XML ACCESS CONTROL FOR SECURITY AND MEMORY MANAGEMENT

Sun-Moon Jo[1], Chang-Mo Yang[2], Weon-Hee Yoo[1]

[1] Department of Computer Science and Information Engineering,
Inha University
253 YongHyun Dong, Nam Ku, Incheon, Korea
sunmoonpink@hanmail.net, whyoo@inha.ac.kr
Department of Computer Education
[2] Sugok-dong, Heung duk-gu, cheongju, Chungbuk, Korea
cmyang@cje.ac.kr

Abstract. Since XML was presented as a standard data type on the web, many data have been made and transformed into the XML type, consequently generating a large amount of XML data. Therefore, the need for efficient management and security of large-capacity XML data is gradually becoming important. The existing access control has problems that DOM trees should be loaded on memory in the process of parsing all XML documents to generate DOM trees, that a large amount of memory is used to search for trees repetitively to set access authorization on all nodes of DOM trees, and that the system becomes inefficient due to complicated authorization assessment. In this paper, we suggest an access control policy model and tree labeling algorithm for secure XML documents. So it can reduce expenses of authorization assessment of the existing access control implemented in a complicated and repetitive way.

1 Introduction

After XML (eXtensible Markup Language) was presented as a standard for data exchange and representation on Internet, many new data have been made in the XML type and the existing data have been transformed into the XML type; consequently, the amount of XML data is increasing drastically [10]. XML can use its merit of describing meaningful information immediately to provide a standard data type in the form of exchanging information on a lot of data generated in the process of companies' database or applied program operation. It is therefore very appropriate for a component label and document management system that needs definition and description of detailed information and its meaning. As a large amount of XML-type information was provided on web environment, developers and users became more concerned about the issue of XML document security.

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD)" (R05-2004-000-11694-0)

As for researches in XML document security and relevant products, control of access to information in web environment and transmission layer security protocols including electronic signature and coding is mostly related to HTML documents, and couldn't deal with access control according to the meaning of partial information, which is the main advantage of XML as file-based access control. So access control applying the advantage of XML became necessary [5]. The existing access control first parses XML documents to get DOM trees if a user demands XML documents. After the parsing, it sets the sign value that means permission (+) or denial (-) of access to nodes of DOM trees in reference to authorization of relevant database and XML documents. Nodes with the sign value set at - in DOM trees are removed and only those with the value set at + are shown to the user in the XML type [3], [4], [5].

However, it has problems that DOM trees should be loaded on memory, that a large amount of memory is used to search for trees repetitively to set access authorization on all nodes of DOM trees, and that the system becomes inefficient due to complicated authorization assessment.

In this paper, we suggest an access control policy model and tree labeling algorithm for secure XML documents. It is therefore possible to make it easy to manage information on access authorization and users and remove unnecessary parsing and DOM tree searching, consequently obtaining better efficiency than the existing access control model.

The paper is organized as follows. Section 2 examines studies and problems about XML access control. Section 3 defines the concept of XACML and an action label type group(ALTG) for access control policy models to describe tree labeling algorithm. Section 4 evaluates the access control policy models and section 5 draws a conclusion and describes the future course of studies.

2 DOM-Based XML Access Control

An XML DOM tree provides API (Application Program Interface) to access elements of XML documents [2]. The existing access control models [1], [3], [4] uses such a DOM tree to set access authorization to elements of DTD and XML documents and control users' access to XML data according to information on access authorization set.

According to the process of changes in documents in Figure 1, there is a request for seeing XML documents. As for all XML documents and DTD concerned, information on access authorization is specified in documents called XAS (XML Access Sheet). XML documents are parsed to obtain DOM trees; then, a value of sign is set which means admission (+) or rejection (-) of access to nodes of DOM trees based on XAS of DTD and XML documents. It is called labeling to set authorization to nodes of DOM trees. The nodes with the value of sign set as - are removed from the labeled DOM trees and only those with the value set as + are restored to the user [1], [3], [4], [5]. Here, although XML documents with nodes removed from DOM trees can fail to be valid (its solution requires the loosening process, with all elements and attributes set as optional in DTD), they can maintain the existing DTD despite the removal of nodes from DOM trees.

