CERIAS Tech Report 2004-65

XML-BASED SPECIFICATION FOR WEB SERVICES DOCUMENT SECURITY

by R.Bhatti, E. Bertino, A.Ghafoor, J.B.Joshi

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

XML-Based Specification for Web Services Document Security

Document security in XML-based Web services has become increasingly important for managing secure business transactions over the Web. The authors propose an XML-based access control specification language to address this security challenge.

Rafae Bhatti
Elisa Bertino
Arif Ghafoor
Purdue University

James B.D. Joshi
University of Pittsburgh

The Internet and related technologies have seen tremendous growth in distributed applications such as medicine, education, e-commerce, and digital libraries. As demand increases for online content and integrated, automated services, various applications employ Web services technology for document exchange among data repositories. Web services provide a mechanism to expose data and functionality using standard protocols, and hence to integrate many features that enhance Web applications.

XML, a well-established text format, is playing an increasingly important role in supporting Web services. XML separates data from style and format definition and allows uniform representation, interchange, sharing, and dissemination of information content over the Internet.[1,2] It is thus a natural contender as a standard for marking up the data that distributed Web-based applications exchange.

This interoperability paradigm lets businesses dynamically publish, discover, and aggregate a range of Web services through the Internet to more easily create innovative business processes and value chains.[3] This advantage, however, is accompanied by security concerns related to disseminating secure documents. Security has become a primary concern for all enterprises exposing sensitive data and business processes as Web services.

XML and Web services provide a simplified application integration framework that drives demand for models that support secure information interchange. Examples of secure Web services that require stricter access controls include searching digital library contents based on user privileges, retrieving results from a medical center's patient database based on user status, and exchanging sensitive financial data between institutions based on user membership levels.

Providing document security in XML-based Web services requires access control models that offer specific capabilities. Our XML-based access control specification language addresses a new set of challenges that traditional security models do not address.

CONTENT-BASED CONTEXT-AWARE ACCESS

Information access may require restrictions based on the content and context related to the access requests. For example, a digital library can contain images depicting scenes inappropriate for children, and a Web service that provides access to such a resource should deny access to users in a certain age group. Similarly, for Web services in the healthcare industry, relevant parties, including physicians, should have access to selective content-based patient information.

0018-9162/04/$20.00 © 2004 IEEE

The RBAC model simplifies authorization administration by assigning permissions to users through roles.

The access control model should also capture security-relevant environmental context and incorporate it in its access control decisions. The model can make access request decisions based on user domains, which are classified by IP addresses. For example, a Web service that provides digital library content can make certain resources always available to users who belong to certain domains.

A practical example of such a service is an online digital library, which lets students of subscribing universities access the library from within the university intranet. In this case, the necessary context information is that the access request comes from an authorized IP address.

Subject and object heterogeneity

The secure documents that XML-based Web services disseminate encompass diverse subjects and objects related to the applications. Object heterogeneity can exist either as abstract concepts or as knowledge embodied in the information that requires protection. For example, the enormous volume of data in a digital library Web service makes exercising access control for high-level concepts rather than for individual objects highly desirable. Further, information content can evolve with time as the library adds new documents and removes or updates old ones, introducing scalability problems in privilege management.

Subject heterogeneity complicates access control specification. It implies that users have diverse activity profiles—characteristics or qualifications that may not be known a priori. Activity profiles are necessary to dynamically transfer authenticated users from one Web service to another. Consider a user who subscribes to Yahoo but does not subscribe to Yahoo's Web services partner, Travelocity. However, during a login session on Yahoo, the system can grant the user access to resources on Travelocity based on the current login information. Another example of dynamically changing subject profiles is a mobile user communicating over a wireless network.

Role-based access control model

Our XML-based specification language incorporates these content- and context-based dynamic security requirements for documents in XML-based Web services. Our approach provides access control with an element-level granularity for Web services with specific document security requirements and enforces concept-level access control on the underlying data repositories. We base our specification on the role-based access control (RBAC) model, which is particularly suitable for Web applications[4] because it can define a diverse set of access control policies.[5,6]

A key advantage of the RBAC model is that it simplifies authorization administration by assigning permissions to users through roles. Thus, it adds a layer of abstraction between users and their permissions.

Researchers have proposed various access-control models for securing XML documents.[1,2,7] A closely related work, the OASIS XACML (Extensible Access Control Markup Language; xml.coverpages.org/xacml.html) standard, uses an extension of XML to define an access-control language specification.

