IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB8250

Chapter XI

Protecting Datasources Over the Web: Policies, Models, and Mechanisms

Silvana Castano
Università degli Studi di Milano, Italy

Elena Ferrari
Università degli Studi dell'Insubria, Italy

ABSTRACT

Since the Web is becoming the main means of disseminating information in private and public organizations, both at internal and external levels, several applications at Internet and intranet level need mechanisms supporting a selective access to data available over the Web. Through XML, the document exchange and acquisition processes, which can be very frequent in Web-based systems, are simplified and standardized. The development of suitable security policies for both access control and information release and distribution are relevant research topics in the security field, and XML compatibility is an important requirement for Web datasource protection. This chapter covers the issues related to the definition of security policies, models and mechanisms for access control and dissemination of Web data, and is organized in two parts. In the first part, we introduce the general issues and requirements related to the definition of different types of security policies for access control and for information release in Web datasources. Then, we present security policies and mechanisms specifically devoted to the protection of XML data. In the second part, we describe the use of XML for the specification of security relevant information, focusing on security policies, subject credentials, and content protection.

INTRODUCTION

Since the Web is becoming the main means for disseminating information in private and public organizations, both at internal and external levels, several applications at Internet and intranet level need mechanisms supporting a selective access to data available over the Web. Moreover, information distribution occurs frequently in the Web context, and policies and mechanisms for a controlled release of information contents from a given source are also required. Information distribution often takes the form of documents that are made available through Web servers or are actively broadcast by Web servers to interested clients. Documents may also be exchanged among the various servers. The typical three-tier architecture for datasources over the Web consists of a Web client, network servers and the back-end information system with a suite of datasources (e.g., databases, XML sources, HTML sites) (Joshi, Aref, Ghafoor, & Spafford, 2001). Within this framework, public-key infrastructures (PKIs) (Stallings, 2000) represent an important development for addressing the security concerns of Web-based applications (e.g., for user authentication). However, these facilities do not provide mechanisms for access control to the contents of the sources, nor for their release and distribution. Protection of information sources over the Web and the development of suitable security policies for both access control and information release and distribution are relevant research topics in the security field. Security policies and related mechanisms can, in fact, be employed in several Web-based application scenarios such as digital library systems, B2B e-commerce applications, or Web-based information systems.

XML (eXtensible Markup Language) (World Wide Web Consortium, 1998) has recently emerged as being the most relevant standardization effort in the area of document representation through markup languages for Web-based information systems. Through XML, the document exchange and acquisition processes, which can be very frequent in Web-based systems, are simplified and standardized. XML compatibility is thus an important requirement for security policies, models and mechanisms for Web datasources. The widespread use of XML is emphasizing the need for models and techniques for securing XML data. These models and techniques are crucial in order to facilitate a selective dissemination of XML data containing information of different sensitivity levels, among (possibly large) user communities.

Securing XML documents entails addressing three main issues: confidentiality, integrity, and authenticity. Ensuring confidentiality means that the data object contents be disclosed only to subjects authorized according to the specified security policies. Ensuring integrity means ensuring that the object contents are not altered during transmission from the source to the intended recipient. Ensuring authenticity means that the subject receiving a data object is assured that the data object actually is from the source it claims to be. Confidentiality is ensured by access control mechanisms; integrity is usually enforced by access control mechanisms and by the use of encryption techniques, whereas authenticity requires the use of digital signature techniques (Stallings, 2000).

The relations between XML and security are not confined to the development of security models, techniques, mechanisms and systems for securing XML data. Another important aspect is whether and how XML can be exploited to specify security relevant information, such as organizational policies or content digital rights. Expressing security information in XML, even for non-XML data, has several advantages in an environment like the Web, in that it facilitates its transmission and exchange among heterogeneous sources.

This chapter covers the issues related to the definition of security policies, models and mechanisms for access control and dissemination of Web data and is organized in two parts. In the first part, we introduce the general issues and requirements related to the definition of different types of security policies for access control and for information release in Web datasources. Then, we present security policies and mechanisms specifically devoted to the protection of XML data. In the second part, we describe the use of XML for the specification of security relevant information, focusing on security policies, subject credentials, and content protection. The chapter ends with a discussion of challenging issues in the Web datasource protection field.

BACKGROUND

Background work relevant to the issues addressed in this chapter is mainly in the areas of *authorization-based access control*, *credential-based access control*, and *encryption-based and XML-based protection techniques* for Web data.

Authorization-based access control. In this area, background work ranges from some basic access control mechanisms provided by Web servers to expressive and powerful access control models, able to express a variety of security policies. Regarding Web server technology functionalities, usually they support the definition of access control lists (ACL) specifying which users are allowed/forbidden to connect to a Web server, such as the configuration files of the Apache server (XERCES PARSER). With this kind of access control mechanism, one can ensure protection at the directory granularity level by specifying a different configuration file for each directory to be protected. These kinds of mechanisms have great limitations when authorizations for controlling accesses to single files or portions of them need to be specified. In fact, a key requirement for providing powerful authorization-based access control mechanisms for Web data is the capability of referencing protection objects at varying granularity levels, based on the structure and semantics of the information contained in a Web document or a Web page. An initial effort in this direction is related to the use of HTML tags for expressing security relevant information for a given document. An access control model for Web documents has been proposed in Samarati, Bertino and Jajodia (1996). Here, HTML documents are considered, organized as unstructured pages connected by links. Authorizations can be given either on the whole document or on selected portions within the document.

The model allows a selective access to a document by authorizing a subject to see only some portions and/or links within the document. However, access control models for Web documents based on HTML have great limitations deriving from the fact that they are not based on a language able to semantically structuring the data. In particular, in order to give access to portions of a document, a manual split of the page into different slots on which different authorizations are given is required. The advent of XML offered new possibilities, well suited for providing powerful access control mechanisms, in that authorizations for access control can be based on semantic information provided by this language for various document components. In particular, the richer structure of XML documents with respect to HTML documents and the possibility of attaching a document type description (i.e., a DTD) to an XML document describing its structure, allow the definition and enforcement of more sophisticated access control policies than the ones devised for HTML documents.

Several research groups from both academia and industry are currently investigating problems related to security and XML. Work in this field has concentrated mainly on the development of access control models and encryption techniques (Pollmann, C. G). Regarding access control and XML, we recall the two main proposals by Bertino, Castano, Ferrari and Mesiti (1999a,1999b, 2000a, 2000b) and Damiani, De Capitani di Vimercati, Paraboschi, & Samarati (2000a). Both models support fine-grained access control to XML data, allowing the protection of single documents, sets of documents, as well as portions of them. Furthermore, they allow the association of a policy to a document type or to a single XML document. They support advanced features for authorization propagation and exception management, with the goal of limiting the number of authorizations to be specified for securing a given document. With respect to Damiani, De Capitani di Vimercati, Paraboschi, & Samarati (2000a), the model proposed in Bertino, Castano, Ferrari, & Mesiti, 2000a, 2000b) considers also the case of documents not conforming/partially conforming to a DTD, and provides a rich variety of options one can choose to deal with such documents. The model proposed in Damiani et al. (2000a) only provides the read access mode. The model in Bertino et al. (2000a, 2000b) supports a number of specialized access modes for browsing and authoring, which allow the Security Administrator to authorize a user to read the information in an element and/or to navigate through its links, or to modify/delete the content of an element, or to add/modify or delete one of the element links. Other research work focused on implementation issues of access control models for XML (Damiani et al., 2000b), design and implementation guidelines for implementing an access control processor for XML documents are discussed. The goal is to realize an access control processor as an extension to existing Web server technology, relying as much as possible on XML companion technologies. In Bertino, Castano, & Ferrari (2001a, 2001b), the *AuthorX* system is presented, providing a comprehensive environment for the protection of XML sources. Original features of *AuthorX* are the support for different policies for securing XML documents also in the case of partially and non-

conforming documents. Furthermore, *AuthorX* provides, in addition to the traditional user demand mode for document release, a broadcast mode based on a combination of *X-Sec* digital signature and encryption techniques, as discussed in Bertino, Castano, & Ferrari (2001b).

Credential-based access control. In a Web environment traditional identity-based schemes for access control are not enough, and the need arises for more flexible methods to qualify subjects entitled to access Web datasources. One of the most relevant research efforts in this area is represented by role-based access control (RBAC) models (Sandhu, Coyne, Feinstein, & Youman, 1996), which are receiving increasing attention as a generalized approach to enforce access control. In the RBAC model, roles represent functions within a given organization and authorizations are granted to roles instead of to single subjects. Examples of roles are Doctor, Cardiologist, Technical Manager, and Secretary. Under role-based models, all authorizations needed to perform a certain activity are granted to the role associated with that activity, rather than being granted directly to subjects. Subjects are then made members of roles, thereby acquiring the roles' authorizations. Subject access to data is mediated by roles; each subject is authorized to play certain roles and, on the basis of the role, he/she can perform accesses on the data. Whenever a subject needs to perform a certain activity, the subject needs only to be granted the authorization of playing the proper role, rather than being directly assigned the required authorizations.

