Specification and Validation of Enterprise Access Control Data for Conformance to Model and Policy Constraints

Ramaswamy Chandramouli
Computer Security Division, ITL, NIST
Gaithersburg, MD 20899, USA

ABSTRACT

The effectiveness of an enterprise access control framework depends upon the integrity of the various components or the building blocks used in that framework. The essential components of that framework are: (a) an Enterprise Access Control Model (b) a Validation mechanism to verify the enterprise access control data developed based on that model, for conformance to the model as well as domain-specific policy constraints and (c) a mechanism to map the enterprise access control data into formats required by native access enforcement mechanisms in the heterogeneous application systems in the enterprise. In this paper we chose the Role-based Access Control Model (RBAC) as a candidate for the enterprise access control model. We develop an XML Schema of an RBAC Model for a specific enterprise context and demonstrate the use of schema features to specify structural and some rudimentary domain constraints. We then annotate that XML Schema of an Enterprise RBAC Model to demonstrate specification and enforcement of some important domain-specific policy constraint using the Schematron language.

Keywords: Role-based Access Control, XML Schema, Policy Constraints, and Enterprise Access Control Data

1. INTRODUCTION

The effectiveness of an enterprise access control framework depends upon the integrity of the various components or the building blocks used in that framework. An enterprise access control framework generally consists of the following components.

(a) EACF-C1: An Enterprise Access Control Model that provides a generic set of entities and the structure needed to express the access restrictions for all the resources in an enterprise irrespective of the application system in which those resources are hosted.

(b) EACF-C2: A means to validate whether the access control data developed based on the enterprise access control model does indeed conform to the structural requirements of the model as well as satisfy the domain's policy constraints.

(c) EACF-C3: A defined way of interpreting the access control data under the enterprise access control model and converting it into formats that the native access control mechanisms in the heterogeneous application systems in the enterprise use to enforce access restrictions.

On analyzing the technologies needed for building the above components, we find that the Role-based Access Control Model (RBAC) [1,2] has emerged as a leading candidate for the enterprise access control model. The reasons for this emergence are that RBAC models have been shown to be policy-neutral and at the same time provide support for arbitrary organization-specific access control policies, the essential characteristics for any enterprise access control model. Further the policy support capability of RBAC models has been shown to adequately simulate traditional access control models like Discretionary Access Control Model (DAC) and the Mandatory Access Control Model (MAC). Hence our choice for enterprise access control model is RBAC.

Building the second component of an enterprise access control framework (i.e. EACF-C2) requires a structured language framework to represent the enterprise access control data and its underlying model. Recently XML [3] and XML Schema [4] specification Languages have been gaining acceptance as standards for representing, interchanging and presenting both meta-data and complex content models in a platform independent fashion. Specifically the XML Schema provides a very extensible means for specifying document structures through a comprehensive type definition language. Hence it is the best candidate for a linguistic framework that is needed to express an access control model (which in our case is RBAC) that embodies multiple policy requirements. The associated access control data for a given enterprise domain can then be encoded in an XML document and the conformance of that data to the enterprise access control model can be obtained by validating the XML document against the XML Schema that represents the enterprise access control model (we will refer to this as XML Schema for Enterprise RBAC Model) through a type of software called XML Parsers. These XML Parsers are based on standard application programming interfaces such as Document Object Model (DOM) [5]. These parser libraries implemented in various procedural languages enable an application program written in the corresponding procedural language to create, maintain

and retrieve XML encoded data. Hence we have a programmable framework to extract enterprise access control data in XML documents, properly interpret them and map them to the native data formats for access control mechanisms present in heterogeneous application systems within the enterprise. This satisfies the functionality required for the third component (EACF-C3) of our enterprise access control framework.