Although previous work, including XACML, supports the notion of content-based context-aware access,[1] it does not provide explicit support to manage subject and object heterogeneity. This work lacks notions of conceptual-level access control on objects or for maintaining and updating dynamically changing user profiles. Hence, these schemes would not be suitable for Web services environments that face subject- and object-heterogeneity challenges. Also, all of these schemes assign permission directly to users rather than assigning roles to abstract permission, which violates the principles of scalability and manageability that motivates developers to use RBAC.[5]

To the best of our knowledge, an XML-based RBAC language for document security in XML-based Web services has not been investigated previously. Our work aligns with the existing work related to Web services security frameworks, such as Microsoft HailStorm (www.microsoft.com/presspass/features/2001/mar01/03-19hailstorm.asp) and a service architecture that IBM and Microsoft jointly proposed (www-106.ibm.com/developerworks/security/library/ws-secmap/).

Our approach does not substitute for the features these frameworks already incorporate, such as Web services security specifications or the passport authentication system. Instead, it complements them by providing a policy specification and enforcement mechanism that could be implemented using existing standards, such as WS-Policy, then incorporated within these XML-based frameworks to meet the target organization's specific needs. Thus, the model we propose is both modular enough for use with existing Web services security frameworks and extensible enough for development into a complete Web services security framework.

RBAC EXTENSIONS

The RBAC model has five primary elements: users, roles, permissions, operations, and objects. These elements are related through set-relations and functions. Permissions are composed of an object-to-operations mapping. Our specification captures both the core RBAC model semantics and extensions to the core model, including role hierarchies and separation of duty constraints.[8,9]

Our model uses a location-based approach to capture the context information. A session parameter records the domain from which the user requests access. In addition to the requesting user's domain, the session schema also contains attributes that capture the user's activity profile such as login_time, login_date, and the session's duration. The model processes such information dynamically and incorporates it into access decisions in which context information can be an important decision parameter.

For example, consider a continuation of the online digital library example in which the library offers a collaborative subscription to a group of small universities that cannot afford an exclusive membership. In such a scenario, although access is available to students from the group of smaller subscribing universities, this access may be restricted based on either prespecified access slots or the access duration for each university that shares the infrastructure. The restriction arises from a need to achieve a balanced system load. Maintaining such context information, and constantly updating it, can be a challenging task in distributed environments, especially those with mobile users undergoing domain transfers due to reasons such as handoff. Correctly restoring valid connections becomes critical, and it is an issue our software architecture addresses.

Our framework allows content-based specification at four levels: conceptual, schema, instance, and element. Grouping information content into concept clusters reduces the complexity of the specification process and security administration.

This approach uses a similarity-based function for content classification.[10] The similarity-based function analyzes the content-related metainformation or schema information available in XML documents, then groups related XML schemas and their instances into a cluster. The classification creates document clusters and assigns roles related to the concept to these clusters. The classification process can organize such roles as a hierarchy that satisfies the aggregate relation.

As Figure 1 shows, a cluster can contain an arbitrary number of XML schemas, XML instances, or their elements and attributes. Once the classification process has created document clusters, the system administrator generally specifies additional fine-grained access restrictions within valid XML document instances. Our approach, however, assumes that the administrator has not specified any negative permissions. Thus, once an administrator at a higher level has granted access, there is no need for an overriding policy specification at a lower level. If a user with new credentials needs a predetermined role, the system might need to create a virtual cluster dynamically based on the new credential information.

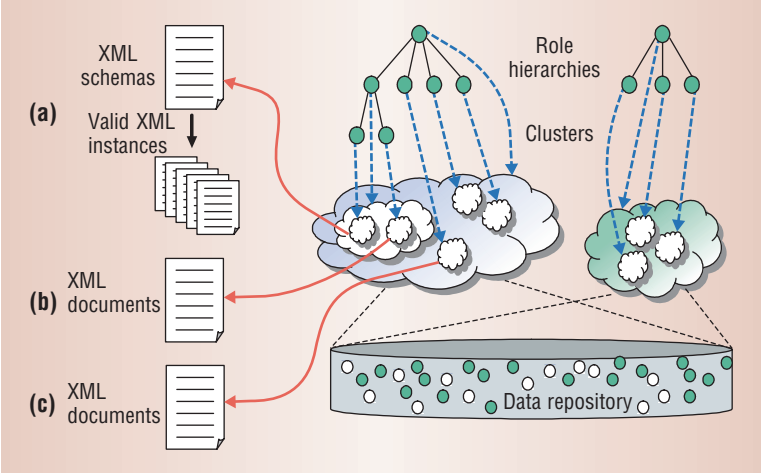

Figure 1. XML document clustering and associated roles. (a) The schema and all of its XML document instances fall under the cluster; (b) the XML document falls under the cluster; (c) the XML document element falls under the cluster.