A more general concept is the notion of *subject credential* first presented by Winslett, Ching, Jones, & Slepchin (1997). The idea is that in distributed systems, access control cannot always be based on subject-ids. Rather, subjects must provide information, typically about themselves (such as age, nationality, affiliation, and so on) allowing the access control mechanism to decide whether or not to give access. Building on this idea, Adam, Atluri, Bertino and Ferrari (2002) provide a formalization of the concept of subject credential by proposing a credential logic-based specification language. The advantage of using a formal language is that, it not only allows flexible specification of credentials, but also enables easy evaluation of authorization and verification of the consistency of specification.

The problem of subject qualification in a Web environment has also been addressed by Herzberg, Mass and Mihaeli (2000). In this chapter, the Trust Policy Language (TPL) is proposed. TPL is an XML-based language able to define mappings of strangers to business roles of a datasource, on the basis of digital certificates issued by third parties. The framework assumes that access to the datasource is governed by an RBAC model. Subjects accessing the datasource from all over the Web present their digital certificates to the datasource, which identifies a role to be assigned to the subject, based on a policy mapping from certificates to roles. After this mapping phase, access control takes place as in traditional RBAC systems. Mapping rules are expressed in TPL. Each role is associated with a set of TPL rules, defining how a certificate holder can become a member of that role. The authors show how TPL is expressive enough to allow the specification of complex

mapping policies, and outline an architecture to support the mapping process. The problem of management of heterogeneous certificates for e-business applications is investigated by the same research group in Herzberg and Mass (2001). They propose a high-level architecture, whose core module is the Credential Manager. The main task of the Credential Manager is to manage heterogeneous digital certificates for the same subject generated by multiple issuers, with possibly different formats. The Credential Manager exploits a set of agents, called collectors, to collect different kinds of certificates from different sources. The goal of the Credential Manager is to map all the certificates into a common and standard format. This is done by using an XML-based language called Credential Markup Language (CML).

Encryption-based and XML-based protection techniques. Several efforts have been made both in the area of encryption techniques for the development of protection systems for Web documents and for the development of XML-based formalisms for encoding security relevant information. For example, several tools are available (Gladney, & Lotspiech, 1997; Mauth, 1998; Sibert, Bernstein & van Wie, 1995; SoftLock Services) enabling secure content distribution. The basic idea common to these systems is the use of encryption techniques for document/content protection. The underlying scheme relies on a publisher responsible for managing documents to be protected and on encryption/decryption keys for document protection. Clients gain access to documents or portions of them by purchasing decryption keys. Protection is enforced by associating multiple encryption keys with the same document and by allowing the clients to purchase as many decryption keys as the number of different document portions they are interested in.

Because of the widespread use of XML and due the relevance of XML security, the World Wide Web Consortium (W3C) has set up several working groups to address the various security aspects. For example, the XML Working Groups of the W3C are working on standards for both an XML representation of digital signatures (W3C XML Signature Working Group) and encrypted contents (W3C XML Encryption Working Group). The OASIS Consortium for the design and development of industry standards language for XML-based interoperability, is setting up the XACML Technical Committee (Oasis Consortium). The committee is studying the definition of a standard model for XML-based security policies. Additionally, the Oasis Consortium is working on a language, called SAML–Security Assertion Markup Language, to allow security authorities in separate organizations to communicate about authentication, authorization, user profiles and authenticated user sessions in an open way.

SECURITY POLICIES FOR WEB DATA PROTECTION AND RELEASE

Security policies for Web data specify the general rules and principles for the protection and release of Web datasource contents. We distinguish two categories

of security policies for Web data: *access control policies* and *dissemination policies*. The former state the authorization rules and conditions according to which *subjects* are allowed to access *objects* (i.e., contents) stored at a given source under a given *access mode*. The latter state the modes under which the release of datasource contents should occur, given the access control policies defined at the source.

Before introducing a general template of security policy type for Web data, we discuss the main issues and requirements arising in defining security policies for Web data protection and release. These requirements are mainly dictated by the need for flexibility, to cope with such a dynamic and evolving environment.

Subject qualification. A major aspect of security policy definition regards the qualification of policy subjects, that is, the users (or the applications on behalf of the users) entitled to access data. Conventionally, subjects are referred to on the basis of an ''ID'' based mechanism, exploiting some information for their identification in the system. Examples of identity information could be a user-ID, an IP address, or a combination of user-ID and IP address. Such a way of referring subjects is usually adopted in database and operating system security policies (Castano et al., 1995). However, the population accessing Web datasources is generally composed of more heterogeneous users than the one of database or operating system environments, characterized by different skills and needs. Moreover, Web population is highly dynamic, in that the number and type of users is not known a priori and can change very frequently over time. In this context, conventional identity-based schemes are not sufficient to suitably qualify subjects, and more flexible and expressive schemes are devised, based on the notion of *subject credentials* (Winslett, Ching, Jones, & Slepchin, 1997). Subject credentials assert arbitrary properties of a subject, either personal characteristics, or characteristics and properties deriving from relationships the subject has with other subjects (e.g., qualifications within an organization). In this way, security policies can be specified in a more direct and intuitive way. We discuss subject credentials in more detail later in this chapter.

Protection granularity levels. A second aspect in security policy definition regards the establishment of the protection granularity levels, that is, the number and typology of protection objects to which a policy applies. Granularity levels are usually identified on the basis of the structure of the data to be protected. Web datasource contents generally have a nested, hierarchical structure, defined in terms of documents (or components) that can be themselves organized into subcomponents (e.g., an XML element and its subelements). Moreover, contents of a Web source are interlinked, to allow hypertextual navigation across related documents. Very often, source contents have varying protection requirements. In some cases, the same access control policy may apply to a set of documents in the source. In other cases, different access control policies may apply to different components within the same document. Many other intermediate situations may also arise. To support a differentiated and adequate protection of the contents of a Web datasource, the access control policies must be flexible enough to support a spectrum of protection granularity levels, such as the whole document, a document portion, a single

document component, or a set of documents. For example, consider a Web document describing a purchase order that provides descriptive information about the order and about the item(s) associated with it, with links to additional documents describing related clients. Having the possibility of specifying fine-grained protection objects, different policies can be formulated for the protection of the purchase order contents, stating, for instance, that information about the name of items could be made available to everyone, whereas information regarding the carrier should be released only to selected subjects, or that the link(s) to client documentation could be kept hidden from most subjects and made accessible only to a restricted number of authorized subjects.

Access privileges. Specification of access control policies requires also the identification of the privileges that subjects can exercise on protection objects. Privileges correspond to the different modes with which data to be protected can be accessed. Web datasource contents are generally accessible under two different modes: *browsing* and *authoring*. A browsing mode allows subjects to see information in a document and/or to navigate through its links without modifying it, whereas, an authoring mode enables subjects to modify document contents and/or document structure. Both browsing and authoring policies are thus required.

Specification levels. Security policies can be specified at two different levels, either instance-level or schema-level. Instance-level policies apply to a single instance (i.e., a Web document) only, while schema-level policies apply to a set of instances simultaneously. In analogy with conventional security policies for relational and object-oriented databases (Castano et al., 1995), the specification of schema-level policies for Web datasources must rely on a notion of document type, that is, an intensional description of a collection of documents in the source. Both instance-level and schema-level security policies are useful, since Web datasources present a wide spectrum of protection needs. In some cases, each single document in a source is characterized by peculiar protection requirements, different from the ones of the other documents in the source, such as in case of Web sites containing multiple, heterogeneous documents. In other cases, sets of documents share common protection requirements since enterprise-wide security policies are adopted, such as with organization sources. If several documents share common protection requirements, the specification of schema-level policies is convenient, since it avoids the definition of as many instance-level policies as the number of documents in the collection.

Exception management and propagation. Supporting fine-grained policies could lead to the specification of a, possibly high, number of access control policies, independent of the level at which policies are specified, either instance or schema level. In fact, one should specify a policy for each different protection granule of a target protection object. Additionally, the presence of several protection granularity levels entails the definition of a flexible and concise way of specifying exceptions, that is, situations where a protection object has security requirements different from those of its siblings in the hierarchical structure. To cope with these two requirements, access control policies for Web data need to rely on the advanced features of

positive and *negative* policies and of *propagation*. Positive policies specify permissions while negative ones specify denials. Propagation means that policies (either positive or negative) specified for a protection object at a given granularity level ''apply by default'' to all protection objects related to it according to a certain relationship in the hierarchical structure. By combining positive and negative policies with propagation, the number of policies to be specified for data protection and exception management reduces sensibly. By exploiting propagation, document protection can be enforced by specifying a variable number of authorizations on the document type, depending on the protection requirements to be enforced. For example, documents with homogeneous protection requirements can be protected by specifying only one policy defined at the document level, by asking a cascading propagation to all protection granules within the document. A document having highly heterogeneous protection requirements can be secured by defining a number of policies, one for each protection object with different protection needs, without propagation or with a limited propagation within the document itself.

