Two-stage Access Control Model for security XML documents

Wei Sun , Da-xin Liu

College of Computer Science and Technology, HARBIN Engineering University,
HARBIN Heilongjiang Province, China
sunwei78@hrbeu.edu.cn

Abstract. XML has become a standard format for data interchanges over the Internet, especially in the electronic commerce. Due to its extensibility, XML was widely adopted in various applications. Although it is extremely important to protect XML documents from being illegally modified or accessed, the development of XML security is still in its infancy stage. XML access control has become an major research topics. Even though there are some traditional access control models, with many different aspects from normal documents, a new method to perform access control on XML warehouse is necessary. In this paper, we proposed a two-stage access control model, which adopts MAC policy used by includes high security system. The model includes two stages: file-level access control and element-level or attribute-level. In particular, our model allows (i) we adopt XBLP policy to file-level security; (ii) Hide-Node View is proposed for element-level security. Companied with other access control algorithms, our model is suitable to high security XML system, and deals with protecting schema information of XML to non-authorization. Furthermore, the overall architecture of the two-stage access control model and implementation are described in detail.

1 Introduction

XML is rapidly emerging as the new standard for data representation and exchange on the Internet. As large corporations and organizations increasingly exploit the Internet as a means of improving business-transaction efficiency and productivity, it is increasingly common to find operational data and other business information in XML format. In light of the sensitive nature of such business information, this also raises the important issue of securing XML content and ensuring the selective exposure of information to different classes of users based on their access privileges. A number of standards such an XACML (OASIS standard) and XACL have already emerged to address the problem of access control for XML documents.

Access control for XML database is non-trivial subjects. And it is different from already existing approaches in relation databases, or file systems for a number of reasons:

- The semi-structured nature of xml documents: unlike file hierarchies or relational tables where the structure is known ahead of time, XML documents do not necessarily have a schema;
- The dependence relation between child node and parent node: unlike relational tables that usually exist as stand-alone entities, an XML node lives with respect to its ancestors, and its children are dependent on the node itself. For instance, a meaningful scenario in the XML but not in the relational context, is the requirement that when a node is granted access, then access is also granted to its descendants;
- The content and schema of XML: XML access control policies all have the file system policy as a special case, but they go much further. For one thing, in the XML context it is useful to specify access control rules that apply to a node if some condition is true (e.g. an attribute of the node has a specific value). As mentioned previously, the big difference with file systems is the absence of a schema in XML. The schema of XML is also needed to protect from unauthorized accessing.

XML has become an active area in database research. XPath and XQuery from the W3C have come to be widely recognized as query languages for XML. In this paper, we are concerned with fine-grained (element-level and attribute-level) access control for XML database systems and document-level access control. We proposed a two-phase access control model for XML database systems.

The rest of this paper is organized as follows. In section 2, we first describe the policy of MAC, that is used in file-level access control. In section 3, we proposed a novel access control policy, named hide-node view, based on DTD too. The policy adopts the merit of security view and enhances the concept of hide-node to conceal the schema information. In section 4, we describe the design and implementation of the two-stage access control system architecture. In the last section, the conclusion and other challenges are proposed.

2 Related Works

A number of recent research efforts have considered access control models for XML data. XACML[6] and XACL [5] propose standards for the access control police of XML documents, but the standard did not specify how to implement the access control model of XML. Dimiani and Bertino discussed the access control based on the DTD of XML documents. Reference [4] proposed a fine grained access control model. The model is based on element-level and attribute-level access control, but the model is suitable to high security system, based on the discretionary access control (DAC). Li L[11] proposed a model of access control based on mandatory access control (MAC). The model implement the fine grained access control again, but MAC policy is not suitable the hierarchical nature of XML documents. In reference [12], role based access control (RBAC) is proposed and RBAC is used on XML documents later. Using RBAC to perform access control in an XML document is not adequate., because RBAC perform access control in file-level, and cannot perform access

control within a XML document. Reference [9] proposed the idea of security view, and security views based on DTD of XML implement to revaluate query on XML documents. But it reveals the information of schema, that can not be known by some low-level access. Reference [10] compares the several models and policies mentioned above using XPath and specifies each aspect of the access control policies.

