X

XML Access Control

Dongwon Lee[1], Ting Yu[2]
[1]College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA, USA
[2]Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Definition

XML access control refers to the practice of limiting access to (parts of) XML data to only authorized users. Similar to access control over other types of data and resources, XML access control is centered around two key problems: (i) the development of formal models for the specification of access control policies over XML data; and (ii) techniques for efficient enforcement of access control policies over XML data.

Historical Background

Access control is one of the fundamental security mechanisms in information systems. It concerns with who can access which information under what circumstances. The need of access control arises naturally when a multi-user system offers selective access to shared information. As one of the oldest problems in security, access control has been studied extensively in a variety of contexts, including operating systems, databases, and computer networks.

The most influential policy models today are discretional access control (DAC), mandatory access control (MAC), and role-based access control (RBAC) models. In DAC, the owner of an object (e.g., a file or database table) solely determines which subjects can access that object, and whether such privileges can be further delegated to other subjects. In MAC, whether a subject can access an object or not is determined by their security classes, not by the owner of the object. In RBAC, privileges are associated with roles. Users are assigned to roles, and thus can only exercise access privileges characterized by their roles.

Typical implementations of access control are in the form of access control lists (ACLs) and capabilities. In ACLs, a system maintains for each object a list of subjects who have access to that object. In capabilities, each subject is associated with a list that indicates those objects to which it has access. ACLs and capabilities are suitable to enforce coarse-grained access control over objects with simple structures (e.g., file systems or table-level access control in relational databases). They are often not efficient for fine-grained access control over objects with complex structures (e.g., element-level access control in XML, or row-level and cell-level access in relational databases).

Scientific Fundamentals

XML access control is fine-grained in nature. Instead of controlling access to the whole XML database or document, it is often required to limit a user's access to some substructures of an XML document (e.g., some subtrees or some individual elements).

An XML access control policy is typically modeled as a set of access control rules. Each rule is a 5-tuple (*subject*, *object*, *action*, *sign*, *type*), where (i) *subject* defines a set of subjects; (ii) *object* defines a set of elements or attributes of an XML document; (iii) *action* denotes the actions that can be performed on XML (e.g., read, write, and update); (iv) $sign \in \{+, -\}$ indicates whether this rule is a grant rule or a deny rule; and (v) $type \in \{LC, RC\}$ refers to either *local check* or *global check*. Intuitively, an access control rule specifies that subjects in *subject* can (when $sign=+$) or cannot (when $sign=-$) perform action specified by *action* on those elements or attributes in *objects*. When $type=RC$, this authorization decision also applies to the descendants of those in *object*.

Object is usually specified using some XML query languages such as XPath expressions. There are

multiple ways to identify *subject*. Following RBAC, *subject* can be specified as one or several roles (e.g., student and faculty) [1]. It may also follow *attribute-based access control*, where each user features a set of attributes and *subject* is defined based on the values of those attributes (e.g., those subjects whose age is over 21). Some access control policy in the literature also follows MAC, where *subject* refers to security classes.

Example 1. Consider two access control rules for the subject of "admin" as follows:

R_1: (*admin, /people/person/name, read, −, LC*)
R_2: (*admin, /people//address//*, read/update, +, RC*)

R_1 Indicates that users belonging to the admin role cannot read textual and attribute data of XML node <name>, the child of <person> that is the child of the root node <people>. On the other hand, R_2 specifies that the same users of the admin role can read and even update any XML data that are descendents of XML node if they are descendents of <address> under <people>.

Once an access control policy is specified, there are two problems that need to be addressed. First, given any access request, one needs to determine whether or not there exists a rule that applies to the request. If so, the policy is said to be *complete*. Most access control systems adopt a closed world assumption. That is, if no rules apply to a request, the request is denied. Second, multiple rules with different authorization decisions may apply to a request. One typical ways to resolve such conflicts is to let denial override permit. For XML access control, it may also be solved by having more explicit rules override less explicit ones. For instance, an LC rule may override an RC rule. For two RC rules, the one that applies to a node's nearest ancestor usually dominates.

Languages for specifying access control policy are proposed in such efforts as XACL by IBM. Therefore, it is also possible to use such more expressive languages to specify access control policy within XML access control models. Finally, the use of authorization priorities with propagation and overriding are related to similar techniques studied in object-oriented databases.

