20-7-2005

Delegatable access control for fine-grained XML

Jing Wu
University of Wollongong, jw91@uow.edu.au

Jennifer Seberry
University of Wollongong, jennie@uow.edu.au

Y. Mu
University of Wollongong, ymu@uow.edu.au

Chun Ruan
University of Western Sydney

Delegatable access control for fine-grained XML

Abstract

The access control mechanisms are critical to ensure security in XML (eXtensible Markup Language). Several such mechanisms have been used or proposed; however, the notion of delegation in XML has not been studied in the literature. In this paper, we propose an access control model encapsuling delegation authorization rules for XML documents that allow flexible data granularity and limited inference protection. Our access control policy specification is basically DTD-based. It can also be considered to be document-based.

Disciplines

Physical Sciences and Mathematics

Publication Details

This paper originally appeared as: Wu, J, Seberry, J, Mu, Y & Ruan, C, Delegatable access control for fine-grained XML, 11th International Conference on Parallel and Distributed Systems, 20-22 July 2005, 2, 270-274.

Delegatable Access Control for Fine-Grained XML

Jing Wu , Jennifer Seberry, Yi Mu
School of IT and CS
University of Wollongong, Australia
jw91,jennie,ymu@uow.edu.au

Chun Ruan
School of Computing and IT
University of Western Sydney, Australia
chun@cit.uws.edu.au

Abstract

The access control mechanisms are critical to ensure security in XML (eXtensible Markup Language). Several such mechanisms have been used or proposed; however, the notion of delegation in XML has not been studied in the literature. In this paper, we propose an access control model encapsuling delegation authorization rules for XML documents that allow flexible data granularity and limited inference protection. Our access control policy specification is basically DTD-based. It can also be considered to be document-based.

1. Introduction

XML (eXtensible Markup Language) has recently emerged as the most relevant standardization effort in the area of document representation through markup languages. The XML mark-up technology also plays a central role for developing the semantic web [12]. XML was initially introduced as a data format for documents; therefore, many researchers assumed well-known techniques for securing documents to be straightforwardly applicable to XML data. With authentication, the server will know what information can be sent to the user based on that user's access level, whereas encryption will only let users with decryption keys see the message [7].

An Access control service is needed when some people want to block or allow access to an entire XML instance, while others would like to control access at the tag level. Developing an access control model, and related mechanism, in terms of XML is an important step. In Role-based access control (RBAC), the senior role inherits junior role's permission by virtue of the role hierarchy. But, the junior role is not allowed to carry out the permission, which is only granted to the senior or other role groups [11]. It would be desirable to specify delegation authorizations for the role-based XML environment. With the rapid development of web environments XML data access control has been intensively studied (e.g., [2, 9, 5, 6, 4, 1, 8]). However, early security research work about XML was not directly related to the delegatable access control.

In this paper, we propose an access control model developing delegation authorization rules for XML documents that allows flexible data granularity and limited inference protection. The policy specification is basically DTD-based and it can also be considered as document-based.

The rest of the paper is arranged as follows. In section 2, we review some basics in XML. In section 3, we give the definitions of some basic components for our system and authorization elements. In section 4, we study the delegatable authorization and in section 5, we describe the access control process with delegatable authorization. In the final section, we conclude this paper.

2. XML Basics

In this section, we review some basic concepts in XML.

2.1. XML DOM Tree Structure

The Document Object Model (DOM) is a platform- and language-neutral interface for dynamically accessing and updating the content, structure, and style of XML documents. The DOM provides a standard set of objects for representing documents, a standard model of how these objects can be combined, and a standard interface for accessing and manipulating them. The basic construct of an XML document is the element and the attribute. Elements can be nested at any depth and can contain other elements. Attributes can also be specified for an element. An XML document can be represented by a tree structure. A node of the tree could be different types (elements, attributes). Each element/attribute in XML can be represented as a node in a DOM tree.