Dissemination modes. Information dissemination occurs frequently in Web environments. Dissemination modes regulate the way in which documents of a Web datasource are released, given the access control policies defined at the source. Operating in the Web environment, it is worthwhile to have security policies enforcing at least two dissemination modes: *pull* and *push*. According to a pull dissemination mode, data reside at one or more servers and the subjects ask them when needed by issuing proper access request(s). Besides the traditional pull mode, a push dissemination mode can also be successfully adopted in the Web context, suitable for documents that must be released to a large community of subjects and which show a regular behaviour with respect to their release (e.g., they must be periodically distributed or when some predefined events happen). According to a push mode, a source periodically broadcasts (portions of) its Web documents to authorized subjects, without the need of an explicit access request by a subject. Dissemination policies are discussed in more detail in the section devoted to access control, since they entail different ways of enforcing the access control mechanisms. In the remainder of this section, we focus on access control policies. We first introduce the *policy type* template, a general template capturing all different types of policies than can be specified for Web data protection. Then, we discuss the instantiation of the policy type template to the protection of XML datasources, XML data protection being the most recent and relevant contribution in the Web datasource security field.

POLICY TYPES

A *policy type* for access control to Web datasources is a tuple of the form:

<Who, What, How, KindOfAccess, KindOfGrant, KindOfProp>

where:

- *Who* ∈ {*UserID, Credential*} denotes the way subjects are qualified in the policy, either identity-based or credential-based. According to the first type of policies, the users holding accounts in the Web datasource are considered as subjects of security policies. Users to which the policy applies are explicitly identified according to an ID-based mechanism (e.g., the login name with which the user connects to the server). The second type of policies relies on the availability of a set of subject credentials in the considered datasource, describing the characteristics of various users with respect to predefined credential types properly defined for access control purposes. Users to which the policy applies are thus implicitly specified by means of *credential expressions*, either simple or composite (i.e., expressed by means of logical operators) conditions on subject credentials.
- *What* ∈ {*WebDoc, DocType, SetofWebDocs, SetofDocTypes*} denotes the target protection objects to which the policy applies, either Web documents, or document types, or sets of them. A policy specified for a document type implies an analogous policy for all the Web documents that conform to this type. In this way, it is possible to specify policies either at the instance or schema level, as well as policies that apply to sets of documents/document types.
- *How* ∈ {*Whole, Portion, Content*} denotes the kind of policy protection granularity for the target protection object specified in the *What* field. A wide range of policies can be supported, ranging from policies that apply to the target as a whole to finer-grained policies that apply to selected portions of the target protection object, based also on its contents.
- *KindOfAccess* ∈ {*Browsing, Authoring*} denotes the kind of access to be allowed/denied on the protection object(s) of the policy. Both browsing and authoring policies can be specified, to cover all access needs related to Web content management.
- *KindOfGrant* ∈ {*Grant, Deny*} denotes the kind of authorization to be granted, either a positive authorization stating a permission or a negative one, stating a denial.
- *KindOfProp* ∈ {*Recursive, Limited, None*} denotes the kind of propagation allowed for a policy. Different propagation options are identified, based on the hierarchical structure of Web documents. These options specify whether and how a policy defined on a given protection object *o* propagates to protection objects that are related to *o* by some relationship. Basic propagation options for the hierarchical structure of Web documents are cascading propagation, by which a policy recursively applies to all sub-objects of a target protection object, limited propagation, by which a policy propagates only to the direct sub-objects of the target object, and no propagation, by which the policy applies only to the target protection object.

The policy type template is general and allows the specification of a variety of access control policies, at both schema-level and instance-level. In this way, the policy type acts as a design tool, allowing the Security Administrator to select the most appropriate type of policy for a target source, given the protection requirements

and rules to be enforced for its contents. For example, the policy type:

<*Credential,WebDoc,Portion,Browsing,Permission,Recursive*>

corresponds to a positive policy for credential-based browsing of a selected portion of a Web document, which recursively applies to all sub-objects contained in the selected portion. A subsequent level of instantiation of the policy type template is bound to the specific kind of datasource to be protected and to the access control model defined for such a source. The ultimate goal is to define a set of authorizations that can be enforced through the access control mechanisms to implement the initial protection requirements for the target source.

PROTECTION OF XML DATASOURCES

In this section we focus on the instantiation of the policy type template for the definition of access control policies for XML datasources. The instantiation is based on the access control model of *AuthorX* (Bertino et al., 2000a). *AuthorX* takes into account XML document characteristics, the presence of DTDs describing the structure of documents at a schema level, and the types of actions that can be executed on XML documents, for implementing security policies tailored to XML.

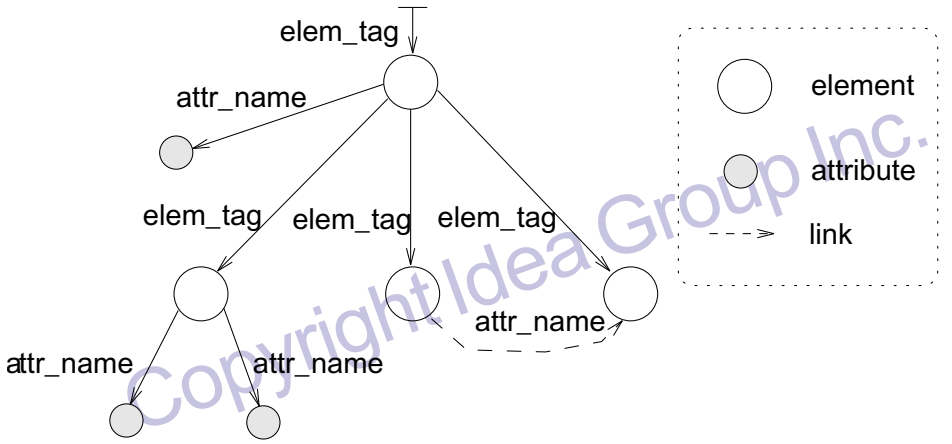
Building blocks of XML (World Wide Web Consortium, 1998) are nested, tagged *elements* (see Figures 1 and 2). Each tagged element has zero or more subelements and zero or more attributes. Elements can be nested at any depth in the document structure. Elements can be linked by means of IDREF(S) attributes. Consequently, target protection objects in *AuthorX* (*What* field of the policy type template) can be either an XML document or a DTD or a set of them. *AuthorX* allows a fine-grained protection of XML documents (*How* field of the policy type) reaching the granularity level of element, attribute, and link within a document. XPath expressions (World Wide Web Consortium, 1999) are used in *AuthorX* to denote specific portions within a target document, also based on content values.

Privileges associated with browsing policies in *AuthorX* (*KindOfAccess* field of the policy type template) are *view*, which authorizes a subject to view an element and/or (some of) its components, and *navigate*, which authorizes a subject to see the existence of a specific link or of all the links in a given element and to navigate through them. Privileges associated with authoring policies in *AuthorX* are *append* and *write*, where the append privilege allows a subject to write information in an element (or

Figure 1: Example of XML document

```
< elem_tag attr_name="..." >
< elem_tag attr_name="...", attr_name="..."
    < /elem_tag >
    < elem_tag attr_name= elem_id/ >
    < elem_tag attr_name="..." >
    < /elem_tag >
```

Figure 2: Graph representation of the XML document in Figure 1



in some of its parts) or to include a link in an element, without deleting any preexisting information, and the write privilege allows a subject to modify the content of an element and to include links in the element. *AuthorX* supports both positive and negative policies (*KindOfGrant* field of the policy type template). Two different types of propagation are provided in *AuthorX*, namely *implicit* and *explicit*. Implicit propagation is always applied by default, that is, it is automatically applied to all the specified policies and is based on the following principles: i) a policy specified at the DTD level automatically propagates to all the DTD instances, and ii) a policy specified on a given document/DTD element automatically propagates to all the attributes and links associated with the element. Implicit propagation principles can be overwritten, when required, however, by specifying proper explicit positive/negative access control policies. Explicit propagation is realized by means of propagation options in the access control policy specification itself to regulate whether and how a given policy propagates to protection objects that are at lower levels in the document or DTD hierarchy than those of the objects that appear in the policy specification. Three different options are provided for explicit propagation to specify that: i) no propagation is enacted (NO_PROP option), that is, the policy applies only to the protection objects which appear in its specification; ii) the access control policy propagates to all the direct child elements of the elements appearing in the policy specification (FIRST_LEVEL option); iii) the policy propagates to all the direct and indirect child elements of the elements appearing in the policy specification (CASCADE option).

