Using Schemas to Simplify Access Control for XML Documents*

Indrakshi Ray and Marianna Muller

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873
{iray,muller}@cs.colostate.edu

Abstract. Organizations are increasingly using the the eXtensible Markup Language (XML) for document representation and exchange on the Web. To protect an XML document from unauthorized access, authorizations are specified on the XML document itself or on the Document Type Definition (DTD) that defines the type of the XML document. Each XML document or DTD is associated with an XML Access Sheet (XAS) that specifies the authorizations. The DTD not being an XML document complicates the specification and enforcement of authorization policies. To overcome the above mentioned problem, XML Schemas need to be used instead of DTDs. In this paper, we show how XAS DTDs can be specified using XML Schemas and propose an access control architecture that can process XAS authorizations. Enforcement of access control allows users to view only those parts of the documents that they are authorized to view. These parts may not conform to the schema of the original document and hence may not be valid. Towards this end we propose a schema loosening algorithm that generates a schema that will be satisfied by documents satisfying the access control requirements.

1 Introduction

Organizations are increasingly using the world wide web to disseminate and distribute information. Most of this information is specified in XML which is emerging as the de-facto standard language for document representation and exchange over the Web. In order to be processed an XML document must be well-formed (obeys the syntax of XML) and valid (conforms to a proper Document Type Definition (DTD) that defines the particular type of XML document). The information distributed by organizations via the web have different levels of sensitivity: some of this information must be distributed internally, some must be shared with other organizations, and others must be disseminated for public use. Protecting information with different levels of sensitivity is non-trivial. To address this problem, researchers [1–9] have proposed models and mechanisms for controlling access to XML documents.

To protect XML documents, authorization policies may be specified on the XML document or on the associated DTD. If an authorization policy is specified on the DTD,

* This work was funded by AFOSR under Award No. FA9550-04-1-0102.

the policy applies to all XML documents conforming to the DTD. Such authorization policies are specified in a document known as XML Access Sheet (XAS). XAS is an XML document and must be well-formed and valid.

DTDs not being XML documents do not satisfy the well-formedness and validity requirements of XML. As pointed out by Zhang et al. [9], using DTDs to validate XML documents causes a number of problems in the specification and enforcement of authorization policies. First, the authorization policies for XML documents and DTDs cannot be specified in an uniform manner because the structure of the documents differ. Second, an XML parser is not sufficient for interpreting the authorizations on a given XML document. Third, the use of DTD limits interoperability.

In this paper, we propose the use of XML Schemas, instead of DTDs, for validating XML documents. This simplifies the specification and enforcement of authorization policies. We show how authorization templates can be specified in the form of XML Schemas. We propose an access control system that can process any authorization specified as an XAS that conforms to some XML schema. Enforcing access control on an XML document often results in a pruned document; the pruned document contains only those information that the user is authorized to see. This pruned document may not conform to the schema of the original document. We propose an algorithm by which the original schema can be transformed to a loosened schema. This loosened schema will be satisfied by all the documents generated from the original XML document that satisfy the access control requirements.

The rest of the paper is organized as follows. Section 2 shows how XML schemas can be used to specify authorization templates. Section 3 presents an access control system that can process different kinds of authorizations associated with the documents. Section 4 concludes the paper with some pointers to future directions.

2 Specifying Authorization Templates using Schemas

Our approach allows for the specification of different kinds of authorization models. We show how authorization policies adapted from the model proposed by Damiani et al. [6] can be expressed using a schema. Each authorization is of the form *(subject, object, action, sign, type)*, where subject is the entity to whom the authorization is granted or denied, object is either a uniform resource identifier (URI) of the resource or is of the form URI:PE, where PE is a path expression on the tree of document URI, action is the operation being authorized or forbidden, sign is either '+' (denoting allow access) or '-' (denoting forbid access), type is one of {LHD, RDH, L, R, LD, RD, LS, RS} depending on the kind of authorization. Figure 1 shows an authorization template for such a model can be specified as an XML schema.