To solve the problem that XML documents with nodes removed from DOM trees can fail to be valid for DTD, the loosening technique is suggested to maintain the existing DTD despite the removal of nodes from DOM trees [3]. However, this method causes a semantic conflict due to the loss of information on the structure. The tree labeling technique is used to maintain information on the structure of documents, which has a problem that it can violate secrecy by showing the existence of data and information on the structure with rejection (-) labeling [3], [4]. Although it applies a strong labeling technique to prohibit access to nodes with rejection (-) labeling [1], this technique has limitations in usability of data by prohibiting access to sub-nodes of prohibited nodes.

Above-mentioned studies have a problem of reducing the efficiency of system as the entire DOM trees should be loaded in memory and much memory is used due to repetitive tree retrieval to set access authorization to all nodes in DOM trees.

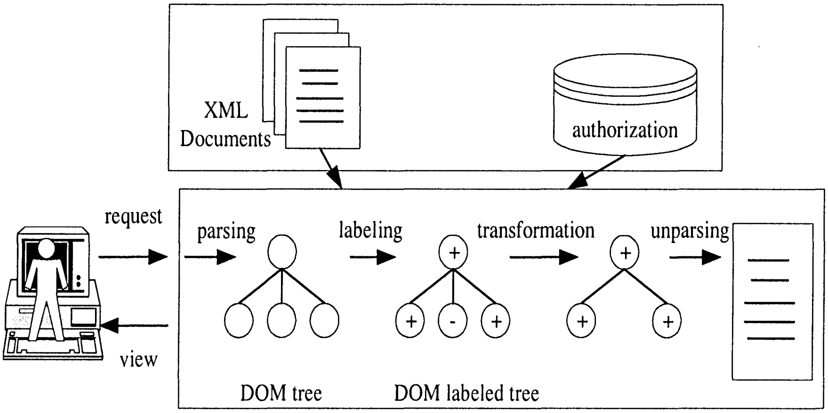

Fig. 1. XML Document Access Control Processor

3 Access Control Model for XML Documents

In this section, we discuss access control policies and tree labeling algorithm technique for securing XML documents.

3.1 XACML Concept

XACML(eXtensible Access Control Markup Language) is an international standard on access control [6], which is composed of policy language described by XML and access control decision request and response language. XML-based access control is composed of XML vocabularies to express rules on authorization. It provides minute access control services for resources requiring security by using XML vocabularies to define access control rules. An access control policy is to determine who can carry out which operation on which resource.

Policy language, which describes general requirements for access control on rules, policies, and policy set, also defines functions, data types, combining logic, and so on. Request language serves to construct questions about which object can perform a specific action for a particular resource; response language is used to express results of the request, with responses indicated in four results: permit, deny, indeterminate, and not applicable [8], [9].

3.2 Requirements for XML Access Control System

The existing web-based access control models can describe authorization in a unit or part of files. However, this method fails to make access based on a meaning of information in order to deal with information by the meaning, which characterizes XML documents most remarkably, or access to such small units as elements. Therefore, requirements for access control to XML documents can be summarized as follows [5]:

① Authorization should be provided in many structural levels.

② Extension to existing Web server technology. XML documents are usually made available by means of Web sites, using a variety of HTTP-based protocols.

③ It is necessary to support fined-grained access control. The access control model should provide access control in many levels such as document set or one element.

④ It is necessary to secure transparency. If it should be transparent to a user to conduct operations of an access control system, it should be impossible to know which part is provided with no authority in a document a requester looks at.

⑤ Smoothness integration with existing technologies for user authentication (e. g. digital signatures). Access control should complement tag level authentication based on digital signatures.