AuthorX supports both DTD-level and document-level access control policies. DTD-level policies are associated with DTDs of an XML source, while document-level policies are associated directly with XML documents. DTD-level policies are well suited for the protection of valid documents, in that valid documents are always instances of some DTD in the source. A DTD-level policy can also be used for well-formed document protection, by first submitting well-formed documents to a classification process against secured DTDs available in the source. The goal of the

classification process is to find the "best-matching" DTD, that is, the DTD having the greatest number of protection objects matching the ones of the well-formed document to be protected. If such a DTD exists, its access control policies are propagated to the well-formed document. *AuthorX* implements a classification procedure based on the analysis of graph-based structures. The best-matching DTD returned by the classification procedure, if any, might be fully-matching or partially-matching, depending on the number of elements that match the ones of the target well-formed document. The interested reader can refer to (Bertino, Castano, & Ferrari, 2001b) for a more detailed description of the document classification process for security policy definition.

Document-level policies are employed for well-formed documents for which a matching DTD is not found, or for well-formed documents that do not undergo the classification process, or for valid documents having peculiar protection requirements.

Tables 1 and 2 summarize the semantics of the browsing and authoring document-level policies. The first column of each table specifies the privilege, the second one the propagation option, whereas the third one specifies the protection object to which the policy applies. Finally, the fourth column gives the authorization semantics of the policy. For simplicity, in these tables we consider positive policies only. The semantics of negative policies can be easily obtained by replacing permissions with denials in Tables 1 and 2, respectively. Analogous tables are easily definable for browsing and authoring DTD-level policies, which are not reported for the sake of conciseness.

As an example, in Figures 3 and 4, we show a portion of an XML document source extracted from the *Sigmod Record Articles XML Database* (Sigmod Record XML Database), while Figures 5 and 6 show its DTD.

For each issue, the document in Figure 3 provides information about the number and volume and about articles contained therein. Each article is characterized by information about title, authors, abstract, initial page and final page in the issue. Moreover, information about related articles is also provided.

Example 1. *Suppose that users with* `ACMmember` *credentials are authorized to see everything in the Sigmod Record XML source and that users with* `noACMmember` *credentials must be authorized to read all information about issues contained in the Sigmod Record XML Source, except articles'* `abstract`. *The first requirement is enforced by specifying one* `READ` *policy for* `ACMmember` *on the whole source (SigmodRecord.xml) with* `CASCADE` *propagation, that is,*

```
<ACMmember,SigmoRecord.xml,/issues,READ,GRANT,CASCADE>
```

The second requirement is fulfilled by specifying the following two credential-based policies for noACMmember:

```
<noACMmember,SigmoRecord.xml,/issues,READ,GRANT,CASCADE>
<noACMmember,SigmoRecord.xml,/issues/issuesTuple/articles/
    articlesTuple/abstract,READ, DENY,NO_PROP>
```

Where the first one is a positive policy for the `READ` *privilege with*

Table 1: Semantics of positive browsing policies on XML documents

Privilege	Prop. option	Prot. object	Semantics
read	cascade	Link	To view the link (without navigating it)
		Attribute	To view the value of the attribute
		Element	It has the same effect as a read policy on: all the attributes in the element, all the links in the element, all the direct and indirect sub elements of the element
		Document	It has the same effect as a read policy on: all the attributes in the document element, all the links in the document element, all the direct and indirect subelements of the document element
	first_lev	Link	To view the link (without navigating it)
		Attribute	To view the value of the attribute
		Element	It has the same effect as a read policy on: all the attributes in the element, all the links in the element, all the direct subelements of the element
		Document	It has the same effect as a read policy on: all the attributes in the document element, all the links in the document element, all the direct subelements of the document element
	no_prop	Link	To view the link (without navigating it)
		Attribute	To view the value of the attribute
		Element	It has the same effect as a read policy on: all the attributes in the element, all the links in the element
		Document	It has the same effect as a read policy on: all the attributes in the document element, all the links in the document element

Table 1: continued

navigate	cascade	Link	To activate the link
		Element	To activate: all the links in the element all the links in the direct and indirect subelements of the element
		Document	To activate: all the links in the document element all the links in the direct and indirect subelements of the document element
	first_lev	Link	To activate the link
		Element	To activate: all the links in the element all the links in the direct subelements of the element
		Document	To activate: all the links in the document element all the links in the direct subelements of the document element
	no_prop	Link	To activate the link
		Element	To activate all the links in the element
		Document	To activate all the links in the document element

`CASCADE` *propagation on the* `issue` *element, and the second one is a negative* `READ` *policy on the* `abstract` *subelement of* `issue`. *Suppose now that user* `Bob Smith` *(*`bob@someuniversity.edu`*) must be authorized to read all the information about the article identified by WB99. This requirement is enforced by specifying the following identity-based positive policy:*

```
<bob@someuniversity.edu,SigmoRecord.xml,/issues/issuesTuple/
    articles/articlesTuple[@id='WB99'], READ,GRANT, CASCADE>.
```

ACCESS CONTROL MECHANISMS

Once access control policies have been specified, according to some template, the next step is the development of suitable access control mechanisms to efficiently disseminate data according to the specified policies. A key point in setting up an access control mechanism for Web datasources is the definition of suitable *dissemination policies*. Dissemination policies state the modes under which the

Table 2: Semantics of positive authoring policies on XML documents

Privilege	Prop. option	Prot. object	Semantics
write	cascade	Link	To modify or delete the link
		Attribute	To modify or delete the value of the attribute
		Element	It has the same effect of a write policy on: all the attributes in the element all the links in the element all the direct and indirect subelements of the element
		Document	It has the same effect of a write policy on: all the attributes in the document element all the links in the document element all the direct and indirect subelements of the document element
	first_lev	Link	To modify or delete the link
		Attribute	To modify or delete the value of the attribute
		Element	It has the same effect of a write policy on: all the attributes in the element all the links in the element all the direct subelements of the element
		Document	It has the same effect of a write policy on: all the attributes in the document element all the links in the document element all the direct subelements of the document element
	no_prop	Link	To modify or delete the link
		Attribute	To modify or delete the value of the attribute
		Element	It has the same effect of a write policy on all the attributes/links in the element
		Document	It has the same effect of a write policy on all the attributes/links in the document element
append	cascade	Element	To add attributes/links in the element and in all its direct and indirect subelements
		Document	To add attributes/links in the document element and in all its direct and indirect subelements
	first_lev	Element	To add attributes/links in the element and in all its direct
		Document	To add attributes/links in the document element and in all its direct subelements
	no_prop	Element	To add attributes/links in the element
		Document	To add attributes/links in the document element

Figure 3: A portion of Sigmod Record XML articles XML database

```
<SigmodRecord>
    <issues>
      <issuesTuple>
        <volume>11</volume>
        <number>1</number>
        <articles>
          <articlesTuple id="WB99" related="KG98">
            <title>Annotated...</title>
            <initPage>45</initPage>
            <endPage>77</endPage>
            <abstract>...</abstract>
            <authors>
              <author AuthorPosition="00">
                Anthony I.Wasserman
              </author>
              <author AuthorPosition="01">
                Karen Botnich
              </author>
            </authors>
          </articlesTuple>
          …
          <articlesTuple id="KG98">
          …
          </articlesTuple>
          …
        </articles>
      </issuesTuple>
    </issues>
</SigmodRecord>
```


release of information from a given datasource should occur, given the access control policies defined at the datasource. In a Web environment, different dissemination policies can be devised. The simplest one is the *pull* dissemination policy (see Figure 7(a)), traditionally used in conventional DBMSs. Under this policy, a subject explicitly requests data to the datasource when needed. The access control mechanism checks those policies which apply to the requesting subject and, on the basis of these policies, returns the subject a view of the requested data, which contains all and only those portions for which the subject has an authorization. If the subject is not allowed to access any portions of the requested data, according to the specified access control policies, then the access is denied. Thus, the key issue in supporting information pull is how to efficiently build data views upon an access request on the basis of the specified access control policies.

