Relational Storage for XML Rules

A. A. Abd El-Aziz
Research Scholar
Dept. of Information Science & Technology
Anna University
Email: abdelazizahmed@auist.net

A. Kannan
Professor
Dept. of Information Science & Technology
Anna University
Email: kannan@annauniv.edu

ABSTRACT

very few research works have been done on XML security over relational databases despite that XML became the de facto standard for the data representation and exchange on the internet and a lot of XML documents are stored in RDBMS. In [14], the author proposed an access control model for schema-based storage of XML documents in relational storage and translating XML access control rules to relational access control rules. However, the proposed algorithms had performance drawbacks. In this paper, we will use the same access control model of [14] and try to overcome the drawbacks of [14] by proposing an efficient technique to store the XML access control rules in a relational storage of XML DTD. The mapping of the XML DTD to relational schema is proposed in [7]. We also propose an algorithm to translate XPath queries to SQL queries based on the mapping algorithm in [7].

  Keywords: XML Security, XML Rules, Relational Database, XPath queries, SQL

I. INTRODUCTION

XML is the de facto standard for the data web exchange. It is being increasingly adopted in communication networks. XML documents over the World Wide Web include critical information such as financial and scientific data. It is required to apply restriction technique on the content of XML docu- ments to protect these sensitive data. Thus, only users who are authorized to access fragments of the XML documents should be granted. "Access control for XML documents should ideally provide expressiveness, modularity, interoperability, and efficiency. The expressiveness assures that a wide range of security policy specification may be written. Modularity concerns policies composition while interoperability concerns the ability of policies to interact. Finally, efficiency assures the ability to determine whether an access to an element is granted or denied by the security policy" [1]. XML documents are stored in native XML databases, relational databases, or hybrid relational-XML databases. There are many algorithms proposed to secure native XML documents, such as [2], [4], [6], [9]. However, there are a few algorithms proposed to apply XML access control technique for XML documents stored in relational databases, such as [10], [11], and [15]. In [10], [11], the XML documents stored in relational database without using XML schema. The proposed technique in [15] used XML schema in storing XML documents in relational database, but it had the

following drawbacks [14]:

1) The proposed technique didn't store the access control rules in the relational database. So the The proposed technique was not a pure relational technique.
2) The proposed technique conducted the access control rules from outside the relational engine which caused a performance overhead.

The author in [14], tried to overcome the drawbacks of [15] by

providing an access control model for schema-based storage of XML documents in relational storage and translating XML access control rules to relational access control rules. However, the proposed algorithms had performance drawbacks, because If a policy rule contains type R and an object XPath expression with descendant axis '//' at the same time, then:

1. The XPath expression of the object should be expanded to its underlying simple path expressions.
2. Each one of these path expressions should be expanded to the extended simple path expressions which include their descendant nodes which lead to complexity of the performance if the XML document is large.

In this paper, we will use the same access control model of [14] and try to overcome the drawbacks of [14] by proposing an efficient technique to store the XML access control rules in a relational storage of XML DTD. The proposed technique based on the proposed algorithm in [7] that maps the XML DTD to relational schema. Many works considered access control for XML documents, such as [13], [5], [8], and [12]. In [13], the authors introduced a static analysis for XML access control. However, their prototype incurred costly runtime security checks for queries.In [5], the authors proposed a complex and expensive prototype to express access control policies using XPath queries. In [8], the authors proposed a security views concept. Their prototype restricted on hiding node values. The authors in [12] solved the limit of [8] by considering constraints based on structural relationships between elements [1].

II. STORING XML ACCESS CONTROL RULES IN A RELATIONAL STORAGE

In this section, we introduce our proposed technique that stores XML access control rules in a relational storage of XML DTD. The technique based on XML access control model that proposed in [14], the proposed algorithm in [7] that maps XML DTD to a relational schema, and the proposed algorithm that translate XPath queries to SQL queries based on the mapping algorithm in [7].

A. ACCESS CONTROL MODEL

In [14], the author proposed access control model that allows security administrators to control the access of XML documents by specifying all the authorization information in XML. The model defines access control rules at the level of the XML node. In the model, an authorization rule is a tuple of the form (subject, object, condition, action, type) where:

- subject (role) denotes to a group of users.

- object refers to the group of data that the subject is allowed to access.
- condition denotes the optional XPath predicate applied to the object node.
- action refers to the type of action (read, update, delete)
- that the subject is not allowed on the object.
- type indicates whether the rule affects only the object or propagates to the descendants of the object.