3 Access Control System

In this section we present an access control system for authorizing XML documents. This architecture is adapted from that proposed in [6]. Since we use XAS Schemas instead of DTD, our architecture is not confined to intra-organizational applications.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="stringtype">
 <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="signvaluetype">
 <xs:restriction base="xs:string">
  <xs:pattern value="+|-"/>
 </xs:restriction>
</xs:simpleType>

<xs:simpleType name="typevaluetype">
 <xs:restriction base="xs:string">
  <xs:enumeration value="LDH"/>
  <xs:enumeration value="RDH"/>
  <xs:enumeration value="L"/>
  <xs:enumeration value="R"/>
  <xs:enumeration value="LD"/>
  <xs:enumeration value="RD"/>
  <xs:enumeration value="LS"/>
  <xs:enumeration value="RS"/>
 </xs:restriction>
</xs:simpleType>

<xs:complexType name="actiontype">
 <cs:attribute name="value" type="stringtype" fixed="read" use="required"/>
</xs:complexType>

<xs:complexType name="signtype">
 <cs:attribute name="value" type="signvaluetype" use="required"/>
</xs:complexType>

<xs:complexType name="typetype">
 <cs:attribute name="value" type="typevaluetype" use="required"/>
</xs:complexType>

<xs:complexType name="authorizationtype">
 <xs:sequence>
  <xs:element name="subject" type="stringtype"/>
  <xs:element name="object" type="stringtype"/>
  <xs:element name="action" type="actiontype"/>
  <xs:element name="sign" type="signtype"/>
  <xs:element name="type" type="typetype"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="set_of_authorizations_type">
 <xs:element name="authorization" type="authorizationtype" minOccurs="1"
   maxOccurs="unbounded"/>
 <xs:attribute name="about" type="stringtype" use="required"/>
</xs:complexType>