Alternatively, data can be released according to the *push* dissemination policy

Figure 4: Graph representation of the XML document in Figure 3

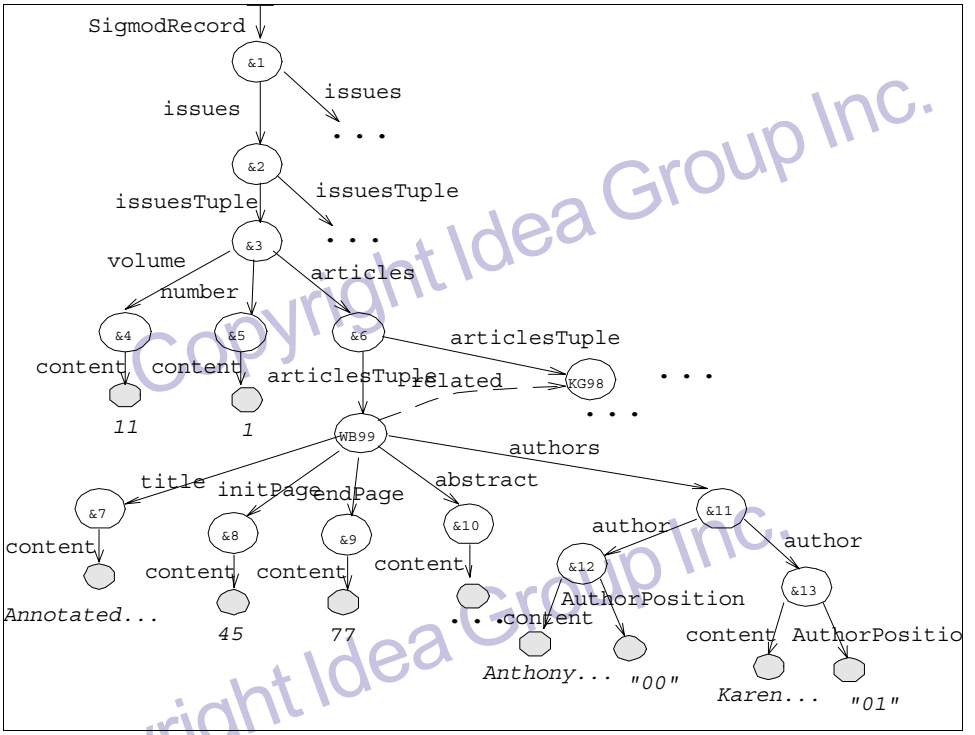


Figure 5: An example of DTD specification for the XML document in Figure 3

```
<!DOCTYPE SigmodRecord [
<!ELEMENT SigmodRecord (issues*)>
<!ELEMENT issues (issuesTuple*)>
<!ELEMENT issuesTuple (volume*,number*,articles*)>
<!ELEMENT volume (#PCDATA)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT articles (articlesTuple*)>
<!ELEMENT articlesTuple (title*,initPage*,endPage*,abstract*,authors*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT initPage (#PCDATA)>
<!ELEMENT endPage (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT authors (author*)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author AuthorPosition CDATA #REQUIRED>
<!ATTLIST article id ID #REQUIRED related IDREFS #IMPLIED>
]>
```

Figure 6: Graph representation of the DTD in Figure 5

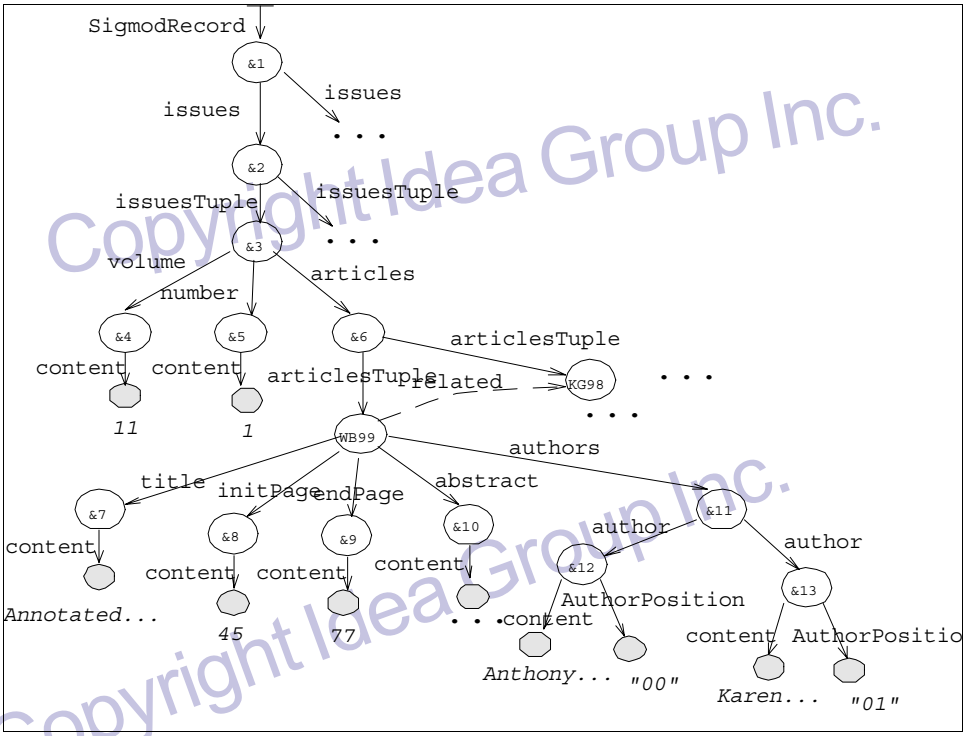
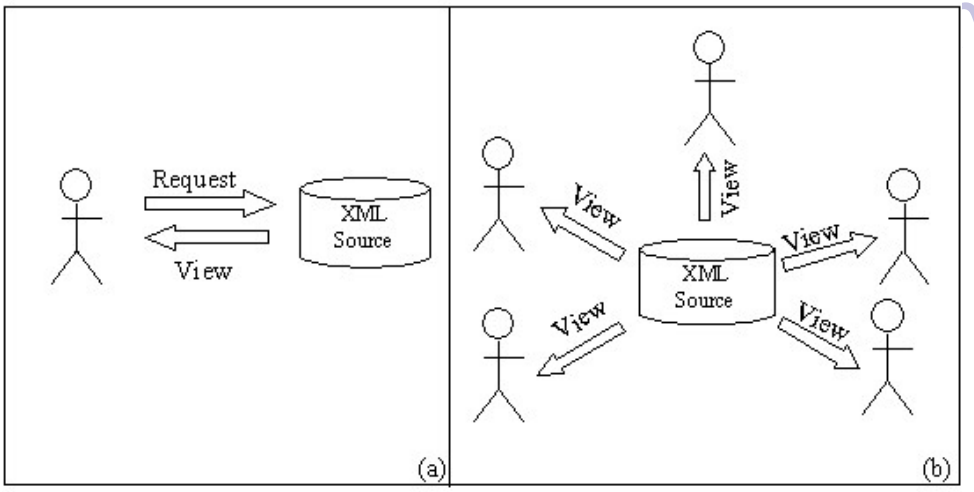


Figure 7: Dissemination policies: (a) information pull; (b) information push



(see Figure 7(b)). If such a policy is adopted, the datasource periodically broadcasts data (or portions of them) to its clients. Even in this case, different subjects may have privileges to see different, selected portions of the same data. Information push can be useful for data that have to be sent periodically (or whenever some events take

place) to a large community of subjects (e.g., the case of a newsletter sent once a week to all users). Information push could be used for both document distribution within the same organization (according to access control policies of the organization) or also for data distribution outside the organization, *owner* of the data. The latter case occurs, for example, when the organization sells information in the form of documents.

Supporting information push may entail generating different physical views of the requested data and sending them to the proper subjects. The number of these views may become rather large and thus such an approach cannot be practically applied. It is thus necessary to devise efficient strategies for multiple views generation and release. In the following, we describe information push and pull support for XML data provided by *AuthorX*.

PULL DISSEMINATION MODE

Under the pull dissemination policy, users submit explicit access requests to web datasources. An access request *r* is represented as a tuple `<user,target,path,acc_mode>`, where `user` is the identifier of the user requesting the access, `target` is the XML document to which the access is requested, `path` is an XPath expression which eventually selects specific portions of the requested document, and `acc_mode` ∈ {`browsing,authoring`} specifies whether a browsing or authoring access is requested.

Upon receiving an access request *r*, *AuthorX* checks which authorizations (both positive and negative) the `user` has on the target document, based on the `user` credentials and the specified access control policies. These authorizations are employed to build the document `view` to be returned to user, in response to his/her access request. The first step in building the user view is the *pruning phase,* during which the system removes all the unauthorized elements and attributes from the requested document. If the request is for a whole document, the pruned document is returned to the user submitting the request if it is not empty (otherwise the access is denied). By contrast, if the request is for selected portion(s) of the `target` document, the path contained in the access request is evaluated against the pruned document and the result is returned to `user`.