The model only considers deny rules, but in our technique
we will consider grant rules. The model uses a fragment of XPath which can include child, descendant and attribute axes to identify the objects in a rule tuple. The model allows using restricted XPath predicates to compare target text nodes or attributes against constants in XPath expressions of rule tuples. The model allows a security administrator to define local or recursive access control rules. While a local rule only affects the object node, a recursive rule is applied to the object node as well as the descendants of the object node. Thus, the policy allows security administrators to define a mix of authorizations easily for an object node and its descendants and/or ancestors without causing any conflict. Figure 1 shows an example of XML access control rule. The rule tuple (student,//address, true, read, R) means that any student (subject) can read the address (object) node regardless of its ancestors. If the Condition is not exist, It means that it has a default true value as there is no XPath predicate applied to the address node. The type R denotes that the student can also read all the descendants of the address node. The type value may be L that means the path will not extended.

Fig. 1. an XML rule

B. Mapping XML DTD to Relational Schema

Our proposed technique based on the proposed algorithm in [7]. The algorithm maps the various definitions in a given XML DTD, such as elements, attributes, parent- child relationships, ID-IDREF(s) attributes to entities and relationships, describes how to handle the Union types that are not present in relational model, and shows that XML's ordered data model can be efficiently supported by the unordered relational data model. The following example

shows an XML DTD for a book as an input to the algorithm
and the corresponding relational schema which is the output of the algorithm:

Example 1.:

```
!DOCTYPE  book [
  !ELEMENT  book (booktitle, author)i,
  !ELEMENT  booktitle (# PCDATA)i,
  !ELEMENT  article (title, author*, contactauthor)i,
  !ELEMENT  contactauthor, EMPTY i,
  !ATTLIST  contactauthor, authorID, IDREFF # IMPLIED i,
  !ELEMENT  monograph (title, author, editor)i,
  !ELEMENT  editor (monograph* )i,
  !ATTLIST  editor name CDATA # REQUIRED i,
  !ELEMENT  title (=PCDATA)i,
  !ELEMENT  author (name, address)i,
  !ATTLIST  author   id  ID # REQUIRED i,
  !ELEMENT  name (firstname' , lastname)i,
  !ELEMENT  firstname(# PCDATA)i,
  !ELEMENT  lastname (# PCDATA) i,
  !ELEMENT  address (city, zip) i,
  !ELEMENT  city (=PCDATA)i,
  !ELEMENT  zip (#PCDATA)i,
]i,
```

Fig. 2. A DTD of a book

The output of the mapping technique after it is applied over the book DTD :
Based on example 1, we will show some examples of applying our technique:
Example 2: let we have the rule:
 1- we have the relational schema of book DTD by applying

TABLE I
A RELATIONAL SCHEMA FOR A BOOK DTD

book

   1.1.1.3. return sql
1.1.2. else if (t) is a table in the mapping, then
   1.1.2.1. determine the parents and ancestors before the last axis (/) until reach to the first
            element in the xpath query or reach to the axis (//)
   1.1.2.2. sql= "Select   t.att or aggfun(t.att)"+the   elements   or   containing the elements
   determined from step (1.1.2.1))"+
put the predicates in the xpath
   1.1.2.3. return sql
 1.2. else if the last axis (//), then
  1.2.1. if (t) is an attribute in the mapping, then
    1.2.1.1. determine its parents from the DTD
    1.2.1.2. sql="Select   t or aggfun(t)"+
            the elements or containing
            the elements determined from step (1.2.1.1))"+
"[Where   put the predicates in the xpath
  1.2.1.3. return sql

1.2.2. else if (t) is a table in the mapping, then
  1.2.2.1. determine its parents from the DTD
  1.2.2.2. sql="Select t.att or aggfunc(t.tt)"+
According to the book DTD and its corresponding relational
schema which are shown in example 1, we will show some examples of translating XPath queries
over the book DTD to SQL queries over the corresponding relational schema:

the mapping algorithm in [7]:
2- By using the getSQL algorithm to translate the XPath query author/name to SQL query, we
will have the following SQL query:
if the rule has a condition element, a condition attribute will be added to the rule table and add
to the where clause of the nested secure query.

with title "The Selfish Gene" :
Q1: XPath query : /book[booktitle=" the selfish Gene
"] / author/(firstname union lastname). According to getSQL technique:
1- From (step 1) the leaf is firstname union
lastname.
2- Since the last axis is (/), we go to (step 1.1.1).
3- Since firstname and lastname are attributes in the mapping, we perform (step 1.1.1.1)
and (step
1.1.1.2).
4- The output from (step 1.1.1.1) is the author
element and the book element.
5- The author element has a corresponding table called
author, and the book element has a corresponding table called book, so the output of (step
1.1.1.2) is the following SQL query:
        select firstname, lastname
cirripedia"]//editor/@name
(Q2 is a recursive query) According to getSQL technique:
1- From (step 1) the leaf is the attribute name.
2- Since the last axis is (/), we go to (step 1.1.1).
3- Since name is an attribute in the mapping, we perform (step 1.1.1.1) and (step 1.1.1.2).
4- In (step 1.1.1.1), we determine the parents and ancestors until the axis (//), so the output is the
element editor.
5- In the relational schema, the monograph table
contains the editor element, so the output of (step
1.1.1.2) is the following SQL query:
        select editorname
indirectly through monograph:
Q3. xpath query : monograph//name (a recursive query) According to getSQL technique:
1- From (step 1) the leaf is the attribute name.
2- Since the last axis is (//), we go to (step 1.2.1).
3- Since name is an attribute in the mapping, we perform (step 1.2.1.1) and (step 1.2.1.2).
4- In (step 1.2.1.1), we determine the parents of the name attribute from the DTD, so the
output is the element author and the element editor.
5- In the relational schema, the monograph table
contains the editor element and the table author
is corresponding to the author element, so the output