XML-BASED SPECIFICATION LANGUAGE

Our XML-based specification language models the RBAC elements and incorporates the functional specifications according to the NIST RBAC standard.[9]

Modeling RBAC elements

Our specification models the five basic RBAC elements and their relationships. We use XML to generate schema definitions for the user, role, and permission elements. Schema definition is unnecessary for the operation and object elements because the specification includes them in a permission definition according to the NIST standard, so the permission schema captures their relationship with other RBAC elements.

User credentials. To evaluate a particular user's role, the specification language uses the notion of credentials.[1] To group users, an administrator defines a credential type by identifying a common set of attribute-value pairs. Consider, for example, the following user credential based on a general credential expression of the form (*cred_type_id*, {*cred_expr*}), where *cred_type_id* is a unique credential type identifier and *cred_expr* is a set of attribute-value pairs:

(Nurse, {(user_id, John, *mand*), (age, 30, *opt*), (level, fifth, *mand*)})

```
<credentials>
 <credential>
  <cred_type
   cred_type_id = 'C100'>
   Nurse
  </cred_type>
  <cred_expr>
   <attribute_value_list>
    <attribute_value_pair>
        <attribute_name>
         user_id
        </attribute_name>
        <attribute_value>
         John
        </attribute_value>
    </attribute_value_pair>
    <attribute_value_pair>
        <attribute_name>
         age
        </attribute_name>
        <attribute_value>
         30
        </attribute_value>
    </attribute_value_pair>
    <attribute_value_pair>
        <attribute_name>
         level
        </attribute_name>
        <attribute_value>
         5
        </attribute_value>
    </attribute_value_pair>
   </attribute_value_list>
  </cred_expr>
  <max_roles>2</max_roles>
 </credential>
</credentials>
```

(a)

```
<roles>
 <role>
  <role_name>Doctor
  </role_name >
  <SSD_Role_Set_id>SSD1
  </SSD_Role_Set_id>
  <junior>Resident</junior>
  <cardinality>8</cardinality>
 </role>
 <role>
  <role_name>DBA</role_name>
  <SSD_Role_Set_id>SSD1
  </SSD_Role_Set_id>
  <DSD_Role_Set_id>DSD1
  </DSD_Role_Set_id>
 </role>
</roles>
<SSD_Role_Sets>
 <SSD_Role_Set
 SSD_Role_Set_id = 'SSD1'
 SSD_cardinality = '1'>
  <SSD_Role>Nurse</SSD_Role>
  <SSD_Role>Doctor</SSD_Role>
  <SSD_Role>Dispenser</SSD_Role>
  <SSD_Role>DBA</SSD_Role>
 </SSD_Role_Set>
</SSD_Role_Sets>
<DSD_Role_Sets>
 <DSD_Role_Set
 DSD_Role_Set_id = 'DSD1'
 DSD_cardinality = '2'>
  <DSD_Role>DBA</DSD_Role>
  <DSD_Role>Accountant
  </DSD_Role>
  <DSD_Role>Cashier</DSD_Role>
 </DSD_Role_Set>
<DSD_Role_Sets>
```

(b)

```
<permissions>
 <permission>
  <perm_id>P1</perm_id>
  <object_type>Cluster
  </object_type>
  <object_id>CL100
  </object_id>
  <operation>read
  </operation>
 </permission>
 <permission>
  <perm_id>P2</perm_id>
  <object_type>Schema
  </object_type>
  <object_id>XS101
  </object_id>
  <operation>all
  </operation>
 </permission>
 <permission>
  <perm_id>P3</perm_id>
  <object_type>Instance
  </object_type>
  <object_id>XI100
  </object_id>
  <operation>all
  </operation>
 </permission>
 <permission>
  <perm_id>P4</perm_id>
  <object_type>Element
  </object_type>
  <object_id>/EyeCareMedic
   alHistory/Patient/Name
  </object_id>
  <operation>navigate
  </operation>
 </permission>
</permissions>
```

(c)

Figure 2. XML instances. (a) XML user sheet, (b) XML role sheet for the Doctor and DBA roles, and (c) XML permission sheet, which defines permissions for objects and associated operations in a given system.