Example 2. *Suppose John, a noACMmember user, wants to get the article identified by WB99 by submitting the following access request:*

```
<john@someuniversity.edu,SigmodRecord.xml,/issues/issuesTuple/
    articles/articlesTuple[@id='WB99'],browsing>
```

Figure 8 shows the access control process. In the figure, we use symbol "-" instead of `DENY`*, and "+" instead of* `GRANT`*. AuthorX extracts from the policy base the browsing policies which apply to* `noACMmember` *and evaluates them against the file* `SigmodRecord.xml`*. The result of the evaluation is a graph, where each node is labeled with symbol "-", if a negative policy applies to the corresponding attribute/element, or with symbol "+", if a positive policy*

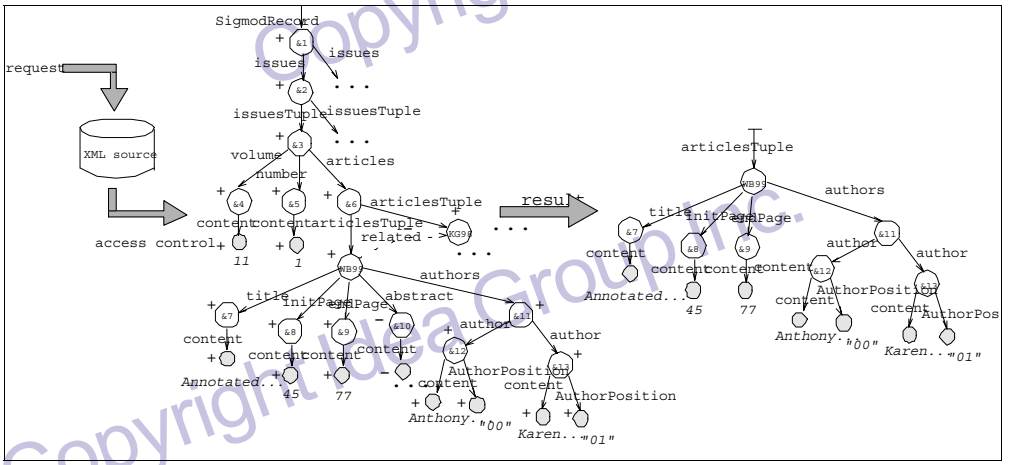
applies to the corresponding attribute/element. The view to be returned is obtained by dropping from the graph nodes with a label different from "+", and by extracting from the resulting graph the elements/attributes identified by the path: `/issues/issuesTuple/articles/articlesTuple[@id='WB99']` *contained in the access request. As a result, a view of article WB99 is returned to the requesting user (shown on the right hand side of Figure 8) that does not contain the* `abstract` *element, since a negative policy is specified on this element for user with a* `noACMmember` *credential.*

PUSH DISSEMINATION MODE

The main problem in supporting information push is that of generating (many) different physical views of the same datasource portion and sending them to a large community of users. To efficiently support the push dissemination policy, *AuthorX* adopts an approach that essentially allows the security administrator to send the same data portion to all subjects, while still enforcing the specified access control policies. This is achieved by making use of encryption techniques and by extending the Cryptolope approach proposed in Gladney and Lotspiech (1997). The idea is that, instead of generating different views of the datasource content for different subjects, a unique encrypted copy of the data to be released is generated, where different portions of the data are encrypted with different keys on the basis of the specified access control policies. Then, the same copy of the requested data is sent to all the subjects. By contrast, each subject receives all and only those decryption keys corresponding to data portions he/she is authorized to access. Enforcing this strategy entails addressing several problems:

- how to encrypt the data, that is, which portions of the data should be encrypted with different keys in such a way that the number of keys to be generated is minimized;

Figure 8: The access control process



- which keys should be distributed to which subject, in that a subject must be given all and only the keys that allow him/her to see the portions of the datasource for which the subject has a corresponding authorization;
- how to securely and efficiently distribute keys to subjects in such a way that keys are received only by the entitled subjects.

In the framework of *AuthorX* data encryption is driven by the specified access control policies: all the data to which the same access control policies apply are encrypted with the same key. Thus, to determine which keys should be sent to a particular subject, it is only necessary to verify which are the access control policies that apply to that subject and then send the keys associated with these policies.

As far as key distribution is concerned, it is important to note that the choice of the key distribution method may depend on many factors such as the number of subjects to which a document must be released, the explicit preferences stated by subjects, the number of keys generated during data encryption, the sensitivity of the data, and the subject behaviour (that is, whether they are always connected to the network or they are mobile users seldomly connected to the net). Since so many factors influence key distribution techniques, the better solution is to provide support for a spectrum of key distribution methods, thereby making the Security Administrator able to select the most appropriate method for each data portion. *AuthorX* supports both *online* and *offline* modes for key distribution. Under online modes the datasource delivers both the keys and the encrypted copy of the data to the subjects. Keys can be delivered together with the encrypted data. In such a case, the encrypted data are encapsulated into a *package* that includes an entry with the keys for each recipient (encrypted with the public key of the recipient). Alternatively, keys can be sent to each subject in a separate message using secure mail techniques (Stallings, 2000). By contrast, offline modes require subjects to retrieve the keys

Figure 9: General XML policy base template

```
<!DOCTYPE policyBase[
<!ELEMENT policyBase (policySpec)*>
<!ELEMENT policySpec (subject, object, priv, type, prop)>
<!ELEMENT subject (userID*|credential)>
<!ELEMENT object EMPTY>
<!ELEMENT priv EMPTY>
<!ELEMENT type EMPTY>
<!ELEMENT prop EMPTY>
<!ELEMENT userID EMPTY>
<!ELEMENT credential EMPTY>
<!ATTLIST userID id CDATA #REQUIRED>
<!ATTLIST credential targetCredType CDATA #REQUIRED credExpr CDATA
IMPLIED>
<!ATTLIST object target CDATA #REQUIRED path CDATA #REQUIRED>
<!ATTLIST userID id CDATA #REQUIRED>
<!ATTLIST priv value CDATA #REQUIRED>
<!ATTLIST type value CDATA #REQUIRED>
<!ATTLIST prop value CDATA #REQUIRED>
]>
```

through further interactions with the datasource. In this case, keys are stored at the server site using the LDAP protocol (Srivastava, 2000), and subjects query the LDAP directory to obtain the appropriate keys.

XML-BASED SPECIFICATION OF SECURITY INFORMATION

In this section, we discuss general principles of XML-based specification of security relevant information related to access control policies and subject credentials. We also provide samples of access control policies and subject credentials encoded using *X-Sec*, the XML-based language for security information specification defined in the framework of the *AuthorX* project (Bertino, Castano, & Ferrari, 2001c).

SECURITY POLICIES

Expressing security policies in XML results in the definition of an XML document type organizing the policy type template information in terms of the syntax and rules of XML. With reference to recent proposals made by the security community, information about subject, protection object, privilege, sign, type, propagation options are generally represented as elements of the XML policy specification, while predefined values associated with a field, if any, are represented through attributes and values of the corresponding element. The advantages of using XML for policy specification are related to the possibility of relying as much as possible on XML

Table 3: Attribute specifications for the XML policy template of Figure 8

Element	Attribute	Type	Value
UserID	id	CDATA	specifies user identity information
Credential	targetCredType	CDATA	subject credential
Credential	credExpr	CDATA	Xpath expression on subject credentials
Object	target	CDATA	denotes the Web document(s) to which the policy applies
Object	path	CDATA	denotes selected portions within the target document(s)
Priv	value	CDATA	specifies the security policy access mode
Type	value	CDATA	specifies whether the security policy is positive or negative
Prop	value	CDATA	specifies the propagation option of the security policy

companion technologies such as DOM, XSL, and XPath for their processing. Moreover, XML encoding of security policies makes easier the exporting of such information when a document migrates from a source to another, in that XML policies can be distributed together with data they refer to, properly linked through XML links.

The *X-Sec* encoding of security policies is based on the concept of policy base, whose general template is shown in Figure 9. According to this DTD, a policy base is an XML document encoding a collection of access control policies defined for a given Web datasource or for a given document. A `policySpec` element of the policy base represents a single policy within the policy base. Various fields of the policy type template are modeled by means of elements with associated subelements and/or attributes in the XML policy base template. Table 3 summarizes attribute specifications of the XML policy base template.

To encode identity-based access control policies, element `subject` in an *X-Sec* policy base has a list of `userID` subelements, each one with attribute `id` containing an ID value. For credential-based policies, `subject` element has a `credential` subelement with two attributes, namely `targetCredType` and `credExpr`. `targetCredType` specifies the name of subject credential(s) to which the security policy applies, while `credExpr` is an optional attribute of `credential` specifying the credential expression to be evaluated on the subject credentials specified in the `targetCredType`. When specified, the attribute restricts the set of users of `targetCredType` to whom the policy applies, based on the conditions specified in the credential expression. When the attribute is not specified, the security policy applies to all users qualified by subject credentials specified in `targetCredType`.

Element `object` of the policy base DTD encodes the protection objects to which the policy refers. It is characterized by the following attributes: `target`,

Figure 10: Policy base template for XML datasources

```
<!DOCTYPE policyBase[
<!ELEMENT policyBase (policySpec)*>
<!ELEMENT policySpec (subject, object, priv, type, prop)>
<!ELEMENT subject (userID*|credential)>
<!ELEMENT object EMPTY>
<!ELEMENT priv EMPTY>
<!ELEMENT type EMPTY>
<!ELEMENT prop EMPTY>
<!ELEMENT userID EMPTY>
<!ELEMENT credential EMPTY>
<!ATTLIST userID id CDATA #REQUIRED>
<!ATTLIST credential targetCredType CDATA #REQUIRED credExpr CDATA
    #IMPLIED>
<!ATTLIST object target CDATA #REQUIRED path CDATA #REQUIRED>
<!ATTLIST userID id CDATA #REQUIRED>
<!ATTLIST priv value (READ|NAVIGATE|APPEND|WRITE) #REQUIRED>
<!ATTLIST type value (GRANT|DENY) #REQUIRED>
<!ATTLIST prop value (NO_PROP|ONE_LEVEL|CASCADE) #REQUIRED>
]>
```

storing the name of the Web document(s) or document types to which the policy refers, and path, denoting a specific portion of the protection object(s) within `target`. `Priv` is an empty element with attribute `value` specifying the privilege, implementing the browsing and authoring policies. Element `type` stores the policy type value denoting whether the policy is positive or negative. Finally, element `prop` stores the value of propagation option for the policy at hand.