Our Contributions. Our first contribution is a novel model for specifying XML security access control. Given an XML document accompanied by a document DTD, we allow a two-stage access control policies to pledge to security access XML document at file-level and element-level respectively. Our security specification model supports MAC policy, and is very suitable for high security system. On the element-level access control, our approach for these access control policies is based on the novel notion of *hide-node views*. While the hide-node view DTD is exposed to authorized users, neither the internal XPath annotations nor the full document DTD is visible. Authorized users can only operate data over the hide-node view, making use of the exposed view DTD to access data. Our hide-node view mechanism guarantees that unauthorized user cannot access sensitive data and protects the schema information from access by unauthorized users. We think that the schema information also is sensitive data and should be protected from gain through the data accessing.

3 File-level Access Control Based on XBLP

The UNIX file system has a built-in access control mechanism with users and groups. FreeBSD 5.X introduced new security extensions from the TrustedBSD project based on the POSIX®.1e draft. Two of the most significant new security mechanisms are file system Access Control Lists (ACLs) and Mandatory Access Control (MAC). Mandatory Access Control allows new access control modules to be loaded, implementing new security policies. If a high security system will adopt XML document to store data, MAC need to be used in file-level access control. Thus the XML security system utilizing MAC features should at least guarantee that a user will not be permitted to change security attributes at will; all user utilities, programs and scripts must work within the constraints of the access rules provided by the selected policies; and that total control of the MAC access rules are in the hands of the system administrator.

BLP Model is a famous model of MAC, proposed by Bell DE and LaPadula LJ at 1976. The BLP Model describes access by active entities, called subjects, to passive entities, called objects. One entity can, depending on type of access, be in both roles. Each current access of a subject S[i] to an object O[j] in operator p is treated as a triple (S[i], O[j], p). All these triples together form the set of current accesses b. All authorized accesses of all subjects to all objects are held in the matrix *M*. Each cell *M[i,j]* of *M* thus contains a subset of *A* with authorized accesses of S[i] to O[j]. A security level is a pair (C, G), C means Security Classification and G means Set of Categories. A security classification is a value out of a hierarchy, e.g. public, confidential, secret, top secret. A category is a formal assignment to a work area. Thus there are two relations: dominates and non-compare. One entity with security

level λ (C[1],G[1]) dominates another entity with λ (C[2],G[2]), if C[1]>=C[2] and G[1] is a superset of G[2]. The property *dominates* over all entities builds a partial order *D*.

The three properties lead to the following rules for access control decisions. A current access (S[i],O[j],p) is only granted, if the following conditions are met:

(a)ss-property: λ (S[i]) dominates λ (O[j]), if p = read.

(b)-property*: λ (O[j]) dominates λ (S[i]), if p = write.

(c)ds-property: x is in cell *M[i,j]* of matrix *M* of authorized accesses.

The first property to be maintained is the simple security property (no read-up). This property states that a subject S[i] may have read access to an object O[j] ((S[i],O[j],read) or (S[i],O[j],write) is a current access), if S[i] dominates O[j]. To prevent copying of an object to a lower security level by a malicious subject, the **-property* (no-write-down) must be maintained. This property states: If a subject S[i] has current read access to an object O[1] and current write access to an object O[2], then O[1] must be dominated by O[2] (O[1] has a lower security level than O[2]). Thus information flow is restricted to upwards.

As a strict accordance to the *-property would significantly reduce the usability of the system, some subjects can be marked as trusted subjects without *-property restriction.

Access control by the matrix *M* of authorized accesses by subjects to objects is called discretionary access control, its security property is called *ds-property*. The current access must always be in the set of authorized access in its matrix cell.

All properties and security levels must be mandatorily enforced by the system. Every property is added to the other ones and can never reduce system security. A state that fulfils all properties is called secure.

The ss-property and *- property pledge that the information changes between two same levels or from low level to high level. For XBLP, BLP Model for XML, we use restrict *-property to replace of *-property, and not used *ds-property.*

*(b) restrict *-property:* λ *(O[j]) =* λ *(S[i]), if p = write.*

4 Element-level Access Control Based on Hide-node Views

In this section we present our view-based security model. We define the concepts of hide-node views, and propose an efficient algorithm for automatically deriving a sound and complete hide-node view definition from an access specification.