Here, *mand* indicates a mandatory attribute and *opt* indicates an optional one. The administrator enforces the specified requirements on the available attributes when it forms the attribute-value pairs.

The XML representation for the above credential information is an XML user sheet (XUS). Figure 2a shows an XUS instance. The max_roles tag indicates the maximum number of roles a user can have. Capturing the user's activity profile might require updating the user credentials dynamically.

Roles. The system administrator also creates roles. A role has an associated set of credentials that the users assigned to that role must satisfy. Figure 2b shows an XML role sheet. The XRS is an XML instance document describing the Doctor and DBA roles along with the corresponding static separation of duty (SSD) and dynamic separation of duty (DSD) role sets.

The role_name is a unique role identifier. The optional SSD_Role_Set_id and DSD_Role_Set_id tags refer to the set of roles that are in the static and dynamic separation of duty categories, respectively.[9] Each set has a cardinality attribute that gives the maximum number of roles it can assign to a user or that it can activate from the set. The optional junior and senior tags capture hierarchical relationships by referring to junior and senior roles.[9] The cardinality is the maximum number of users associated with a role at any time. The administrator can specify a cardinality to limit the number of users assigned to a role. However, if no cardinality is explicitly supplied, the number of users is assumed to be unlimited.

In Figure 2b, the Doctor role belongs to the SSDRoleSet identified by SSD1, with cardinality 1, so any user cannot be assigned to more than one role from within this set. Similarly, the DBA role belongs to the DSDRoleSet identified by DSD1, with cardinality 2, and so a user cannot activate more than two roles from within this set at once.

Permissions. Our specification defines the permissions for a given system in terms of objects and associated operations. The permission component usually consists of system-dependent operations such as read, write, delete, or modify. Figure 2c shows an XML permission sheet (XPS), which the system administrator creates to define the objects and corresponding operations in a given system.

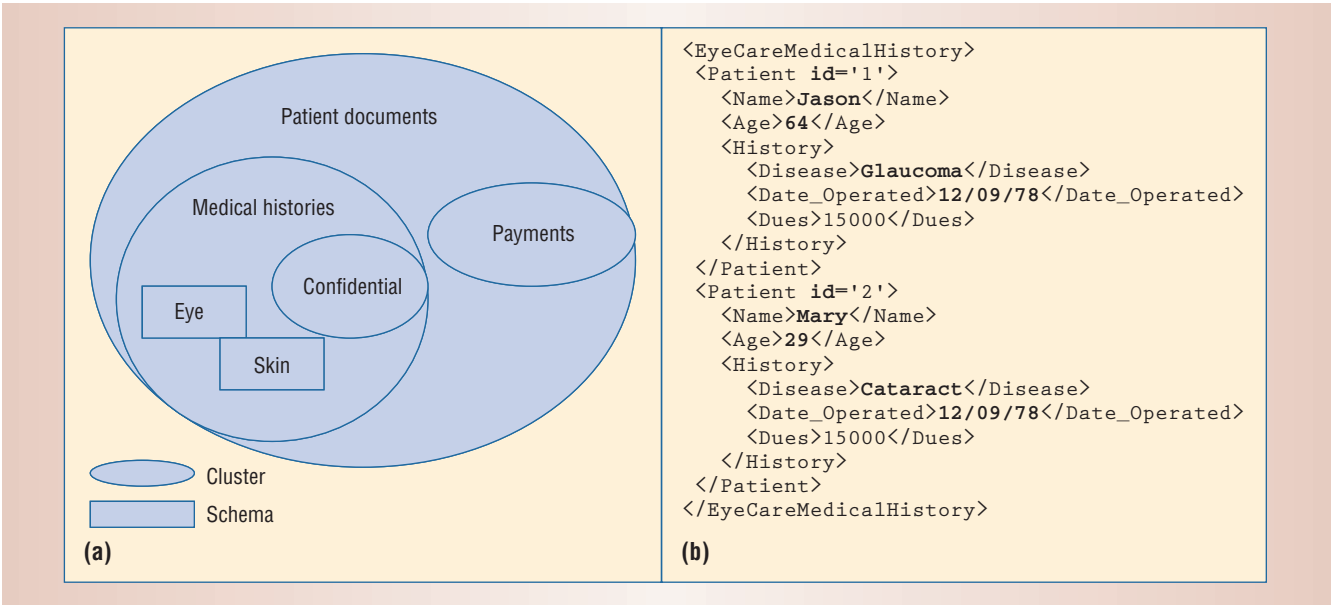

Figure 3. Permissions and their propagation options. (a) An object hierarchy of clusters and schemas; (b) an XML instance of eye care medical histories.