Since XML has a tree-shaped hierarchical structure and an XML document is composed of several elements [6]. Each element corresponds to a node in the tree. Access control must be processed using a changed DOM tree.

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE

2.2. Fine-Grained Access Requirement for XML

With access control technology, the access control policies control how an XML document appears to different users. The policies also insure the document is securely updated as specified by the privileged subjects. As we can see, an XML document is structured into elements and attributes. Elements can contain elements. An attribute can also be represented by an element. Protection of XML sources with a large number of documents has various protection requirements. For example, in a hospital environment, information about nurses in a department can be made available to everyone, whereas information about an patient could be kept hidden from most subjects and made visible only to a restricted number of authorized subjects. To ensure a differentiated and appropriate protection of the contents of XML documents, different levels of protection granularity from an attribute to the whole document are required.

2.3. DTD

DTD (Document Type Definitions) is a standardized means of describing XML document type. It is similar to the basic principle of object oriented programming: objects are grouped and described as an object class. DTD can be attached to XML documents, specifying the authorization rules that XML documents may follow. DTD level policies automatically propagate to all DTD instances and policies specified on DTD elements automatically propagate to all associated attributes and links. The following is the example of DTD.

In the above example, the `hospital` can be referred to as the root of the tree. The `elements` following the root denote the nodes of the tree, which describe the information of `patient`.

3. Specification for Authorization Rules

3.1. Subject

A subject is specified by user-set and group-set, where user-set and group-set are a set of role names and group names, respectively. Both users and groups may have hierarchical structures.

```
<! ELEMENT  subjects (users, groups)>
<! ELEMENT  users (numbers*)>
<! ELEMENT  numbers (name?)>
<! ELEMENT  name #PCDATA>
<! ELEMENT  member id ID>
            Idref  IDREF>
<! ELEMENT  groups ((ANY|member)*)>
```

3.2. Xpath and Object

XPath is a language to express a path or selective nodes in a XML tree. An object can be any node of an Xpath tree. In the proposed Access Control Model, XPath models an XML document as a tree of nodes. We define each object as a node of a DOM tree and the objects will be denoted by the path expression XPath. For example the following Xpath expression could locate the name elements of the patient whose blood sugar level is greater than 11.5 and whose name starts with a letter J.

```
//descendant::patient[descendant::
blood_sugar_level>11.5/descendant::
name[starts-with(child::textnode( ),"J"]
```

An object represents a set of elements or a single element in a target XML document. We identify it by a XPath expression, specifying it with an "href" attribute. Authorizations specified for an element are intended to be applicable to all its child non-element nodes such as attribute nodes and text nodes. For example, a read authorization for an element node means that one can read all its child nodes except for sub-elements. The objects have an element-based hierarchical structure.

```
<! ELEMENT  object  EMPTY>
<! Attlist  object href CDATA #REQUIRED>
```

3.3. Authorization Types

In the proposed model, we have the constant set of authorization type T = -, +, $*$. Negative "-" specifies that the access must be forbidden; positive "+" specifies that the access must be granted and the symbol "$*$" is a special character denoting delegatable administrative privilege. It specifies that the access must be delegated as well as granted. ("$*$" equals to "+" plus administrative privilege on the access).

3.4. Privileges (Access rights)

We also regard the privilege as the access right. Our model supports two kinds of privileges: browsing and updating.

Browsing privilege authorizes a subject to view an element or navigate through the link of the element. Updating privileges allow subjects to modify (or delete) the content of an element or to append new information in an element.

3.5. Propagation Options

There are three types of propagation options available in our model:

Type	Semantics
NO_PROP	The security policy applies only to the protection objects defined in its specification; no authorization propagation is performed.
UP_CASCADE	The authorization of an element (object) is propagated to its parent element if there is no authorization for its parent element.
DOWN_CASCADE	The authorization of an element (object) is propagated to its sub -elements if there is no authorization for the sub-elements.