4.1 The Security Problem

For showing the security problem, we use the Example as follow. Consider the example of reference [9], and the DTD represented as a graph in Fig. 1. A hospital document conforming to the DTD consists of a list of departments (dept), and each dept has children describing clinical trials, patients, and medical staff (doctors and nurses) in the department. A staff member can be either a nurse or a doctor, indicated

by dashed edges; similarly for treatment. The patient data is organized into two separate groups depending on whether or not the patients are involved in clinical trials.

Suppose that the hospital wants to impose a security policy that authorizes nurses to access all patient data except for information concerning whether a patient is involved in clinical trials. A plausible solution to enforce this policy is allow nurses to access patientInfo, while denying their access to clinicalTrial, regular, trial, and test; moreover, the document DTD is revealed to nurses so that they can formulate their queries, like [6]. However, even though nurses can not explicitly refer to the clinicalTrial element in their queries, it is still possible for them to circumvent this restriction and infer the identity of patients in clinical trials via (multiple) queries by exploiting the DTD structure. Indeed, the confidential information can be deduced from the results of the following two permissible queries:

p_1: //dept//patientInfo/patient/name,

p_2: //dept/patientInfo/patient/name.

The difference between the results of p_1 and p_2 in conjunction with the full DTD tells exactly which patients are in clinical trials. Specifically, while p_2 finds the names of all patients not involved in clinical trials (via the path hospital/dept/patientInfo), p_1 actually finds the names of *all patients* (via hospital/dept/($\varepsilon \cup$ clinicalTrial)/patientInfo).

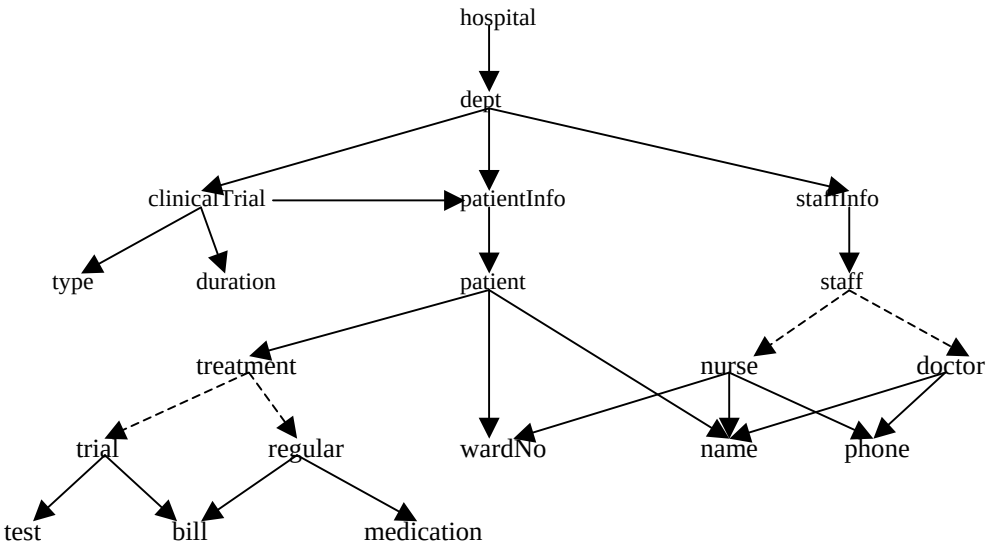

Fig. 1. The DTD of XML document

If there is an authority to nurses, the view DTD hides confidential information, e.g., clinicalTrial, trial, regular. The XPath annotations can specify how to extract relevant accessible data from the original document. While the view DTD is provided to the nurses, the XPath annotations are hidden from them. The existence of element

clinicalTrial, trial and regular is also hidden from them. In other words, the schema is a kind of data for XML too. The nurses have not been authorized to know the existence of these elements. The query //dept/patientInfo/patient/treatment/ trial/bill should be answered by nurses, although the element bill can be accessed. If the query has results, the nurses, who are low authorized user, can gain the schema information of //dept/patientInfo/patient/treatment/ trial/bill.