The technologies proposed in our framework so far still fall short of addressing all the integrity issues associated with enterprise access control data. Integrity issues for enterprise access control data are an outcome of the fact that the data represent an instance of an access control model and contains domain specific data. Hence addressing integrity issues in enterprise access control data involves verification of the enterprise access control data for both structural constraints (conformance to the components, relationships and constraints inherent in the access control model) as well as domain constraints (conformance to rules and properties dictated by the enterprise domain). Through the XML Schema for Enterprise RBAC Model, we can specify structural constraints that the enterprise access control data must satisfy as well as certain rudimentary domain constraints through the feature for specifying complex data types. However, studies have shown [6,7] that access control information for large enterprises is highly context-sensitive and hence schemas for representing that information require logic mechanisms to express rules and properties involving their contents.

One approach that has been adopted to represent domain constraints is to annotate an XML Schema that has been used for representing a model for a domain, with ontological information regarding the domain using pattern based languages such as RDF [8] and Schematron [9]. In this paper we have annotated the XML Schema for Enterprise RBAC Model with Schematron constraints that specify rules that that the access control data pertaining to the chosen enterprise domain has to satisfy.

Based on the approach outlined above, our first task is to develop an XML Schema for an Enterprise RBAC Model. This is the topic for section 2 of this paper. In section 3 we describe as to how the structural constraints and some rudimentary domain constraints have been represented in the XML Schema. Section 4 discusses the domain constraints and provides examples of how such constraints may be incorporated in the XML Schema through Schematron language annotations. In Section 5 we briefly describe other XML-based approaches for access control data representation as well as for constraint validation. Section 6 points out the scope for further work in this area.

2. AN XML SCHEMA FOR AN ENTERPRISE RBAC MODEL

The enterprise environment we have chosen for the enterprise RBAC model is a commercial bank. Before we describe the steps involved in developing an XML Schema for an Enterprise RBAC Model for a bank environment, we provide brief background information on RBAC and XML Schema respectively.

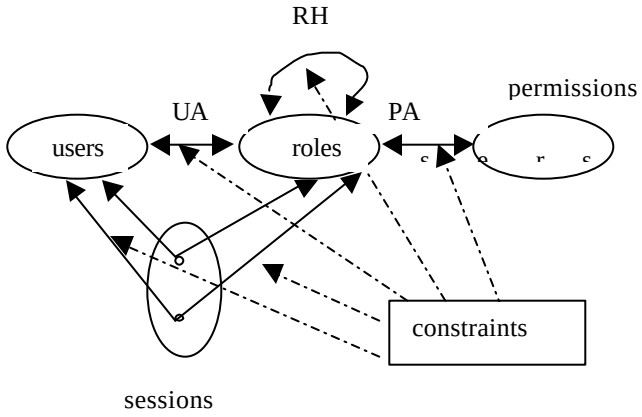
2.1 Role-based Access Control Model (RBAC)

The Role-based Access Control Model (RBAC) provides a generalized approach for representation of many types of access control policies (each describable only using a specific access control model) through the abstraction concept of roles. Since there are many RBAC models proposed in the literature, we use as our reference the NIST Standard RBAC Model [10]. This RBAC model has four main components – users, roles, permissions and sessions. Roles generally represent organizational functions (e.g. Teller in a bank). Users are assigned to roles and permissions are assigned to roles as well. Users derive all their permissions by virtue of their role memberships. Users interact with the system through sessions and roles are assigned to a particular sessions as well. Now the interactions among these four components of the RBAC model results in the following relationships:

(a) User-Role Assignments (UA)
(b) Role Hierarchies (RH)
(c) Role Permission Assignments (PA)
(d) User-Session Assignment (US)
(e) Role-Session Assignment (RS).

A schematic diagram of our reference RBAC model is given in Fig 1.

Fig 1 – Relationships in a RBAC Model



2.2 XML Schema

The XML Schema [4] is a language specification that has been issued by the World Wide Web Consortium (W3C) for description of data (or metadata). In this metadata or schema language, an entity or a component (whether it is abstract or physical) is represented as an element with defined place holders to specify the name, the attributes, the minimum and maximum number of occurrences and the data type of the element. The XML Schema provides support for rich data typing (both built-in and user defined). In summary, the XML Schema language provides good support for explicit structural,

cardinality and data typing constraints but not for representing domain-specific (or content-based) constraints.