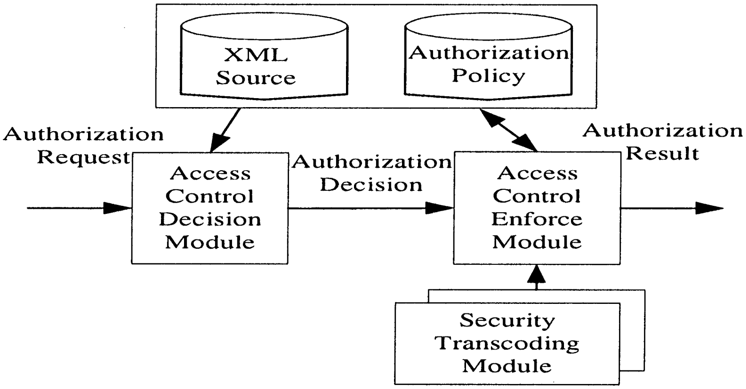

Fig. 2. Access Control Architecture for XML Documents

The Figure 2 shows the structure of an access control model for XML documents. A user requests access to resources in a system; the system determines whether to admit or reject by referring to information on access control policy and XML documents requested by ACM (Access Control Module) after confirming that the requester is a legitimate user. This determination is transmitted to ACM (Access Control Module); in case of a request for operation and reading of XML documents concerned, the documents are manufactured only with information which a user is authorized to read and then transmitted to the user [7].

3.3 Authorization Subjects and Authorization Objects

In our model, a subject is a user. It is not the purpose of this paper to give detailed information on how these subjects are organized. Each user has an identifier which can be the login name. Each user is the member of one or several user groups. For the sake of simplicity we assume that groups are disjoint but they can be nested.

An object is Resource pointer of target object such as a directory path expression. Since we deal with XML documents in this paper, we use an XPath [11] tree to specify the target objects.

3.4 Access Authorizations Policy Rules

XML document access authorizations are composed of subject, object, action, action label type group, sign, and type:

- Subject: User name, IP address, computer name (A subject who accesses XML documents and provides user group and pattern);
- Object: XPath 1.0 (An element of XML documents, which is expressed in XPath);
- Action: Read, write, create, delete (An operation the subject can implement);
- Action Label Type Group: R(read operator group), DSLG(Document and Structure Label Group; operator group);

- Sign: {+, -} is the sign of the authorization, which can be positive (allow access) or negative (forbid access);
- Type: {L, R, LDH, RDH, LS, RS, LD, RD} is the type of the authorization (Local, Recursive, Local DTD Hard, Recursive DTD Hard, Local Soft, Recursive Soft, Local DTD, and Recursive DTD, respectively).

The subject of authorization can be described as id or the access-requested position. The object means a resource to protect access. XPath language, which is a W3C standard of path expression, or expanded Uniform Resource Identifier (URI), is used to express the object [11]. Path expression is a list of pre-defined functions or element names differentiated by a divider (/) on the structure of document tree. Action refers to an operation the subject can implement; according to how much action label type group(ALTG) operators change XML, the operator group can be classified as follows:

- Read Label Group: A set of operators that read but never change documents in XML (Read).
- Document Structure Label Group: A set of operators that change XML documents and structures (Insert, Delete, Rename).

If a new operator is added to an access control model, information on access authorization becomes complicated because the operator's information should also be included in the information on access authorization. And labeling and DTD verification processes reduce the efficiency of the system due to repetitive DOM tree retrieval and parsing.

3.5 Propagation Policy Rule

A Propagation policy rule is a security policy to use for regulating authorization conflicts to set access authorization. As for authorization interpretation, the final sign (+ or -) is determined by reflecting propagation and overriding in each element. If there are both permission and denial for the same subject, only one access authorization is determined according to the conflict settlement principle. The following steps are rules to determine precedence of authorization in case of authorization conflicts.

Rule 1: Authorization on the most specific subject described according to partial order of subjects takes precedence.

Rule 2: Directly described authorization rather than that occurring by transmission takes precedence.

Rule 3: Authorization directly described on XML documents rather than that described on DTD takes precedence.