4. Delegatable Authorization Rule

4.1. Delegation Management

The delegation policies allow the grantor of an authorization policy to delegate some or all of their access rights to a new set of subjects (grantees). Effectively, when a grantor performs a delegation action, a new administrator for that access is created. If users want to distribute the administration of rights without further control, they just delegate the rights to others and let them do grant thereafter. Whenever users want to take some control of the rights, their grants will always have higher priorities [3] based on our conflict resolution policy. The delegation is a transference of a access right from one grantor to grantee. After the delegation, the grantee retains the access right. For example, in the hospital environment, a nurse has no permission to perform surgical operations, which was granted to doctors and specialists only. So the nurse could not access the some sensitive information regarding the patient. But, sometimes a nurse may assist surgical operation upon a doctor's request. In this case, the doctor may delegate some privileges to the nurse who was requested to assist the operation and the nurse may get permission to access that portion of information about this patient. In this delegation, the nurse is grantee and the doctor is grantor. With delegatable access control the permissions of accessing XML can be performed temporarily.

4.2. Conflict Resolution

Since we support different types of access rights, conflicting authorizations may arise where a user is granted two different types of access rights. Thus a proper conflict resolution policy is needed. We solve conflicts as follows. First, trace the delegation relation path explicitly. When a conflict occurs, we will see if the two grantors fall into a delegation path. If they do, then let the authorization with the grantor as the predecessor in the path override the other one. In other words, along the delegation path, the predecessors' grants have higher priority than the successors' grants. This policy can support well-controlled delegation. Second, if the conflicts can not be solved by the above policy, we will use Negative-take-precedence based policy to resolve the conflicts. That is, we will resolve the conflicts according to their types, and the priority sequence is $- > + > *$. This policy favors security.

4.3. Definition of the Authorization Rules

In this section, we introduce the definition of our authorization rules, including the general access rule and the delegatable authorization rule.

Definition 1 *(General Access Rule). We define the general access rule as a term of type $S \times O \times T \times A \times P$. S is a set of subjects which can be users, roles or groups. O =* `target` + `path`*(V,E).* `target` *is an XML or DTD.* `path` *is an XPath expression that eventually selects specific portions (object) of the XML document in XML tree where V is a set of nodes. E is a set of edges. T is a set of authorization types. A is the set of access rights. P is the propagation option.*

The general authorization is represented as a 5-tuple of the form:

```
D =<subject,target+path,authorization_type,
            access_right,propagation_type>
```

which denotes subject grant the access right on Xpath specified object with authorization type and propagation type. Here, `subject` $\in S$,

`target` $\in O$,
`authorization_type` $\in \{+, -\}$,
`access_right` $\in$ {`browsing`,`updating`},
`propogation_type` $\in$ {`NO_PROP`, `UP_CASCADE`,`DOWN_CASCADE`}.

In the delegation process, the entity that has been given the access right to delegate by another entity, who has the delegatable access right, can perform valid delegations. The following is the definition of our delegatable rule.

Definition 2 *(Delegatable Rule). The delegatable authorization is a term: $S \times O \times T \times A \times S \times P$ rule, where S is a set of subjects which would be grantor or grantee; O =* `target` + `path`*(V, E),* `target` *is an XML or DTD, path is an XPath expression that eventually selects specific portions (object) of the XML document in XML tree where V is a set of nodes and E is a set of edges; T is an authorization type; A is the set of access rights, and P is the propagation option.*

The delegatable authorization is represented as a 6-tuple of the form:

```
D =< grantee,target+path,authorization_type,
   access_right,grantor,propagation_type>
```

which reads that grantee is granted by grantor the access right on Xpath specified object with authorization type and propagation type. Here, grantor $\in S$,

grantee $\in S$,
target $\in O$,
authorization_type $\in \{+, -, *\}$,
access_right $\in$ {browsing, updating},
propogation_type $\in$ {NO_PROP, UP_CASCADE, DOWN_CASCADE}.