4.2 Access Specification

An *access specification S* is an extension of a document DTD *D* associating security annotations with productions of *D.* Specifically, *S* is defined to be (D, acl), where acl is a partial mapping such that, for each node (element type B) in D. acl(A, B, op) is an annotation of the form: acl(A, B, op)= $Y\left|\left[q\right]\right|N$, where $\left[q\right]$ is a qualifier based on XPath. Intuitively, a value of Y, $\left[q\right]$, or N for acl(A, B, op) indicates that the element B in an instantiation of D are accessible, conditionally accessible, and inaccessible, respectively. If acl(A,B, op) is not explicitly defined, then B inherits the accessibility of its parent. On the other hand, if acl(A,B, op) is explicitly defined, it may override the accessibility of parent. If the D is accessible, the root of D is annotated Y by default.

4.3 Hide-node Views

Abstractly, a hide-node view defines a mapping from instance of document DTD D to instances of a marker-view DTD D_v . Let S=(D, acl) be an access specification. A hide-node view definition by $V:S\to D_v$, is defined as a pair $V=(D_v,\sigma)$, where σ defines XPath annotations used to extract accessible data from an instance of D. Specifically, for each element B in D_v , $\sigma(B,op)$ is an XPath set defined over document instance of D such that. If the document is accessible, A special case is $\sigma(r_v,op)=r$, where r_v is the root type of D_v and r is the root D, i.e., σ maps the root of T to the root of its view.

4.4 Hide-node Views Derivation Algorithm

We now present a novel algorithm that, given an access specification S=(D, acl), automatically computes a hide-node view definition $V=(D_v,\sigma)$ w.r.t. S such that, for any instance T of the document DTD, if the computation of T_v terminates (i.e., does not abort), it comprises all and only accessible elements of T w.r.t. S_ In particular, V is sound and complete w.r.t. S if and only if such a view definition exists for S. Algorithm is shown in Fig.2.

Procedure build_view(S, A)
Input: specification S=(D, acl) and an element type A in D
Output: hide-node view $V = (D_v, \sigma)$ for A and its descendants in D
If visited[A] then return else visited[A]:=true;
Case the A-production $A \to \alpha$ in the document DTD D of
(1) $A \to B_1, ..., B_n$:
  for i from 1 to n do
    if acl(B_i ,op)=Y then $\sigma(B_i, op) = \sigma(A, op)$; build_view(S, B_i);
    else if acl(B_i ,op)= $[q]$ then $\sigma(B_i, op) = \sigma(A, op) \cup [q]$; build_view(S, B_i);
    else $\sigma(B_i, op) = \phi$; build_view(S, B_i);
(2) $A \to B_1 + ... + B_n$:
  /*similar to (1)*/
(3) $A \to B*$:
  /*similar to (1)*/
(4) $A \to str$
  /* do null */
for i from 1 to n do
  reduce_path($\sigma(B_i, op)$); //expression $\sigma(B_i, op)$ reduction
return;

Fig. 2. Hide-node Views Derivation Algorithm

For example, we see the hide-node view of nurses.
Production: $hospital \to dept*$
$\sigma(dept, op) = hospital / dept[*/ patient / wardNo = \$wardNo]$
Production: $dept \to patientInfo$
$\sigma(patientInfo, op) = hospital / dept /(\in \cup clinicalTrial)[*/ patient / wardNo = \$wardNo]$
Production: $treatment \to trail + regular$
$\sigma(trial, op) = \Phi$, $\sigma(regular, op) = \Phi$
Production: $A \to \alpha$
$\sigma(B, op) = \sigma(A, op)$ /*for all $B \in \alpha$ */

5 Two-stage Access Control Model

In this section, Two-stage Access Control Model will be introduced. Firstly, In order to manage different documents in XML database, an authentication server component,

plays an administrator role in XML database. Secondly, another two components are XBLP component for file-level access control and Hide-node views (element-level access control) component for XML document partial access. When a user wants to access to query some XML document, he or she must enter his valid information given by administrator. Then the authentication server component sends authentication information to Security information repository. The next step is that XBLP component handles the request of the user for certain XML document, and determines whether the user can access the XML document. If the user can access the XML document, XBLP component passes the request to Hide-node views component through the message broker. After Hide-node views component receives the information from the broker, it starts to retrieve the data that the user can access. Intuitively, XPath language is used to help Hide-node views component to specified which region element can be access. The Hide-node views component will rewrite the user requests, and send rewrited requests to XML database. The file-level access control component and the element-level access control component constitute the XMAC, MAC policy for XML.