Rule 4: Authorization on nodes rather than that of its forefather takes precedence.

3.6 Default Policy

When there is no permission(grant or deny) for an access request or when the confliction resolution policy “nothing takes precedence” is enforced, we need to make a decision according to the specified default policy. This can be specified in the <default> element for each action The default policy is “deny” by default for every action.

3.7 Access Control Technique

Labeling is the process of using information on access authorization defined by a security manager to set access authorization to nodes of DOM trees requested by a user. The information on labeled authorization is used in determining whether to admit or reject the user's request. To label information on authorization to DOM trees based on an operator, it was necessary to repeat the labeling process as many times as the number of kinds of operators included in a question. Suggested is labeling algorithm based on the ALTG to remove such a repetitive labeling process.

■ Document Tree Labeling Algorithm ■

```
Input : A requester rq and an XML document URI

Output : The view of the requester rq on the document URI

Method : /* L is local, R is recursive, LDH is Local DTD Hard, RDH is Recursive DTD Hard, LS is Local Soft, RS is Recursive Soft, LD is local DTD-level, RD is recursive DTD-level */

1. A.xml A = {a= <subject, object, action, ALTG, sign, type> | a ∈ authorization, rq ≤ AS subject, uri(object)= =URI OR uri(object) ==(URI)}

2. Let r be the root of the tree T corresponding to the document URI, n is a node other than r, p is the parent node of n

3. AM( ) : returns the ALTG of a node specified in the authorization rule,

4. Type() : returns the type specified in the authorization rule,

5. Propagation_rule() : returns the ALTG determined by propagation rules

6. For each c ∈ children(r) do label(c, r)

7. For each c ∈ children(r) do prune(T, c)

8. L1r = AM(r) in A.dtd ,  L2r = AM(r) in A.xml

9. initial_label(r)

10. For each c ∈ children(r) do label(n,p)
```

```
Procedure initial_label(n)

     if  L1r ∪ L2r ={ }, Lr=default(r)

     else  Lr = propagation_rule([L1r, L2r])

Procedure label(n,p)

if type (p) in [L, R, LDH, RDH, LS, RS, LD, RD]

       if  L1n & L2n = { }, Ln = Lp

       else Ln = propagation_rule([Lp, L1n, L2n])

else

     if  L1n & L2n = { }, Ln = default(n)

     else  Ln = propagation_rule([L1n, L2n])

Procedure prune(T, n)

   /* Tree representing the document, Determines if n
has to be removed from T */

For each c ∈ children(n) do prune(T, c)

if children(n) = { } and Ln ≠ '+'  then

   remove the current node from T
```

Existing XML access control techniques determine whether to allow a query to access or not after labeling the DOM tree. Thus the system has to keep all information necessary for right tests to the end unnecessary right tests were repeated [5]. Such extra tasks slow down the speed of access control.

4 Evaluation

```
<!DOCTYPE authorizations[
<!ELEMENT set of authorizations (authorization)+>
<!ELEMENT authorization (subject, object, ALTG, action, sign, type)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT object (#PCDATA)>
<!ELEMENT ALTG empty>
<!ELEMENT action empty>
<!ELEMENT sign empty>
<!ELEMENT type empty>
<!ATTLIST set of authorizations about CDATA #REQUIRED>
<!ATTLIST ALTG value(R, DSLG) #REQUIRED>
<!ATTLIST action value (read, write, create, delete) #REQUIRED>
<!ATTLIST sign value (+ | – ) #REQUIRED>
<!ATTLIST type value (L|R|LDH|RDH|LS|RS|LD|RD) #REQUIRED> ]>
```

Fig. 3. XML Access Sheet base DTD

Our processor takes as input a valid XML document requested by the user, together with its XML Access Sheet (XAS) listing the associated access authorizations at the instance level. The processor operation also involves the documents DTD. The processor output is a valid XML document including only the information the user is allowed to access. To provide a uniform representation of XASs and other XML-based information, the syntax of XASs is given by the XML DTD depicted in Figure 3.