III. RELATED WORK

Very few research works have been done on XML security over relational databases. The proposed techniques in [2], [4], [6], [9], and [13] proposed to secure XML documents in native databases. In [3], the authors have proposed a technique for access control based on an authorization graph. In [10] and [11], the XML documents stored in relational database without using XML schema. The proposed technique in [15] used XML schema in storing XML documents in relational database, but it had performance drawbacks. In [14], the author tried to overcome the drawbacks of [15] by providing an access control model for schema-based storage of XML documents in relational storage and translating XML access control rules to relational access control rules. However, the proposed algorithms had performance drawbacks. In this paper, we will use the same access control model of [14] and try to overcome the drawbacks of [14] by proposing an efficient technique to store the XML access control rules in a relational storage of XML DTD. The proposed technique based on the proposed algorithm in [7] that maps the XML DTD to relational schema. We also propose an algorithm to translate XPath queries to SQL queries based on the mapping algorithm in [7].

IV. CONCLUSION

In this paper, we use the same access control model of [14] and try to overcome the drawbacks of [14] by proposing an efficient technique to store the XML access control rules in a relational storage of XML DTD. The proposed technique based on the proposed algorithm in [7] that maps the XML DTD to relational schema. We also propose an algorithm to translate XPath queries to SQL queries based on the mapping algorithm in [7].

REFERENCES

[1]   R. Abassi, F. Jacquemard, M. Rusinowitch, and S. G. El Fatmi. XML Access Control:from XACML to Annotated Schemas. *In Proceedings of the 2nd International Conference on Communications and Networking (ComNet)*, pages 1–8, 2010.

[2]   E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):290 331, Aug., 2002.

[3]   S. Chang, A. Chebotko, S. Lu, and F. Fotouhi. Graph Matching Based Authorization Model for Effiient Secure XML Querying. *In Proceedings of the AINA Workshops*, pages 473–478, 2007.

[4]   E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Design and Implementation of an Access Control Processor for XML Documents. *Computer Networks*, 33(6):5971, 2000.

[5]   E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Securing XML documents. *In Proceedings of the International Conference on Extending Databas Technology*, pages 121–135, 2000.

[6]   E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. A Fine- Grained Access Control System for XML Documents. *ACM Transac- tions on Information and System Security (TISSEC)*, 5(2):169202, May, 2000.

[7]   A. A. Abd El-Aziz and A. Kannan. Mapping XML DTDs to Relational Schemas. *In Proceedings of the 2nd International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–7, 10-12 Jan., 2012.

[8]   W. Fan, C.Y Chan, and M. Garofalakis. Secure XML quering with security views. *In Proceedings of SIGMOD 2004*, pages 587–598, 2004.

[9]   A. Gabillon and E. Bruno. Regulating Access to XML Documents. *In Proceedings of the Working Conference on Database and Application Security*, July, 2001.

[10] D. Lee, W. Lee, and P. Liu. Supporting XML Security Models using Relational Databases. *A Vision. Lecture Note In Computer Science, Springer-Verlag Berlin Heidelberg*, pages 267–281, Sep., 2003.

[11] B. Luo, D. Lee, and P. Liu. Pragmatic XML Access Control using Off- the-shelf RDBMS. *In Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS)*, pages 55–71, 24-26 Sep., 2007.

[12] S. Mohan, A. Sengupta, Y. Wu, and J. Klinginsmith. Access Control for XML-A Dynamic Query Rewrinting

Approach. *In Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005.

[13] M. Murata, A. Tozawa, and M. Kudo. XML Access Control using Static Analysis. *In Proceedings of the 10th ACM conference on Computer and communications security*, pages 73–84, Oct., 2003.

[14] J. Patel and M. Atay. An Efficient Access Control Model for Schema- Based Relational Storage of XML Documents. *In Proceedings of the 49th Annual ACM Southeast Regional Conference*, pages 97–102, 24-24 March, 2011.

[15] K. L. Tan, M. L. Lee, and Y. Wang. Access Control of XML Documents in Relational Database Systems. *In Proceedings of the International Conference on Internet Computing (IC)*, pages 185–191, 2001.