, lc_n>$ is an instance of a label type $LT =< LC_1, LC_2, \ldots, LC_n >$. Where $LC_i \,(1 \leq i \leq n)$ is a defined label component type, and lc_i is a label component of LC_i.

Definition 5. Label access rule. The label access rule is a four-tuple *LAS=(LA, LT, OP, type),* where *LA* is a set of labels and *LT* is label type of *LA*, $type \in \{r, w\}$ which indicates the rule is a read access rule(*type=r*) or a write access rule(*type=w*), and *OP* is n-tuple $OP=<op_1, \ldots, op_n>$. For any two labels $l_1, l_2 \in LA$, each op_i indicates the operation between $l_1.c_i$ and $l_2.c_i (1 \leq i \leq n)$. For an ordered label component type in *LT*, the $op_i \in$ *{EQ, LE, GE, GT, LT}* which denotes operators in *{=, <=, >=, >, <}* respectively. For an unordered label component type in *LT*, the $op_i \in$ *{IN, INTERSECTION, CONTAIN, EQUAL}*.

Definition 6 label access policy The label access policy is a triple *LP=(S, O, LASS)*. Where *S* is a set of subjects and *O* is a set of objects, and *LASS* is a set of label access rules. The *LASS* consists of a read access rule and a write access rule.

LP indicates when a subject $s \in S$ reads an object $o \in O$, the labels for the subject and the object must conform to the read access rule. The labels for the subject and the object are compared according to the operations specified in *LP.LASS.OP* to determine whether the subject can read the object or not. In the following discussion, if a label *L(s)* and a label *L(o)* conform to a read access rule in *LP.LASS*, it is denoted as $L(s)\omega_{r \text{ in } LP.LASS} L(o)$. If a label *L(s)* and a label *L(o)* conform to a write access rule in *LP.LASS*, it is denoted as $L(s)\omega_{w \text{ in } LP.LASS} L(o)$.

For a label access policy, we can define label access rules in XML format. The format of a rule for specifying an operator between two components in the labels of a subject and an object in XML is:

Subject.component-name <operator> Object.component-name.

The MAC policy compares the labels of a subject and an object according to the label access rules. Our policy supports the *INTERSECTION* operator when we compare two labels; this is different from the other models [4, 6] for XML databases. However, when the operator *INTERSECTION* appears in the read access rule, the domination relationship between two labels cannot be well defined as that in BLP model. So we did not define domination relationship between two labels in our policy.

3. Flexible Mandatory Access Control Policy

When a subject tries to access an object, the read access rule is applied to evaluate whether the labels of a subject and an object are matched. If they are matched, the user can read the object. When a subject tries to insert an object into a XML document, or tries to update or delete an object from a XML document, the read access rule must be evaluated first to locate the object to be inserted or updated or deleted. In our flexible MAC policy any user can read what he/she wrote. Otherwise, the integrity of data is not maintained. We have the following constraint.

Rule 1. The constraint of label access policy. *For a label access policy LP=(S, O, LASS), the write access rule contains the read access rule. Namely, for any subject* $s \in S$*, and any object* $o \in O$*, the privilege of* $L(s)\ \omega_{w \text{ in } LP.LASS}\ L(o)$ *is higher than* $L(s)\ \omega_{r \text{ in } LP.LASS}\ L(o)$*.*

3.1 The labeled subjects and XML documents

The subjects of our MAC policy are the users who access the XML databases or application programs or agents on behalf of users. The administrator creates label types and labels, and assigns them to users. Every user except the administrator has a unique label.

The objects are elements/attributes in XML documents and schemas. We denote the objects in XML documents with XPath[3]. As the labeled schema and its XML documents have the same notations, we formally define the labeled XML document in definition 7. The same principle can be applied for formally defining XML schema.

Definition 7 Labeled XML document. The XML document *XDoc* with labels is an eleven-tuple *XDoc=(Ve, vr, Va, Ns, Ls, LT, LP, elemR, attrR, nameR, labelR)*. We have:

(1) *Ve* is the set of all elements in the document;

(2) *vr* is the root of the document, *vr* is also an element of in the document, $vr \in Ve$;

(3) *Va* is the set of all attributes in the document;

(4) *Ns* is the set of name, including the name of elements and attributes;

(5) *elemR* is a binary-tuple, $elemR \subseteq Ve \times Ve$. If $e1 \in Ve$, $e2 \in Ve$, then $(e1, e2) \in elemR$ denotes *e2* is a sub-element of *e1* or there exists a link in *e1* associated with *e2*;

(6) *attrR* is a binary-tuple, $attrR \subseteq Ve \times Va$. If $e \in Ve$, $a \in Va$ then $(e, a) \in attrR$ denotes *a* is an attribute of *e*;