2.3 Modeling RBAC Components in XML Schema

Our motivation in describing the process of developing an XML Schema for an Enterprise RBAC Model (for a bank enterprise) is to illustrate the concepts involved in the representation of an enterprise access model and using that framework for verification of structural constraints on the access control data that conform to the enterprise access control model. In this paper, we describe the development of XML Schemas for specification of the enterprise RBAC model for a commercial banking environment. We restrict our illustration to modeling the User, Role and User-Role Assignments (UA) (since the other concepts of the RBAC model are conceptually similar).

In the XML schema, an RBAC Model component is represented as an element with an associated data type. Hence the concept 'User' in our Enterprise RBAC Model is represented as:

```
<xs:element name="user"    type="userType"/>
<xs:complexType name="userType">
<xs:attribute name="userID" type="xs:ID"
use="required"/>
<xs:attribute name="fullname" type="xs:string"
use="optional"/>
</xs:complexType >
```

The above definition of the data type 'userType' means that a user is represented as having two attributes 'userID' and 'fullname' with the former declared as a mandatory attribute and the latter declared as an optional attribute. Please note that the data type for 'userID' attribute is designated as 'xs:ID' which implies that the value for 'userID' attribute must be unique and hence no duplicates are allowed.

Similarly we represent the component 'Role' of our Enterprise RBAC model and its associated data type with the following declarations in the XML schema.

```
<xs:element name="role"  type="roleType"/>
<xs:complexType name="roleType">
<xs:attribute name="roleID" type="xs:ID"
use="required"/>
<xs:attribute name="rolename" type="validRole"
use="required"/>
<xs:attribute name="cardinality" type="roleLimit"
use="optional"/>
</xs:complexType>
```

To complete our definition of role component, we need to define the data types "validRole" and "roleLimit". The data type definition of "validRole" lists the set of permissible role names in the bank enterprise while that for the "roleLimit" is used to specify a number that stands for the minimum and maximum number of users that can be assigned to that role.

```
<xs:simpleType name="validRole">
    <xs:restriction base="xs:string">
<xs:enumeration value="BranchManager"/>
<xs:enumeration value="Customer_Service_Rep"/>
<xs:enumeration value="SD_Vault_Officer"/>
<xs:enumeration value="Loan_Officer"/>
<xs:enumeration value="Accounting_Manager"/>
<xs:enumeration value="Internal_Auditor"/>
<xs:enumeration value="Teller"/>
<xs:enumeration value="Accountant"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="roleLimit">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>
```

We now provide the XML Schema representation for the User-Role Assignment (UA) relationship of our Enterprise RBAC Model .

```
<xs:element name="UserRoleAssignment"
            type="URAType"/>
<xs:complexType name="URAType">
    <xs:sequence>
        <xs:element name="user" type="xs:IDREF"
                    maxOccurs="10"/>
    </xs:sequence>
    <xs:attribute name="role" type="xs:IDREF"
            use="required"/>
</xs:complexType>
```

Finally the fact that the entire RBAC model for the bank enterprise domain is made up of the components User, Role and User-Role Assignment relationship is represented using a root element called the 'BANK_RBAC_Model' in the XML schema which contains elements, that represent the components User, Role and User-Role Assignment relationship, as sub-elements.

```
  <xs:element name="Bank_RBAC_Model"
              type="BankRBACModelType"/>
<xs:complexType
        name=" BankRBACModelType">
    <xs:sequence>
      <xs:element ref="user" maxOccurs="unbounded"/>
      <xs:element ref="role" maxOccurs="unbounded"/>
      <xs:element ref="UserRoleAssignment"
                          maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Observe that some of the elements specified above do not have a name (like other element definitions we have seen before) but refers to the already defined elements through the value specified in the 'ref' attribute. The above XML Schema definition was verified to be syntactically correct using the XML Schema Validator tool – XML Spy [11].

2.4 Encoding the enterprise access control data in XML

Now that we have developed an XML Schema for the Enterprise RBAC Model, we now encode the enterprise access control data in an XML document whose tag structure should correspond to the element definitions in the XML Schema.

We represent a sample set of users (by providing instances of the 'user' element in XML schema) as given below:

```
<user userID="DrayJ" fullname="Jim Dray"/>
<user userID="GranceT" fullname="Tim Grance"/>
<user userID="VincentH" fullname="Vincent Hu"/>
```

A sample set of encodings for role instances is:

```
<role roleID="BRM" rolename="BranchManager"
                    cardinality="1"/>
<role roleID="CSR"
rolename="Customer_Service_Rep"
                    cardinality="3"/>
<role roleID="SDV" rolename="SD_Vault_Officer"
                    cardinality="2"/>
```

A sample set of User-Role Assignments is:

```
<UserRoleAssignment role='BRM'>
        <user>GranceT</user>
        <user>JansenW</user>
</UserRoleAssignment>
<UserRoleAssignment role='CSR'>
        <user>Sheila</user>
        <user>TomK</user>
</UserRoleAssignment>
```

3. CONSTRAINTS EXPRESSED USING THE XML SCHEMA

Let us now review the structural constraints and some rudimentary domain-specific constraints that we have been able to specify in our XML Schema of the Enterprise RBAC Model. (Please note that here we include only constraints that can be validated by an XML Schema parser).

3.1 Structural Constraints represented using the XML Schema

Based on the discussion of the XML Schema specification of the Enterprise RBAC Model, we can summarize the structural constraints we have been able to express.

(a) Specification of mandatory and optional attributes

(b) Identification of attribute whose values must be unique (no duplicates allowed)

(c) Cardinality constraints showing the number of times (instances) an RBAC model component can occur in the XML document that contains the access control data.

3.2 Domain Constraints represented using the XML Schema

(a) The role names that occur in an XML encoded access control data document should be one of the valid names specified in the XML Schema (through the validRole data type).

4. SPECIFICATION OF DOMAIN-SPECIFIC POLICY CONSTRAINTS

The specification of an RBAC model for an enterprise can only be useful if it can capture many domain-specific policy constraints. Since the XML-schema representation has limitations in achieving this goal, we annotate the XML schema specification of the Bank-Enterprise RBAC model by expressing policy constraints using the schematron constraint specification language [9]. We also illustrate the verification of our enterprise access control data (encoded in XML) for conformance to these constraints and the generation of meaningful violation messages using the schematron validation tool [12].

In a schematron constraint definition, constraints are defined using the following tags:

(a) a 'rule' tag to define the context (in terms of the XML schema element) for the constraint and

(b) one or more 'assert' tags: Each 'assert' tag contains the Boolean expression for the property that each of the instances of the element (named in the context) has to satisfy. Any violation of the property will be flagged off as an error.

(c)one or more 'report' tags: Each 'report' tag contains the Boolean expression for the property that each of the instances of the element (named in the context) should not satisfy. Any instance where the property is satisfied will be flagged off as an error.

(d)A set of 'diagnostic' tags: Each of these provides information on the violating data.

(e) The above tags are also enclosed within a named 'pattern' tag.

With the above primer on Schematron, we now illustrate the specification of some important policy constraints that govern the access control requirements for the bank enterprise environment.

<u>Constraint 1:</u> The cardinality limit (the maximum number of users that can be assigned) specified in the role definition for a role should not be violated in the actual user assignments for that role.

The role definition for the Branch Manager role(roleID = 'BRM') in our XML encoded access control data file is as follows:

<role roleID="BRM" rolename="BranchManager" cardinality="1"/>

The reference to the above data through the XML Schema components forms the context. The context therefore is a role instance definition whose roleID attribute is 'BRM" (for Branch Manager). This context is expressed in schematron as:

```
<sch:rule
context="Bank_RBAC_Model/role[@roleID='BRM']">
```

The assertion to be made in this context is that in the corresponding User-Role Assignment (where the @role='BRM'), the count of the number of users should not exceed the number specified through the cardinality attribute (@cardinality = 1). The assertion and the corresponding diagnostic messages expressed in schematron through the assert and diagnostic tags respectively are given below:

```
<sch:assert test = "./@cardinality >=
count(../UserRoleAssignment/user[../@role = 'BRM']) "
diagnostics="Cardinality_Exceeded">Cardinality for the
role exceeded
</sch:assert>