As an example, in Figure 10 we show the policy base DTD realized in *AuthorX* for the protection of XML sources.

Example 3. *The X-Sec policy base storing the policies of Example 1, is shown in Figure 11.*

Figure 11: An example of policy base

```
<policyBase>
  <policySpec>
   <subject><credential targetCredType="ACMmember"/></subject>
   <object>< target="SigmodRecord.xml" path="/issues"/></object>
    <priv value="READ"/>
    <type value="GRANT"/>
    <prop value="CASCADE"/>
  </policySpec>
  <policySpec>
   <subject><credential targetCredType="noACMmember"/></subject>
   <object>< target="SigmodRecord.xml" path="/issues"/></object>
    <priv value="READ"/>
    <type value="GRANT"/>
    <prop value="CASCADE"/>
  </policySpec>
  <policySpec>
   <subject><credential targetCredType="noACMmember"/></subject>
   <object>< target="SigmodRecord.xml" path ="/issues/issuesTuple/
    articles/articlesTuple/abstract"/></object>
    <priv value="READ"/>
    <type value="DENY"/>
    <prop value="NO_PROP"/>
</policySpec>
<policySpec>
  <subject><user userid="bob@someuniversity.edu"/></subject>
  <object><target="SigmodRecord.xml" path="/issues/issuesTuple/ar-
    ticles/articlesTuple[@id='WB99']"/></object>
    <priv value="READ"/>
    <type value="GRANT"/>
    <prop value="CASCADE"/>
</policySpec>
</policyBase>
```

SUBJECT CREDENTIALS

Expressing credentials using an XML-based language has the undoubted benefit of facilitating credential submission and distribution. XML encoding of subject credentials require to address two main issues: how to represent subject properties and characteristics in XML and how to ensure authentication of these properties. In the following, we present XML subject specification in *AuthorX*, which is compliant with mainly of the XML representation of subject credentials proposed in the literature (such as for instance Herzberg & Mass, 2001 and Damiani et al., 2000b).

Credentials in *AuthorX* are based on the concept of *Credential-types*. A credential-type is a group of subject credentials with a similar structure. Examples of credential-types in the `Sigmod Record` domain are `ACM members`, `non ACM members`, and `special interest user`. Conceptually, a credential-type can be modeled as a tuple <*CredentialName, CredentialProperties*>, where *CredentialName* is the name of a credential-type, and *CredentialProperties* is a set of property specifications. A property specification gives the name and the domain of a property. Credential-type properties can be either simple or composite. Simple properties take values from basic domains (e.g., `integer`, `string`), whereas composite properties take values from domains defined by applying conventional constructors (e.g., `set`, `record`, `list`) on basic domains. In an XML-based representation of subject credentials, credentials expressed as XML documents, are *instance* of credential-types, expressed as DTDs. In *AuthorX*, both XML credentials and credential-types are specified using *X-Sec*. As an example, Figures 11(a) and (b) reports the *X-Sec* `ACMmember` credential-type and a corresponding subject credential, respectively.

An *X-Sec* credential-type is thus a DTD where simple properties are modeled as empty elements and composite properties as elements with element content, whose subelements model composite property components. For instance, the *X-Sec* credential-type in Figure 11(a) contains the basic information about an ACM member, namely, his/her name, organization, email, and membership number. With

Figure 11: Example of X-Sec credential-type (a) and of a corresponding X-Sec subject credential (b)

```
<!DOCTYPE ACMmember [
<!ELEMENT ACMmember (name,organization,
                     email*, memberNr)>
<!ELEMENT name (fname, mname?, lname)>
<!ELEMENT organization (#PCDATA)>
<!ELEMENT mail (#PCDATA)>
<!ELEMENT memberNr (#PCDATA)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT mname (#PCDATA)>
<!ELEMENT lname (#PCDATA)>
<!ATTLIST ACMmember CredID ID #REQUIRED>
<!ATTLIST ACMmember CIssuer CDATA #REQUIRED>
]>
```

(a)

```
<ACMmember CredID="154" CIssuer="CA16" >
    <name>
       <fname>Tom</fname>
       <lname>Tom</lname>
    </name>
  <organization>MIT</organization>
    <email>twatson@mit.com</email>
    <memberNr>2001</memberNr>
</ACMmember>
```

(b)

reference to Figure 11(a), `<!ELEMENT email #PCDATA>` is an example of simple property, representing the email of an ACM member, whereas `<!ELEMENT name(fname, lastname)>` is an example of composite property, because name is composed by fname and lname properties. An *X-Sec* credential is an XML document instance of an *X-Sec* credential-type, and specifies the set of property values characterizing a given subject against the credential-type itself. Credentials are certified by the credential issuer (e.g., a certification authority) using standard digital signature techniques (Stallings, 2000) (attribute `cIssuer` in Figure 11). The credential issuer is also responsible for certifying properties asserted by credentials themselves.

FUTURE TRENDS

The development of encryption and digital signature techniques and access control mechanisms specifically tailored to XML raises several new issues and requires extensive revisiting of techniques and models developed in other contexts. In the following, we discuss some of the most promising research directions in this field.

ADVANCED SCHEMES FOR CREDENTIAL MANAGEMENT AND EVALUATION

An important issue in supporting credential-based access control for Web datasources is to devise suitable schemes for credential evaluation and management in a heterogeneous and dynamic environment like the Web. In the following, we discuss possible schemes that can be adopted in the Web environment.

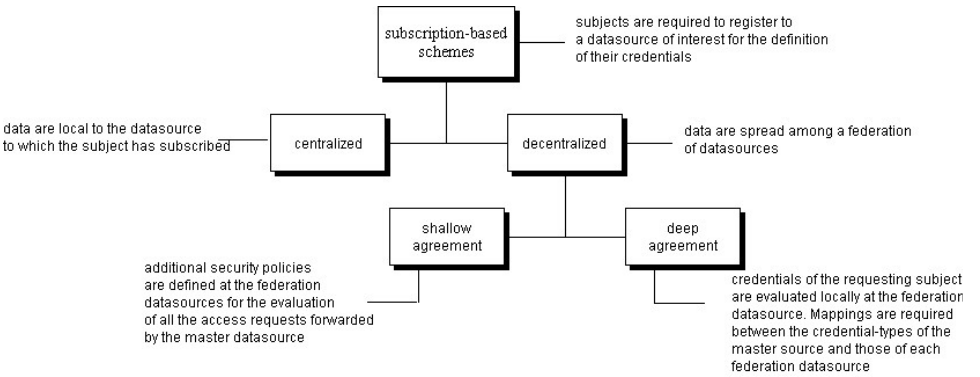
Subscription-based schemes. An option is to use a scheme, traditionally used in digital library systems. According to this scheme, subjects are required to register with a datasource of interest, during a mandatory subscription phase, for the definition of their subject credentials. Among the subscription-based schemes, a further distinction can be made on the basis of where the data reside, i.e., whether they are local to the datasource to which the subject subscribes (*centralized scheme*) or are distributed among a federation of datasources (*decentralized scheme*). According to a centralized subscription scheme, a single datasource is responsible for managing the policy and credential bases for the data to be protected. The source defines its own credential-types and generates subject credentials based on information supplied by subjects during the subscription phase. Credentials and access control policies are then used for governing access to documents upon subject requests. By contrast, in a decentralized subscription scheme, data are distributed among several datasources forming a *federation*. Each datasource of the federation manages its own policy and credential bases and is responsible for their definition. One of the datasources of the federation plays the role of *master datasource* to

which all subjects must subscribe. Subjects submit access requests to the master datasource. The master datasource processes the access request locally by evaluating the credentials of the requesting subject. Moreover, it forwards the initial subject requests to all or part of the federation datasources. Thus, such scheme requires also an *agreement phase* between the master datasource and each datasource of the federation. The agreement phase has the goal of reaching an agreement on the access control policies and the credential evaluation methods to be used by federation datasources for processing an access request received from the master datasource. Two alternatives are possible: a *shallow agreement* requires the definition of additional access control policies, for governing the accesses of the master datasource to the data stored at each federation datasource. In this case, access requests of different subjects forwarded by the master datasource to a given federation source are treated uniformly at this latest source, according to the master datasource policies specified at this datasource, and thus the credentials of the requesting subject are not evaluated locally at the federation datasources. By contrast, the *deep agreement* requires the definition of mappings between the credential-types defined at the master datasource and the credential-types defined at each federation datasource, to guarantee semantic conversion of possibly heterogeneous, subject credentials. The goal is to make subject credentials defined at the master datasource understandable by a given federation datasource. In this mode, access requests of subjects with different credentials forwarded by the master datasource to a given federation datasource are treated differently at this latest datasource, on the basis of mapped credential-types. Under the deep agreement mode, the credential base of the master datasource needs to be extended with *conversion rules*, to enforce required credential-type mappings.