(7) *nameR* is a binary-tuple, $nameR \subseteq Ns \times (Va \bigcup Ve)$. If $n \in Ns$, $v \in Va \bigcup \in Ve$, then $(v, n) \in nameR$ denotes that *n* is the name of *v*. As different elements or attributes in the same document may have the same names, one member of *Ns* may be mapped into a different member of $Va \bigcup Ve$;

(8) *Ls* is the set of all labels with a label type *LT*;

(9) *LT* is the label type which specifies the structure of the labels in the document;

(10) *LP* is the label access policy, and *LP* determines the set of label access rules including a read access rule and a write access rule;

(11) *labelR* is a binary-tuple, $labelR \subseteq (Va \bigcup Ve) \times Ls$. If $L \in Ls$, $v \in Va \bigcup Ve$, then $(v, L) \in labelR$ denotes *L* is the label of *v*, or $L=L(v)$. Different elements or attributes may have the same label, and every element or attribute has only one label.

For the document not being labeled, it is a seven-tuple *Doc= (Ve, vr, Va, Ns, elemR, attrR, nameR)*. The meanings of these symbols are the same as those in the *XDoc*. In the following discussion, we use *XDoc* and *XSch* to denote a labeled XML document and schema respectively. And from the definition of *XDoc*, we have: for any label access rule $las \in XDoc.LP.LASS$, $XDoc.LT=las.LT$.

3.2 Label assignment rules for XML objects

Multiple label access policies can be defined for different security requirements but every document can only be assigned one label access policy. After a XML document or schema is loaded, we should first assign label access policy for them. Then, labels for elements or attributes in XML documents and schema are assigned.

3.2.1 The constraint of label access policy between XML schema and documents

A XML schema defines a set of XML documents with the same structure and similar content. The label access policy should be the same for the schema and its documents.

Rule 2. The constraint of label access policy between XML schema and document. *If the labeled XML schema and its document are XSch and XDoc respectively, then XSch.LP=XDoc.LP.*

3.2.2 The rules for label assignment and propagation

When a XML schema is created, the label for the root of the schema should be assigned. For an ordinary user, when he/she creates a XML schema, the label for the root of the schema is equal to the label of the user. We have the following *Rule 3*.

Rule 3. The label for the root of XML schema. *For an ordinary subject s, if s creates a XML schema sch, then L(sch.vr)=L(s).*

If the schema is created by the administrator, the label must be assigned explicitly. There are a large number of elements and attributes in a XML document. If each element or attribute is assigned a label, the workload for management of these labels is high. We can make use of the features of XML to alleviate the administrator's workload.

The administrator only needs to assign labels to XML schemas and some elements in XML documents. Then, the labels are propagated to the instances of the elements or attributes in XML documents of the schema, or the labels are propagated to descendent elements and attributes of the labeled elements downward from root to leaves in XML documents.

Rule 4. The label propagation from XML schema to XML documents. *For labeled XDoc and XSch, an instance object* $io \in XDoc.Va \bigcup XDoc.Ve$, *and a schema object* $so \in XSch.Va \bigcup XSch.Ve$, *assume io is one of the instances of the schema object so, then L(io)=L(so).*

Rule 5. The label propagation from an element to its sub-elements and attributes. *Assume XDoc is a labeled XML document, for any element* $e1, e2 \in Ve$ *(or attribute* $a1 \in Va$*), if* $(e1, e2) \in elemR$ *(or* $(e1, a1) \in attrR$*), then L(e2)=L(e1)(or L(a1)=L(e1).*

3.2.3 The solution for conflictions of label assignments

Rule 4 and *Rule 5* enhance the flexibility of label assignment and alleviate the workload of the administrator, but may cause assignments of several different labels to one object. For example, an element in a XML document may have three labels. One is propagated from its ancestor, one is propagated from the schema, and the last one is assigned directly by the administrator. In order to guarantee that every object in a XML document has only one label, we introduce a rule for solving label assignment conflicts.

Rule 6. The label calculation rule. *Assume a read access rule lasr=(L, LT, OP),* $L_1 =<a_1, a_2, ..., a_n> \in L$ *and* $L_2 =< b_1, b_2, ..., b_n > \in L$ *are two labels of label type LT, if* L_1 *and* L_2 *are two labels assigned to the same object by direct assignment or by propagation respectively, we can calculate a new label* $L_3=<c_1, c_2, ..., c_n>$ *for the object, where each component of* L_3 *is calculated as follows:*

(1) For the ordered label components type in the LT, if the operator in the $lasr.OP.op_i$ *is:*

(i) GE or GT, then $L_3.c_1=max(L_1.a_1, L_2.b_1)$;

(ii) LE or LT, then $L_3.c_1=min(L_1.a_1, L_2.b_1)$;

(iii)EQ, then $L_3.c_1=max(L_1.a_1, L_2.b_1)$;

Here the $max(l_1, l_2)$ *is a function to calculate the maximum of* l_1 *and* l_2, $min(l_1, l_2)$ *is a function to calculate the minimum of* l_1 *and* l_2.