The structure of the authorization rules can be specified at the level of a DTD schema or document instance level. The following is an example of the policies for a XML documents using the delegatable authorization rules:

```
<policy>
    <access_rule subject="nurses"
        target = "patient_information.xml"
        path="//patient/personnel_information"
        authorization_type = "-"
        access_right = "updating"
        propagation_type = "NO_PROP"
    />

    <delegatable_rule
        grantee="//nurse[department="surgery"]"
        target = "patient_information.xml"
        path = "//patient[blood_sugar_level>
        11.5]/name[starts-with "J"] "
        authorization_type = "*"
        access_right = "browsing"
        grantor = "doctor"
        propagation_type = "DOWN_CASCADE"
    />
</policy>
```

The first rule says that the nurses are forbidden to update the content of the patients personnel information. The second rule says that the nurse in the department of surgery was delegated the permission of browsing by the doctor to access the information on the patient whose blood sugar level is greater than 11.5 and the name begins with "J". The propagation type DOWN_CASCADE means the authorization is propagated to all sub-elements of the object.

4.4. Define the structure of authorization rules on DTD level

As we described in the earlier chapter, a DTD defines the structure of a set of XML documents and the authorizations (at different granularity levels) that can be specified at the DTD level and propagated to all documents. We could define the structure of authorization rules on DTD as following,

```
<! ELEMENT delegation_rules EMPTY>
  <! ATTLIST  grantee (#PCDATA)>
  <! ATTLIST  object (#PCDATA)       >
  <! ATTLIST  authorization_type( + | - | * )
                          #REQUIRED>
  <! ATTLIST  access_right(browsing|updating)
                          #REQUIRED>
  <! ATTLIST  grantor (#PCDATA)>
  <! ATTLIST  propagation_type (NO_PROP,
       UP_CASCADE,DOWN_CASCADE) #REQUIRED>
```

The delegation rule is the element with 6 attributes which are grantee, object, authorization type, access right, grantor and propagation type. It also lists the types of attributes being available or declared.

5. Delegatable Access Control Process

Access control processes are determined based on information about access rights, security policies, authorization rules, and other factors depending on the system environment. Through access control, the system can restrict unauthorized users' access to resources in the system and guarantees the confidentiality and integrity of the resources. Our delegatable access control process has to complete the following steps. The concept of the model is illustrated in the Figure 2.

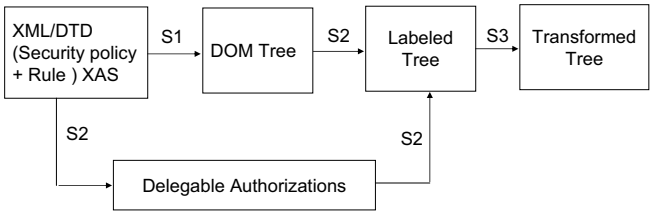

Figure 1. XML delegatable access control process.

In the process, the step S1 is the parsing step which consists of the syntax check of the requested document with respect to the associated DTD and its compilation to obtain an object-oriented document graph according to the DOM format. Step S2 is Tree labelling and authorization checking. This step involves the propagation of the labelling of the DOM tree according to the authorizations listed in the XAS (XML Access Sheet) associated with the document and its DTD, both at the organization and at site level. The authorizations relevant to the user are analyzed and applied to the

nodes. S3 is the step of transformation and unparsing. The transformation phase is a pruning of the DOM tree according to its labelling.

The example in Figure 3 shows how a tree is pruning. With access control specification, the read accessibility view is implied by the labeled XML tree (T1). Nodes O_4 and O_8 are forbidden to have view accessibility with respect to specific user U_1. After transformation, the pruned tree(T2) only remains the nodes with view privileges. This pruning preserves the validity of the document with respect to the loosened version of its original DTD.