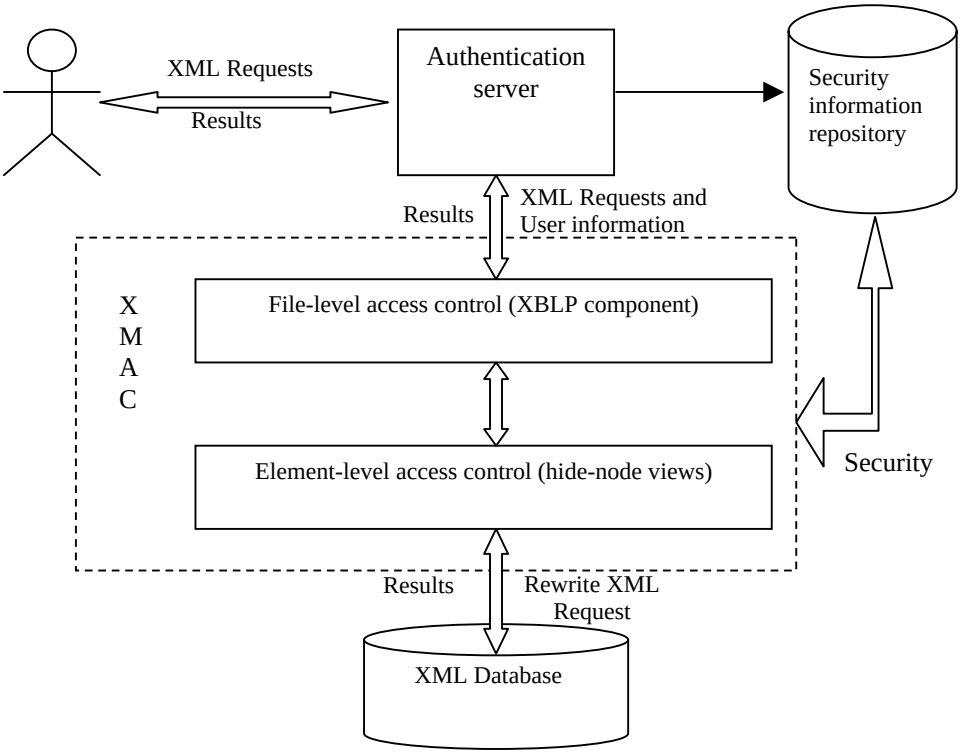

Fig. 3. Two-stage Access Control Model

5 Conclusions

We have proposed a new model for high securing XML data (based on the MAC security policy) and thoroughly studied its algorithm on the Hide-node views. This yields the first XML security model that provides both content access control and schema availability. There are file-level and element-level access controls on the model, implemented by XBLP and Hide-node views respectively. Further, our model can be implemented in query engine for query rewriting and optimization for XML. We are validating these techniques yield substantial reductions in processing time through experimental results. At a summery, this two-stage model not only provides fine-grained access control for XML document in file-level and element-level, but also provides much flexibility compared to the transitional access control system.

References

1. E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *TISSEC*, 5(3): 290-331, 2002.
2. S. Cho, S. Amer-Yahia, L. Lakshmanan, and D. Srivastava. Optimizing the secure evaluation of twig queries. In *VLDB*, 2002.
3. E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati. Securing XML documents. In *EDBT*, 2000.
4. E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *TISSEC*, 5(2):169-202, 2002.
5. S. Hada and M. Kudo. XML access control language: Provisional authorization for XML documents. http://www.trl.ibm.com/projects/xml/xacl/xacl-spec.html.
6. Oasis. eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/xcaml.
7. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, 2003.
8. M. Murata, A. Tozawa, M. Kudo, and S. Hada. XML access control using static analysis. In *CCS*, 2003.
9. W. Fan, C-Y. Chan, and M. Garofalakis. Secure XML Querying with Security Views. In SIGMOD, 2004.
10. Irini Fundulaki, Maarten Marx. Specifying Access Control Policies for XML Documents with XPath. In SACMAT, 2004.
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.
12. R.Sandhu, E.J.Coyne, H.L.Feinstein, “Role based access control models”, IEEE Computer, 29, (2), pp.38-47, 1996.
13. Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari, "TRBAC: A Temporal Role-Based Control Model", ACM Transactions on information and System Security", vol.4, no.3, Aug.2001, Pages 191-223.