The perm_id is a unique permission identifier. An object can represent either a cluster, schema, instance document, or document element to which the system administrator assigns permission. Our specification introduces the notion of an object type to distinguish the associated resource. The system administrator provides IDs that identify clusters, schemas, and documents. When the accessed objects are elements within an XML document, the system uses XML Path Language (XPath) expressions to identify them. Having access privileges to a cluster implies having access to all schemas and instance documents within that cluster's scope, and having access privileges to a schema implies having access to all conforming instance documents.

For example, Figure 3a shows a healthcare Web service cluster hierarchy in which a user with access privileges on either the medical histories cluster or the eye schema could view all instances of eye care medical history, such as the one that Figure 3b shows. The associated operations define the extent of this access.

A permission can have a propagation option that indicates whether or not it propagates down the object hierarchy. Our specification allows the propagation options no_prop, first_level, and cascade.[1] In Figure 3a, a first_level propagation option on the patient documents cluster means that the user is authorized to view the documents within the two immediate descendant clusters, namely payments and medical histories—the latter including all instance documents conforming to the eye and skin schemas—but not within the confidential cluster.

Similarly, in Figure 3b, a user who has access privileges to a patient element could also view the contents of the corresponding history element if the permission offers a cascade propagation option. In general, if the permission does not explicitly specify the option, it is assumed to be no_prop, that is, there is no propagation. However, the administrator can specify a different propagation option at the time of permission-to-role assignment if a role demands sufficient privileges.

In Figure 2c, P1 identifies a permission that allows a read operation on all documents within the cluster's scope that CL100 identifies with the default propagation option. Similarly, P2 and P3 identify permissions that allow all operations on all document instances conforming to the schema that XS101 identifies and the document instance that XI100 identifies, respectively, with the default propagation option. P4 identifies a permission that allows the navigate operation on the XML name element, also with the default propagation option.

The associated XPath expression that identifies the element imposes a specific structure on the container document. In this case, the permission applies to the name element contained in the instance document in Figure 3b.

Policy administration

The policy administration process uses the information about users, roles, and permissions available from the corresponding XML documents. The system administrator uses these XML sheets to specify the policy base for the protected documents from criteria that system designers specify. The documents that the system generates in this phase include an XML user-to-role mapping (XURM) and an XML permission-to-role mapping (XPRM). Our model captures these mappings through XML schemas that describe the user-role and permission-role assignments. Keeping the user, role, and permission specifications separate from their mappings allows independent design and administration of the policy.

Once the system generates these XML documents, they constitute a part of the policy base. The information from the policy base then enforces the autho-

```
<xurm>
 <urm urm_id='URM1' />
  <role_name>Doctor</role_name>
  <cred_type>Nurse</cred_type>
  <conditions>
   <condition>
    <mode value='AND'>
     <predicate>
       <operation>gt</operation>
       <parameter1>level</parameter1>
       <parameter2>5</parameter2>
     </predicate>
     <predicate>
       <operation>lt</operation>
       <parameter1>age</parameter1>
       <parameter2>80</parameter2>
     </predicate>
    </mode>
   </condition>
  </conditions>
 </urm>
</xurm>
```

(a)

```
<xprm>
 <prm prm_id='PRM1'>
    <role_name>Eye_Doctor</role_name>
    <permissions>
     <perm_id>P1</perm_id>
    </permissions>
</prm>
<prm prm_id='PRM2'>
   <role_name>DBA</role_name>
   <permissions>
    <perm_id>P2</perm_id>
    <perm_id>P3</perm_id>
   </permissions>
</prm>
<prm prm_id='PRM3'>
    <role_name>Dispenser</role_name>
    <permissions>
     <perm_id>P4</perm_id>
    </permissions>
</prm>
</xprm>
```

(b)

Figure 4. Role and permission assignment. (a) XURM and (b) XPRM capture the user-to-role and permission-to-role mappings through XML schemas that describe assignment conditions on users and permissions.

rization constraints. More specifically, users have access to resources based on the roles that XURM assigns and the permissions that the XPRM specifies.

Figure 4a shows an XML instance document for mapping users to a role based on user credentials. Here, the access control processor parses and recognizes the condition part of the credential to evaluate the operation. This example associates a set of credentials to the Doctor role. It states that the administrator can assign all users with the Nurse credential type to the Doctor role only if level is greater than 5 and age is less than 80.