(2)In the following formulas, if there is an ordered label component type in the label type then $i \geq 2$, *otherwise* $i \geq 1$. *For the*

unordered label components type in LT, if the operator lasr.OP.op$_i$ is:

(i) IN, then $L_3.c_i = L_1.c_i \cap L_2.c_i$;

(ii) CONTAIN, then $L_3.c_i = L_1.c_i \cup L_2.c_i$;

(iii) INTERSECTION, then $L_3.c_i = L_1.c_i \cap L_2.c_i$;

(iv) EQUAL, then $L_3.c_i = L_1.c_i$.

Rule 6 can be extended to calculate a unique label for an object which is assigned three labels due to direct assignment or propagation. Because the least upper bound does not exist for *INTERSECTION* operator in the sense of the least upper bound defined for *CONTAIN* operator in [4], a simple calculation rule for *INTERSECTION* operator is used in *Rule 6*. Namely, when *INTERSECTION* operator is specified for some component in a label access policy, we specify that the component of the result labels is the intersection of corresponding components of two labels. The case may occur in which some object cannot be accessed by any users except the administrator or the creator of the XML document which the object belongs to. Although we lose some availability, the secret is kept. We use a function *label_comput(L_1, L_2)* to denote the result from *Rule 6* where L_1, L_2 are two labels.

3.2.4 Operations for implementation of the MAC policy in a XML database

We implemented our flexible MAC access control policy in a XML database management system. As for the space limit, we do not illustrate the architecture of the system and experiments for performance comparison. We only discuss the operations for implementation of the MAC policy.

The query operations deal with read-only operations. When a user submits a query request, the system parses the query request and searches the label access rules. Then, the query is rewritten according to the label of the user and label access rules, and then the rewritten query is executed. By this way, we can prevent users from inference inferring unauthorized data.

The update operations may impact on the labels or structure of XML documents. When a subject loads a XML document, the label of the subject is compared with the label of the root of the schema corresponding to the XML document. If they conform to the write access rule in the label access policy, the XML document is loaded and the label of the root is calculated from the label of the subject and the label of the root of the schema of the loaded XML documents by *Rule 6*.

When a subject modifies the objects or deletes some objects in a XML document, the label of subject may not be equal to the label of the objects. If the update operation would change some other user's access privileges covertly, this would cause security problems such as covert information transmission. We permit the updating operation success only when the label of subject and the label of the objects conform to the write access rule in the label access policy.

For insertion operation, we should first find the object o_1 which is the parent of the object o_2 to be inserted. According to *Rule 4* and *Rule 5*, the inserted object o_2 may have two labels propagated from its ancestor and schema respectively. We should calculate $L(o_2)$ from *Rule 6* and then compare the label of user with $L(o_2)$. If the label of user and $L(o_2)$ satisfy the write access rule in the label access policy, the insert operation is permitted.

4. Conclusion

Providing a flexible mandatory access control policy for XML databases is important for many applications. We have proposed an approach which can provide MAC policies for different purposes with different requirements, including multilevel secure XML database systems. In our MAC policy, label access policies can be defined according to the requirements of various applications, which can enhance the flexibility of MAC policy in general. A constraint between the access control rules for a read access rule and a write access rule is proposed to maintain the integrity of data. Rules for label assignment and propagation are proposed to alleviate the workload of label assignments, and a rule for solving label assignment conflicts is also proposed. Moreover, operations for implementation of the MAC policy in the XML database are discussed. The MAC policy in this paper can be regarded as a generalization of the BLP model.

REFERENCES

[1] D.E. Bell and L.J. LaPadula, “Secure Computer Systems: Unified Exposition and Multics Interpretation,” Technical Report MTR-2997, The Mitre Corp., Bedford, Mass., 1976.

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

[3] J. Clark and S. DeRose. XML Path Language (XPath). W3C Working Draft, Nov. 1999.

[4] Lan Li, Xinghao Jiang, and Jianhua Li, Enforce Mandatory Access Control Policy on XML Documents, in Proceedings in 7th International Conference of Information and Communications Security:, ICICS 2005, Beijing, China, December 10-13, 2005.

[5] Primary Author: Jeffrey E. Levinger. Oracle® Label Security Administrator’s Guide10*g* Release 1 (10.1) Part No. B10774-01, December 2003.

[6] SungRan Cho, Sihem Amer-Yahia, Laks VS Lakshmanan, and Divesh Srivastava, Optimizing the Secure Evaluation Of Twig Queries. Int. Conference On VLDB, 2002.

[7] Walid Rjaibi, Paul Bird. A Multi-Purpose Implementation of Mandatory Access Control in Relational Database Management Systems. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[8] Xiaodong Yuan, Ying Feng, The Model of Mandatory Access Control with Extended Security Label, Chinese Journal of Computers, October 2000: 1096~1100.