The existing access control is the repetitive tree labeling process and DTD verification process consume a lot of memory for XML parsing and DOM tree search, which may degrade system performance.

Table 1. XML Documents Transformations by the security processes

Execution Processes	
Request	1) XML documents request
Security Processor	2) Parsing(DOM tree) 3) Tree Labeling 3) DTD Validation 5) Check for right to change DTD structure 6) Change DTD 7) Change documents structure 8) Unparsing
View	9) XML documents result

In the Table 1 above, if the existing access control technique is used as in the case that a user's authorization changes XML documents and structures, the following procedure is necessary [3], [5].

Step 1: User sends an access request.
Step 2: Parsing of XML documents to examine an operator's authorization.
Step 3: Labeling of authorization to DOM trees using information on access authorization.
Step 4: Determining if structure is changed in the stage of testing DTD.
Step 5: Conducting exchange operation if the DTD test identifies that operation leads to no structure change.
Step 6: Parsing to obtain new DOM trees as XML contents were changed after the operation.
Step 7: Testing authorization of insertion operation.
Step 8: Labeling authorization to DOM trees and testing DTD.
Step 9: An insertion operator is denied because it was shown to change DTD.
As seen above, the existing access control can make the system inefficient with the labeling process to assess authorization after each demand by a user and repetitive visits to DOM trees.

To the contrary, the suggested access control policy model can separate operators' collection into ALTG and thus prevent delay in complicated authorization assessment and responding.

5 Conclusion

In this paper, we suggested an access control policy model and tree labeling algorithm for XML documents. Action label group was defined to solve problems in efficiency while adding operators to the model during access. The existing access control generated XML documents into DOM trees according to a user's demand, identified the XML access control list, and removed nodes with access denied and provided only those with access permitted to a user.

However, the definition of the ALTG made it easy to manage information on access authorization and users and remove unnecessary parsing and DOM tree searching, consequently providing rapid access control. It has a disadvantage of making the system inefficient through the labeling process to assess authorization after each demand by a user and repetitive visits to DOM trees.

Further studies are necessary on access control that reflects each property in other applications using XML type.

References

1. A. Gabillon and E. Bruno, "Regulating Access to XML Documents", In Proc. IFIP WG11.3 Working Conference on Database Security, 2001
2. Document Object Model(DOM), Avaiable at http://www.w3.org/DOM/
3. E. Bertino, S. Castano. E. Ferrari, M. Mesiti, "Specifying and Enforcing Access Control Policies for XML Document Sources", WWW Journal, Baltzer Science Publishers, Vol.3, N.3, 2000.

4. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "Securing XML documents," in Proc. Of the 2000 International Conference on Extending Database Technology(EDBT2000), Konstanz, Germany, March 27-31, 2000
5. E. Damiani, S. Vimercati, S. Paraboschi, and P. Samarati, "Design and implementation of an access control processor for xml documents". In proceedings of the 9th International WWW Conference, Amsterdam, May 2000.
6. IBM Tokyo Lab, "XML Access Control Language", 2000, Http://www.tr.ibm.com/projects/xml/xacl/xaclpec.html
7. Michiharu Kudo. Satoshi Hada "XML Document Security based on Provisional Authorization" CSS 2000, Athens, Greece
8. OASIS-XACMLTC, "OASIS eXtensible Access Control Markup Language", Working Draft 15, 12 July 2002, http://www.oasisopen.org/ommittess/xacml/repository/draft-xacml-schema-policy-15.doc
9. Sun's XACML Implementation. http://sunxacml. soureefore.net/.
10. T. Bray et al. “Extensible Markup Language(XML) 1.0”. World Wide Web Consortium(W3C). http://www.w3c.org/TR/REC-xml(October 2000).
11. World Wide Web Consortium(W3C), “XML Path Language(XPath) Version 1.0”, http://www.w3.org/TR/PR-XPath 19991008, (October 1999).