The associated XML schema can accommodate nested Boolean expressions as well, and a predicate within a condition expression can itself contain another condition.

Mapping permissions to corresponding roles reflects the policy specifications at the conceptual, schema, instance, and element levels. Implicitly, such an association generates a permission-role assignment. Our schema specifies these associations in an XPRM.

Conceptual-level access-control policies use roles related to XML document clusters. This leads to the schema specification for assigning permissions to XML objects that represent clusters. Figure 4b shows an instance of such a schema specification. Here, PRM1 identifies a mapping that associates the Eye_Doctor role with permission P1, which refers to the object cluster on the XPS that Figure 2c shows. In this case, an Eye_Doctor role is authorized to read all the documents within the cluster identified by cluster ID CL100. This cluster contains all information relevant to the eye care concept.

The system uses the same mechanism to implement schema, instance, and element level access control. For instance, the mapping that PRM2 identifies in Figure 4b associates the DBA role with permissions P2 and P3, which refer to a schema object and an instance document, respectively. From Figure 3c, permissions P2 and P3 authorize one to read/write/navigate all instance documents conforming to the schema that XS101 identifies and also the instance document that XI100 identifies. Similarly, PRM3 identifies a mapping that associates the Dispenser role with permission P4, which refers to a Name element (in some XML instance document) that an XPath expression identifies. Thus, the Dispenser role is authorized only to navigate the Name element in all conforming instance documents, such as the document in Figure 3b.

SOFTWARE ARCHITECTURE

Figure 5 depicts a proposed software architecture for a single-enterprise Web-service-enabled application that disseminates secure documents. The proposed architecture meets all the RBAC functional specifications of the NIST standard.[9]

Document composition module

The XML document composition module (XDCM) provides the main graphical interface for composing XML schemas for RBAC elements and policy administration documents. The same interface composes both sets of documents, which the policy base stores. This module provides all the administrative functions as part of RBAC functional specifications.

Access control module

The access control module (ACM), the architecture's key component, interfaces with various other functional modules and information repositories to extract relevant information while mak-