Once XML access control is specified in a given model, it can be enforced in a variety of ways. By and large, most of the existing XML access control methods are either *view-based* or relying on the XML engine to enforce access control at the node-level of XML trees. The idea of view-based enforcement (e.g., [2,3]) is to create and maintain a separate view for each user (or role) who is authorized to access a specific portion of an XML data [1]. The view contains exactly the set of data nodes that the user is authorized to access. After views are constructed, during run time, users can simply run their queries against the views without worrying about access control issues. Although views can be prepared offline, in general, view-based enforcement schemes suffer from high maintenance and storage costs, especially for a large number of roles: (i) a virtual or physical view is created and stored for each role; (ii) whenever a user prompts *update* operation on the data, all views that contain the corresponding data needs to be synchronized. To tackle this problem, people proposed a method using compressed XML views to support access controls [3]. The basic idea is based on the observation of accessibility locality, i.e., elements close to each other tend to have the same accessibility for a subject. Based on this observation, a compressed accessibility map only maintains some "crucial" nodes in the view. With simple annotation on those crucial nodes, other nodes' accessibility can be efficiently inferred instead of explicitly stored. Each node in the compressed view is associated with a label (*desc*, *self*), where *desc* can be either d+ or d−, indicating whether its descendants are accessible or not, and *self* can be either s+ or s−, indicating whether the node itself is accessible. Given any node in an XML tree, by its relationship to those labeled nodes in the compressed view, we can infer its accessibility.

Example 2. Consider the XML tree in (i) with squares and circles denoting accessible and inaccessible nodes, respectively. The corresponding compressed view is shown in (ii). Note that since node C is a descendant of B and B is labeled (d−, s+), C can be inferred to be inaccessible.

In the non view-based XML access control techniques (e.g., [4–6]), an XML query is pre-classified against the given model to be "entirely" authorized, "entirely" prohibited, or "partially" authorized before being submitted to an XML engine. Therefore, without using pre-built views, those entirely authorized or prohibited queries can be quickly processed. Furthermore,

those XML queries that are partially authorized are re-written using state machines such that they request for only data that are granted to the users or roles.

Example 3. Consider three access control rules for the security role "admin." Furthermore, an administrator "Bob" requested three queries, Q_1 to Q_3 in XPath as follows:

R_1: (admin, /people/person/name, read, −, LC)
R_2: (admin, /people//address//, read/update, +, RC)*
R_3: (admin, /regions/namerica/item/name, read, +, LC)
Q_1: /people/person/address/street
Q_2:/people/person/creditcard
*Q_3:/regions//**

Then, Q_1 by Bob can be entirely authorized by both R_1 and R_2, but entirely denied by R_3. Similarly, Q_2 is entirely authorized by R_1, entirely denied by both R_2 and R_3. Finally, Q_3 is entirely accepted by R_1, entirely denied by R_2, and partially authorized by R_3 and needs to be re-written to /regions/namerica/item/name in order not to be conflicted with R_3.

Key Applications

As XML has been increasingly used not only as a data exchange format but as a data storage format, the problem of controlling selective access over XML is indispensable to data security. Therefore, XML access control issues have been tightly associated with secure query processing techniques in relational and XML databases (e.g., [2,7]). The access control policy model of XML can also be extended to express security requirements of other semi-structured data such as LDAP and object-oriented databases. The aforementioned access control enforcement techniques can be further extended to protect privacy during the exchange of XML data in distributed information sharing system. For instance, in PPIB system [8], XML access control is used to hide what query content is or where data objects are located, etc. In an environment where XML data are stored in a distributed fashion and users may ask sensitive queries whose privacy must be kept to is utmost extent (e.g., HIV related queries in health information network), XML access control techniques can be used, along with XML content-based routing techniques [9].

Cross-references

► Relational Access Control, ► Secure XML Query Processing

Recommended Reading

1. Damiani E., Vimercati S., Paraboschi S., and Samarati P. A fine-grained Access Control System for XML Documents. ACM Trans. Inform. Syst. Secur., 5(2):169–202, 2002.
2. Fan W., Chan C.-Y., and Garofalakis M. Secure XML querying with security views. ACM SIGMOD, New York, NY, 2004, pp. 587–598.
3. Yu T., Srivastava D., Lakshmanan L.V.S., and Jagadish H.V. A compressed accessibility map for XML. ACM Trans. Database Syst., 29(2):363–402, 2004.
4. Bouganim L., Ngoc F.D., and Pucheral P. Client-based access control management for XML documents. In Proceedings of the International Conference on Very Large Data Bases (VLDB), Toronto, Canada, 2004.
5. Luo B., Lee D., Lee W.C., and Liu P. QFilter: fine-grained run-time XML access control via NFA-based query rewriting. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), Washington DC, USA, 2004.
6. Murata M., Tozawa A., and Kudo M. XML access control using static analysis. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Washington DC, 2003.
7. Cho S., Amer-Yahia S., Lakshmanan L.V.S., and Srivastava D. Optimizing the secure evaluation of twig queries. In Proceedings of the International Conference on Very Large Data Bases (VLDB), Hong Kong, 2002.
8. Li F., Luo B., Liu P., Lee D., and Chu C.H. Automaton segmentation: a new approach to preserve privacy in XML information brokering. In Proceedings of the ACM Conference on Computer and Communication Security (CCS), Alexandria, VA, 2007.
9. Koudas N., Rabinovich M., Srivastava D., and Yu T. Routing XML queries. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), Boston, MA, 2004.
10. Bertino E. and Ferrari E. Secure and selective dissemination of XML documents. ACM Trans. Inform. Syst. Secur., 5(3):290–331, 2002.