<xs:element name="set_of_authorizations" type="set_of_authorizations_type"/>

</xs:schema>
```

Fig. 1. XAS Syntax Specified as a Schema

Figure 2 shows the architecture of such a system. The following reasons motivate the need for implementing such a system on the server side. First, this prevents the client (user) from viewing or processing information that he is not allowed to see or process. Second, this obviates the need for the client browser to provide XML support for translating XML documents to HTML.

This architecture is based on the fact that an XML document is internally represented as an object-oriented document graph according to the Document Object Model (DOM) Level 1 specifi cation. DOM provides an object-oriented Application Programming Interface (API) for HTML and XML documents.

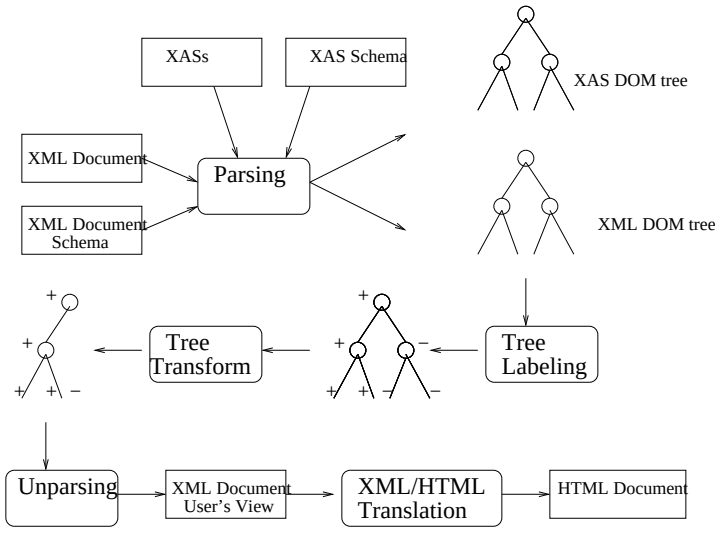

Fig. 2. Architecture of the Access Control System

The following steps are performed by the access control system after receiving a request to access an XML document. The input to the process is the document being requested, the XML Schema against which the document must be validated, the XAS for the document and the Schema, the XAS Schema against which the XAS is validated, and the identity of the requester.

Parsing: The goal of this step is to generate DOM trees for XML and XAS documents. First, the syntax of the XML document is checked with respect to the XML Schema. If the syntax is correct, the XML document is compiled. The compilation results in generating the object-oriented document graph according to the DOM format. The same process is followed for the XAS documents. The output of this step are XML DOM tree and XAS DOM tree(s).

Tree Labeling: The goal of this step is to label the nodes of the XML DOM tree that indicates whether the requester has or does not have access to the node. We follow other researchers and label a node with '+' indicating that the requester has access, a

node with '-' indicating that the requester does not have access. The XAS DOM tree(s) are consulted to determine the access the requester has on the different nodes of the XML document. Note that, in determining the access the entire XAS DOM tree need not be consulted. We can prune the parts of the tree that are not related to the access request. The output of this step is a labeled XML DOM tree.

Tree Transformation: The goal of this step is to generate an XML DOM tree that represents the information the requester is permitted to view. The step proceeds as follows. The label of the tree is consulted and the tree is pruned using a preorder traversal. The output is the pruned XML DOM tree. Note that this document may not satisfy the validity requirements of the original XML schema. For this reason, the XML Schema is transformed into a loosened XML Schema that this new document will satisfy. The generation of loosened schema is defined by algorithm 1. We propose generating the loosened schema off-line and not while the access request is getting processed.

Unparsing: The goal of this step is to convert the pruned XML DOM tree into a text version. The step involves a translation process. The output is a text form of the pruned XML document.

Translation: The goal of this step is to translate the XML document such that users having browsers without XML capability can view the document. This step is a translation of the XML Document to an HTML document that can be viewed by the user.

Algorithm 1 Schema Loosening Algorithm
Input: (i) **A** = $\{a_1, a_2, \ldots, a_n\}$ - the set of access rights obtained from the XASs of the XML Document and the XML Schema for the document. Each element a_i in the set **A** is of the form $(sub_i, obj_i, act_i, sign_i, type_i)$ (ii) obj – the XML Document
Output: XML Schema S'- the loosened schema that specifies the type of all pruned XML documents that satisfy the access control requirements.

Procedure LoosenSchema(**A**, obj)
begin
$S = schema(uri(obj))$ /* get schema associated with XML Document */
$T = CreateTree(S)$ /* function creates tree T from schema S */
for each a_i in **A do**
if $sign_i ==$ '$-$'
$S_i = schema(pe(obj_i))$ /* get sub schema associated with obj_i */
$T_i = CreateTree(S_i)$ /* function creates tree T_i from schema S_i */
$N = \{n | n \in T \wedge n \in T_i\}$ /* set of nodes identified by obj_i
for each $n \in N$ **do**
if n.Attribute().use == "required"
n.Attribute().use = "optional"
if n.minOccurs == "1"
n.minOccurs = "0"
$S' = CreateSchema(T)$ /* Create schema with the updated nodes of tree T */
return S'
end

In this algorithm, we first generate the schema for the XML document and get the corresponding tree. We look at each negative authorization from the set of authoriza-

tions. We identify the object pertaining to this authorization and get the tree corresponding to the object. In this tree, we mark all the required attributes and elements as optional. Repeating this process for all the negative authorizations, we get all the set of attributes and elements of the tree that are optional. Generating the schema from the modified tree gives us the loosened schema.

4 Conclusion

In this paper we propose an access control system that is suitable for XML documents that are validated using XML schemas instead of DTDs. We show how to specify authorization templates in the form of XML schemas, and provide an architecture of an access control system that can process the authorizations specified on XML documents and schemas. Enforcement of access control results in pruning of the document such that the users have the authorization to view this pruned document. The pruned document may not conform to the schema of the original document and hence may not be valid. Towards this end we propose a schema loosening algorithm that generates a schema that will be satisfied by documents satisfying the access control requirements. In future, we plan to investigate how to reduce the time taken for evaluating the authorizations specified on XML documents.

References

1. E. Bertino, S. Castano, and E. Ferrari. On Specifying Security Policies for Web Documents with an XML-based Language. In *Proceedings of the First ACM Symposium on Access Control Models and Technologies*, pages 57–65, May 2001.
2. E. Bertino, S. Castano, and E. Ferrari. Securing XML Documents with Author-χ. *IEEE Internet Computing*, 5:21–151, June 2001.
3. E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web Journal*, 3(3):139–151, May 2001.
4. E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. *ACM Transactions on Information and System Security*, 5(3):290–331, August 2002.
5. E. Damiani, S. Paraboschi, and P. Samarati. A Fine-Grained Access Control System for XML Documents. *ACM Transactions on Information and System Security*, 5(2):169–202, May 2002.
6. E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati. Design and Implementation of Access Control Processor for XML Documents. In *Proceedings of the Ninth International World Wide Web Conference*, May 2000.
7. A. Gabillon and E. Bruno. Regulating Access to XML Documents. In *Proceedings of the Fifteenth IFIP WG 11.3 Working Conference on Data and Applications Security*, Niagara On the Lake, Canada, July 2001.
8. J. P. Yoon. Bitmap-based High-speed Access Control for XML Documents. In *Proceedings of the Seventeenth IFIP WG 11.3 Working Conference on Data and Applications Security*, Estes Park, CO, August 2003.
9. X. Zhang, J. Park, and R. Sandhu. Schema Based XML Security: RBAC Approach. In *Proceedings of the Seventeenth IFIP WG 11.3 Working Conference on Data and Applications Security*, Estes Park, CO, August 2003.