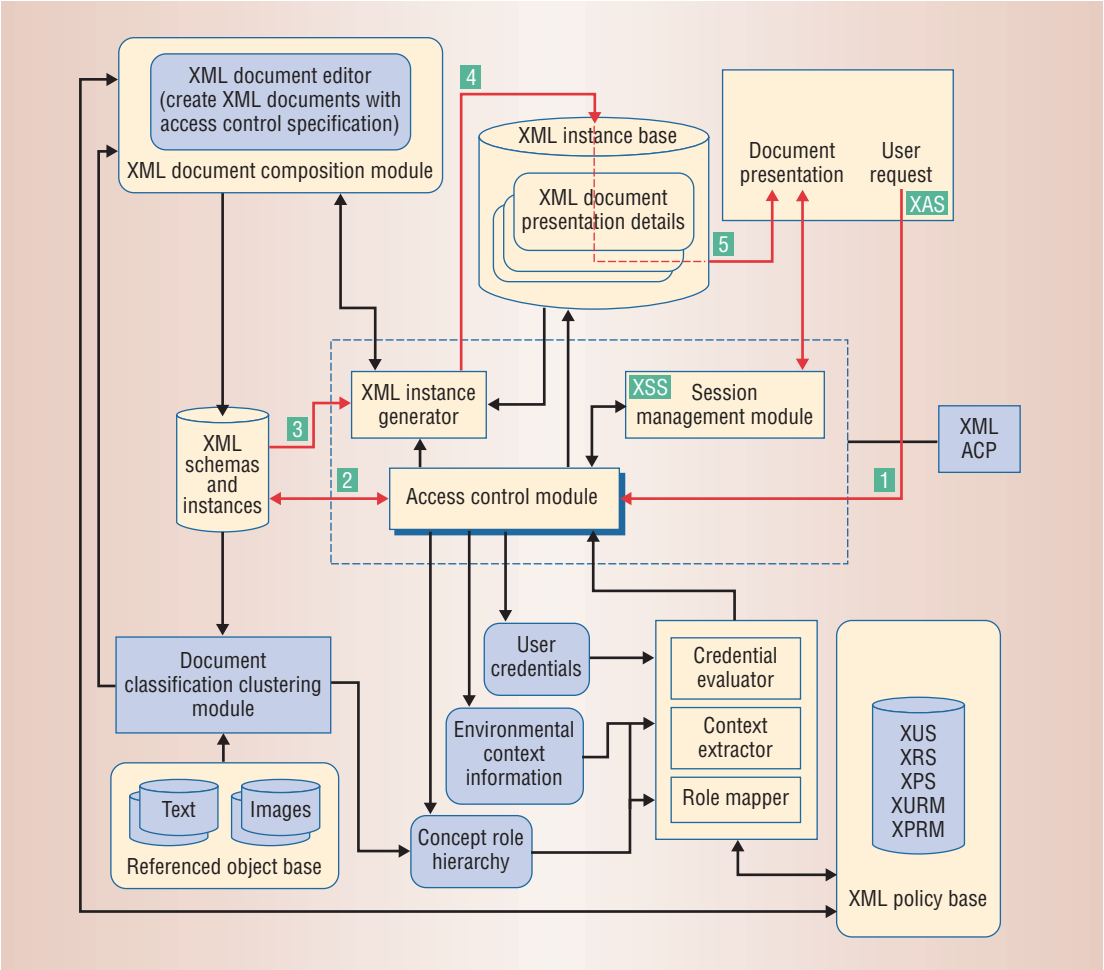

Figure 5. Framework for a single-enterprise Web-service-enabled application that disseminates secure documents. The numbered arrows show the steps in the authorization process.

ing authorization decisions. The ACM extracts the policy information from the policy base and works closely with the XML instance generator (XIG) module to enforce authorization constraints.

The XIG module gets information from the ACM about the access permissions allowed on XML documents associated with an access request and generates XML views accordingly in response to that request. The XML instance base (XIB) caches these views. XIG can simply be an extension of an XML document processor. Along with the session management module (SMM), the ACM manages the supporting system functions listed in the NIST RBAC functional specifications.

Session management module

The SMM monitors session activities, capturing relevant, dynamic context information that updates user credentials and thus might affect future access control decisions. The system maintains this information in an XML session sheet (XSS) and communicates it to the ACM. The ACM then updates the user credential information in the policy base. ACM, XIG, and SMM together form the XML access control processor.

SMM's flexible session management capability is particularly significant. For example, in a mobile Web services environment, a user could start a session that might later be suspended by a user request or due to a handoff.

Here, SMM must store the current context information to support the user's reconnection. By the time the user requests reconnection, some context conditions may have changed. SMM must take these changes into account when granting reconnection requests, possibly with a new set of authorizations. SMM's ability to capture dynamic context information allows the system to incorporate this feature.

Document classification module

The document classification/clustering module (DCM) manages classification and clustering of all documents. It organizes the concept clusters hierarchically. The role mapper associates roles with concepts and generates the XRSs for these roles and their hierarchy. This module provides functionality to add or delete clusters, as well as to create virtual clusters based on a new set of user credentials. Additionally, this module also handles the classification of new documents entering the source. The module can assign a new document to an existing cluster based on its conformance to the schemas that compose the cluster. Previous work proposed similar approaches for document classification.[1]

Credential evaluator, context extractor, and role mapper

The credential evaluator module (CEM) evaluates the credentials the ACM presents. It also assigns the user to an existing credential type or creates a new credential type if the user credentials do not match any existing credential specification. With the help of the role mapper, the CEM maps the credentials to a role using the assigned credential type. The context extractor evaluates the contextual information the ACM provides and sends back relevant information for access decision after consulting the policy base.

Repositories

The referenced object base constitutes the physical objects present in the system from which the system administrator composes the XML documents. The XML schemas and instances contain actual XML sources to which the user will be requesting access. The XML policy base contains all policy-related XML documents that XDCM composes. The system can retrieve the information content necessary for all review functions, as stipulated by the RBAC functional specifications, from the policy base, with support from SMM and role hierarchy components as necessary.

The numbered arrows in Figure 5 show the steps involved in the authorization process. In step one, the user sends a request to the ACM in the form of an XML access sheet (XAS) that contains the user's login information and a list of access requests. In step two, the ACM generates a set of authorizations based on the policy after identifying the XML sources. In step three, the XIG generates the document instance according to the authorizations the ACM generates. In step four, the XIG applies the presentation formats, and in step five the XIB presents the authorized document view. Other arrows indicate the retrieval of information needed for access control decisions.

Although our framework and the corresponding system architecture act as a policy specification and enforcement mechanism within a Web services environment, users can extend the framework to incorporate a more complete set of features from the Web services security specifications. To do so, the framework employs any XML-based standard messaging protocol, such as SOAP, between a set of cascaded modules, each implementing a specific set of specifications, such as WS-Security or WS-Privacy.