A taxonomy for the subscription schemes discussed so far is reported in Figure 12.

Negotiation-based schemes. Another alternative is the adoption of a *negotiation-based* scheme, according to which the requesting subject can access datasource contents by presenting its credentials directly. A datasource can be locally responsible for accepting credentials presented by a subject or can refer to

Figure 12: Taxonomy for subscription-based schemes



a third party (or a chain of third parties) for getting the appropriate certificates. Issues related to the development of a negotiation-based scheme are part of the trust management problem in distributed systems (Winsborough, Seamons, & Jones, 2000), for which some preliminary techniques have been proposed in the literature, that we surveyed in the related work section. In this respect, we note that using XML to express access control policies makes it easier to export such information when data migrates from one datasource to another.

INCORPORATING PROTECTION MECHANISMS INTO EXISTING TECHNOLOGY AND ARCHITECTURES

A challenging direction for research regards the way protection mechanisms are to be developed for Web datasources so that they can be easily incorporated into existing technology and Web-based enterprise information system architectures. The Web community regards XML as the most important standardization tool for information exchange and interoperability. We argue that compatibility with XML companion technologies is an essential requirement in the implementation of any access control and protection mechanism. In this way, XML-based access control can constitute the core mechanism of Web-based enterprise architectures. Crucial aspects of this process are related to performance issues both at the theoretical and at the experimental level, as well as to scalability aspects. An additional direction of research concerns the development of protection mechanisms compliant with existing standards, protocols, and technologies in order to achieve secure interoperation among multiple heterogeneous systems in the Web environment. Powerful and flexible access control models need to be implemented on top of existing workflow management systems and e-business infrastructure technology. Within this framework, a new research area has emerged devoted to studying models, protocols, and mechanisms for securely invoking and accessing e-services in the XML context (Box, 2000).

COMPLIANCE WITH EMERGING XML, SECURITY, AND WEB-BASED STANDARDS

The development of suitable mechanisms and models for the protection of Web datasources cannot neglect to consider the most relevant standardization efforts in the area of Web-based applications, security, and XML. Main relevant standards to be mentioned are, for example, the standard for an XML representation of digital signatures (W3C XML Signature Working Group) and the standard for encrypted contents (W3C XML Encryption Working Group), developed by the XML Working Group of the W3C. The OASIS XACML Technical Committee (Oasis Consortium) is developing the definition of a standard model for XML-based security policies. The

Oasis Consortium is also working on the definition of an XML specification for authentication and authorization information called SAML, to allow security authorities in separate organizations to exchange security relevant information in an open way.

Other relevant standards in the security field are those related to digital certificates, such as X.509 or SPKI certificates (Stallings, 2000). Some proposals already exist for an XML encoding of these certificates (e.g., XML-SPKI (Paajarvi, 2000)), and the XML encoding of X.509 certificates proposed in Herzberg, Mass and Mihaeli (2000). Integration of credential-based access control with conventional certificate management approaches is an important issue and a key requirement for effective use of credential-based access control in the Web environment. We believe that it is important that new proposals of access control models for Web datasources be designed to be compliant with such emerging standards. On the other site, it is also crucial that new standards be influenced by the existing research proposals and incorporate as much as possible their main features.

CONCLUSIONS

In this chapter, we have discussed the issues related to the definition of security policies, models and mechanisms for access control and dissemination of Web data, with particular attention to the protection of XML datasources. We also described the use of XML for specification of security relevant information related to security policies.

Protection of Web datasource is a new and rapidly-evolving area in which the XML language can play an important role both for data and security-related information representation. In this chapter, we have covered the main issues related to the definition and enforcement of security policies for Web datasources and outlined main future trends.

REFERENCES

Adam, N., Atluri, V., Bertino, E. and Ferrari, E. (2002). A content-based authorization model for digital libraries. *IEEE Transaction on Knowledge and Data Engineering*, *14*, 296-315.

Bertino, E., Castano, S., Ferrari, E. and Mesiti, M. (1999). Controlled access and distribution of XML documents. *Proceedings of the 2nd ACM Workshop on Web Information and Data Management (WIDM'99)*, Kansas City, Missouri.

Bertino, E., Castano, S., Ferrari, E. and Mesiti, M. (2000a). *Author-X*: a Java-based system for XML data protection. *Proceedings of the 14th Annual IFIP WG. 11.3 Working Conference on Database Security*, Schoorl (near Amsterdam), The Netherlands.

Bertino, E., Castano, S., Ferrari, E. and Mesiti, M. (2000b). Specifying and enforcing access control policies for XML document sources. *World Wide Web Journal*, *3*(3), 139-151.

Bertino, E., Castano S. and Ferrari, E. (2001a). Securing XML documents: The *Author-X* project demonstration. *Proceedings of the SIGMOD 2001 Conference*, Santa Barbara, California.

Bertino, E., Castano, S. and Ferrari, E. (2001b). *Author-X*: A comprehensive system for securing XML documents. *IEEE Internet Computing*, *5*(3), 21-31.

Bertino, E., Castano, S. and Ferrari, E. (2001c). On specifying security policies for Web documents with an XML-based language. *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, Fairfax, Virginia.

Blaze, M., Feigenbaum, J. and Lacy, J. (1996). Decentralized trust management. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California.

D. Box (2000). *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium. http://www.w3.org/TR/SOAP.

Castano, S., Fugini, M. G., Martella, G. and Samarati, P. (1995). *Database Security*. Reading, MA: Addison-Wesley.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. (2000a). Securing XML documents. *Proceedings of the Int'l Conference on Extending Database Technology (EDBT2000)*, Konstanz, Germany.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. (2000b). Design and implementation of an access control processor for XML documents. *Proceedings of the Int'l Conference on World Wide Web (WWW9)*, Amsterdam, The Netherlands.

Gladney, H. and Lotspiech, J. (1997). Safeguarding digital library contents and users: Assuring convenient security and data quality. *D-lib Magazine*.

Herzberg, A. and Mass, Y. (2001). Relying party credentials framework. *Proceedings of the RSA Conference*, San Francisco, California.

Herzberg, A., Mass, Y. and Mihaeli, J. (2000). Access control meets public key infrastructure, or: assigning roles to strangers. *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, California.

Joshi, J.B.D., Aref, W.G., Ghafoor, A. and Spafford, E.H. (2001). Security models for Web-based applications. *Communications of the ACM*, *44*(2).

Mauth, R. (1998). Better copyright protection. *Byte Magazine*.

OASIS Consortium**.** (2002). http://www.oasis-open.org.

Paajarvi, J. (2000). *XML Encoding of SPKI Certificates*. Internet Draft. http://www.ietf.org/ietf/1id-abstracts.txt.

Park, J., Sandhu, R. and Ahn, G.J. (2002). Secure Attribute Services on the Web. *ACM Transactions on Information Systems and Security*. To appear.

Pollmann, C. G. (2002). *The XML Security Page*. http://www.nue.et-inf.uni-siegen.de/~geuer-pollmann/ xml/security.html.

Samarati, P., Bertino, E. and Jajodia, S. (1996). An authorization model for a distributed hypertext system. *IEEE Transactions on Knowledge and Data Engineering*, *8*(4), 555-562.

Sandhu, R., Coyne, E., Feinstein, H. and Youman, C. (1996). Role-based access control models. *IEEE Computer*, *29*(2), 38-47.

Sibert, O., Bernstein, D. and van Wie, D. (1995). The DigiBox: A self-protecting container for information commerce. *Proceedings of the 1st Usenix Workshop on Electronic Commerce,* New-York.

Sigmod Record XML Database. (2002). http://www.dia.uniroma3.it/Araneus/Sigmod.

SoftLock Services, Inc. (2000). *SoftLock*. http://www.softlock.com.

Srivastava, D. (2000). Directories: Managing data for networked applications. Tutorial presented at the *16th IEEE International Conference on Database Engineering (ICDE'00)*, San Diego California.

Stallings, W. (2000). *Network Security Essentials: Applications and Standards*. Englewood Cliffs, NJ: Prentice Hall.

Winsborough, W., Seamons, K. and Jones, V. (2000). Automated trust negotiation. *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'2000)*.

Winslett, M., Ching, N., Jones, V. and Slepchin, I. (1997). Using digital credentials on the World Wide Web. *Journal of Computer Security*, 7.

World Wide Web Consortium. (1998). *Extensible Markup Language (XML) 1.0*. http://www.w3.org/TR/REC-xml.

World Wide Web Consortium. (1999). *XML Path Language (XPath), 1.0, W3C Recommendation*. http://www.w3.org/TR/xpath.

W3C XML Encryption Working Group. (2001). http://www.w3.org/Encryption/2001.

W3C XML Signature Working Group. (2001). http://www.w3.org/Signature.

XERCES PARSER. (2001). *Apache*. http://www.apache.org.