By unparsing, the resulting XML document together with the loosened DTD, can then be transmitted to the user who requested access to the document.

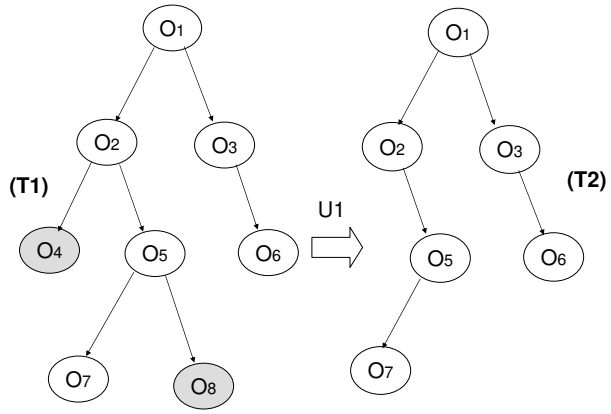

Figure 2. DOM tree pruning transformation.

The process sets all elements and attributes in DTD to be optional so that the existing DTD be preserved even if nodes are removed from the DOM tree. The user who has been authorized through such a process views only that part of the XML document, for which he/she has been given the right to access, and the user who has not been authorized cannot access the other part. This is the way how data security (confidentiality) is guaranteed.

6. Conclusion

XML Access Control aims at providing XML documents with a sophisticated access control model and fine-grained access control specification language. Access control is a means to allow or deny subjects to carry out operations on objects in the computer system [2, 10]. In XML documents, users have to share resources and communicate with each other to perform their jobs more efficiently. There is a strong demand for the access control of distributed-shared resources. For better performance, it is important to keep resources and the information integrity from an unexpected user on fine-grained level. In this paper, our access control model for XML is described allowing for definition and enforcement of access restrictions directly on the structure and content of XML documents, thus providing a simple and effective way for users to protect information at the same granularity level provided by the language itself. Our model also supports authorization propagations and authorization delegations.

References

[1] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM Transaction on Information and System*, 5(3):290–331, 2002.

[2] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security*. Addision-Wesley, 1995.

[3] Y. Z. Chun Ruan, Vijay Varadharajan. Logic-based reasoning on delegatable authorizations. In *Proceedings of the Syposium on Methodologies for Intelligent Systems*, pages 185–193, Lyon, France, 2002.

[4] E. Damiani, S. D. C. di Vimercati, E. Fernndez-Medina, and P. Samarati. An access control system for svgdocuments. In *Proceedings of the Sixteenth Annual IFIPWG 11.3 Working Conference on Data and Application Security*, pages 121–135, 2000.

[5] E. Damiani, S. Vimercati, S. Paraboaschi, and P.Samarati. Design and implementation of an access processor for XML documents. In *Proceedings of the 9th international WWW conference*, pages 59–75, 2000.

[6] E. Damiani, S. Vimercati, S. Paraboaschi, and P.Samarati. Securing XML document. In *Proceedings of International Conference on Extending Database Technology (EDBT2000)*, pages 121–135, 2000.

[7] B. Dournaee. *XML Security*. McGraw-Hill, New York, 2002.

[8] M. Kudo and S. Hada. XML document security based on provisional authorizations. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000.

[9] C.-H. Lim, S. Park, and S. H. Son. Access control of XML documents considering update operations. In *Proceedings of ACM Workshop on XML Security*, pages 49–59, 2003.

[10] T. F. Lunt. *Access Control Policies for Database Systems*. Elsevier, North-Holland, Amsterdam, 1989.

[11] M. Sloman. Policy driven management for distributed system. *Journal of Network and System Management*, 2(4), 1994.

[12] J. Wu, J. Seberry, and Y. Mu. Security mark-up for semantic web. In *Proceedings of the 15th Australasian Workshop on Combinatorial Algorithms*, pages 206–212, 2004.