Our proposed mechanism ties into the specifications at the WS-Policy level. The end user requesting access to the target system would need to interact only with a top-level interface, and the user credentials and queries would be passed as SOAP-encoded XML messages between the various modules. The extension mechanism will likely be investigated in the future.

VALIDATION AND IMPLEMENTATION

We validated our proposed model in two steps: We used XML Schema to check the policy documents for conformance with the model, and we used the XML access control processor to evaluate the XML documents for conformance to domain-specific constraints. A preliminary version of our proposed software architecture has been implemented and tested using a Java-based XML-enabled application.[8] The prototype includes a policy validation module that verifies and validates all the XML files that compose the policy base. An XML parser module maps the XML syntax to lower-level language constructs and supplies the policy information to the ACM. Access requests are received and authorization decisions are returned as HTTP requests over the Web.

A key feature of our framework is that it separates language schemas, which allows specifying multiple components of the access control policy independently and in an interoperable manner. The implementation of our model shows that our software architecture can be applied to a single-enterprise Web-service-enabled application that disseminates secure documents.

We are now working on a scheme to extend this framework to incorporate a more complete set of features from the Web services security specifications. In other research, we plan to extend our XML specification language to allow specification of policies in a distributed, multiple-enterprise environment. ■

Acknowledgments

Portions of this work have been supported by the sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University and the National Science Foundation under NSF grant no. IIS-0242419.

References

1. E. Bertino et al., "Controlled Access and Dissemination of XML Documents," *Proc. Workshop Web*

Information and Data Management, ACM Press, 1999, pp. 22-27.
2. E. Damiani et al., "A Fine-Grained Access Control System for XML Documents," *ACM Trans. Information and System Security (TISSEC)*, vol. 5, no. 2, ACM Press, 2002, pp. 169-202.
3. J.Y. Chung, K.J. Lin, and R.G. Mathieu, "Guest Editor's Introduction—Web Services Computing: Advancing Software Interoperability," *Computer*, Oct. 2003, pp. 35-37.
4. J.B.D. Joshi et al., "Security Models for Web-Based Applications," *Comm. ACM*, Feb. 2001, pp. 38-72.
5. R.S. Sandhu et al., "Role-Based Access Control Models," *Computer*, Feb. 1996, pp. 38-47.
6. S.L. Osborn, R. Sandhu, and Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Trans. Information and System Security* (TISSEC), vol. 3, no. 2, ACM Press, 2000, pp. 85-106.
7. S. Hada and M. Kudo, "XML Access Control Language: Provisional Authorization for XML Documents," 16 Oct. 2000, Tokyo Research Laboratory, IBM Research.
8. R. Bhatti et al., *Access Control in Dynamic XML-Based Web Services with X-RBAC*, CERIAS tech. report 2003-36.
9. D.F. Ferraiolo et al., "Proposed NIST Standard for Role-Based Access Control," *ACM Trans. Information and System Security* (*TISSEC*), vol. 4, no. 3, ACM Press, 2001, pp. 224-274.
10. H. Chen, "A Machine Learning Approach to Document Retrieval: An Overview and an Experiment," *Proc. 27th Hawaii Int'l Conf. System Sciences*, vol. 3, IEEE CS Press, 1994, pp. 631-640.

Rafae Bhatti *is a PhD candidate in electrical and computer engineering at Purdue University. His research interests include information systems security and distributed systems. Bhatti received an MS in electrical and computer engineering from Purdue University. He is a student member of the IEEE. Contact him at rafae@purdue.edu.*

Elisa Bertino *is a professor of computer science and director of research at CERIAS at Purdue University. Her research interests include security, privacy, and database systems. Bertino received a PhD in computer science from the University of Pisa, Italy. Contact her at bertino@cs.purdue.edu.*

Arif Ghafoor *is a professor of electrical and computer engineering at Purdue University. His research interests include multimedia systems, information security, distributed systems, and broadband multimedia networking. Ghafoor received a PhD in electrical engineering from Columbia University. Contact him at ghafoor@ecn.purdue.edu.*

James B.D. Joshi *is an assistant professor in the Department of Information Sciences and Telecommunications at the University of Pittsburgh. His research interests include information systems security, distributed systems, multimedia systems, and systems survivability. Joshi received a PhD in computer engineering from Purdue University. Contact him at jjoshi@mail.sis.pitt.edu.*