<sch:diagnostics>
   <sch:diagnostic id="Cardinality_Exceeded">The
actual number of users assigned is: <sch:value-of
select="count(../UserRoleAssignment/user[../@role =
'BRM'])"/> while cardinality limit is: <sch:value-of
select="./@cardinality"/>
    </sch:diagnostic
 </sch:diagnostics>
```

The actual data in our access control data file is:

```
<UserRoleAssignment role='BRM'>
        <user>GranceT</user>
        <user>JansenW</user>
</UserRoleAssignment>
```

The schematron validator therefore generated the following error message:

From pattern "Checking for Role Cardinality":
Assertion fails: "Cardinality for the role exceeded" at
/Bank_RBAC_Model[1]/role[1]
<role roleID="BRM" rolename="BranchManager" cardinality="1">...</> The actual number of users assigned is: 2 while cardinality limit is: 1

<u>Constraint 2:</u> A user assigned to the Internal Auditor role (@role='AUD') should not be assigned to the Accountant role (@role='ACC') since Internal Auditor and Accountant are conflicting roles.

The context, the assertion and the diagnostic tags used to specify the above constraint is as follows:

```
<sch:rule
context="Bank_RBAC_Model/UserRoleAssignment[@rol
e ='AUD']/user">

 <sch:assert test= "not(text() =
(../../UserRoleAssignment[@role='ACC']/user/text() ))"
diagnostics="SOD_AUD">There should not a common
user in Audit and Accounting roles.
 </sch:assert>
 <sch:diagnostics>
      <sch:diagnostic id="SOD_AUD">The violating
assignment  is made for user: <sch:value-of
select="text()"/>
      </sch:diagnostic>
  </sch:diagnostics>
  </sch:rule>
```

For our access control data, the schematron validator generated the following message:

From pattern "Checking for Separation of Duty":
Assertion fails: "There should not a common user in Audit and Accounting roles." at

/Bank_RBAC_Model[1]/UserRoleAssignment[6]/user[1]
<user>...</> The violating assignment is made for user: VincentH

<u>Constraint 3:</u> Users John Wack (user/text() ='JohnW') and Susan Wack (user/text() = 'SusanW') should not be assigned to the same role (whatever be the role) since they have spousal relationship.

The schematron description of the above constraint is:

```
<sch:pattern name="Checking for Conflicting Users">
  <sch:rule

context="Bank_RBAC_Model/UserRoleAssignment">
    <sch:assert test="2 > count (user [text () =
'JohnW'])
                    + count(user [text() = 'SusanW'])"
        diagnostics="Wack_Violate">John Wack and
     Susan Wack should not be assigned to the same role
    </sch:assert>
    <sch:diagnostics>
      <sch:diagnostic id="Wack_Violate">The violating
          assignment is for the role: <sch:value-of
          select="@role"/>
       </sch:diagnostic>
      </sch:diagnostics>
  </sch:rule>
 </sch:pattern>
```

<u>Constraint 4:</u> Every user assigned to Safe Deposit Vault role (@role='SDV') should already be assigned to Customer Service Representative role (@role='CSD').

The Schematron syntax for the above constraint is:

```
<sch:pattern name="Checking for Dependent Role
                    Assignments">
  <sch:rule
context="Bank_RBAC_Model/UserRoleAssignment[@rol
e='SDV']/user">
    <sch:assert test="text() =
(../../UserRoleAssignment[@role='CSR']/user/text())"
diagnostics="SDV_CSR_Depend">A user assigned to
SDV must already be assigned to CSR role
    </sch:assert>
    <sch:diagnostics>
      <sch:diagnostic id="SDV_CSR_Depend">The
following user is assigned to SDV role but not to CSR
role: <sch:value-of select="text()"/>
     </sch:diagnostic>
    </sch:diagnostics>
  </sch:rule>
 </sch:pattern>
```

Constraint 5: The user Tom (user/text()= 'TomK') should not be assigned more than two roles.

Schematron expresses this as follows:

```
<sch:pattern name="Checking for limit on Tom's Assignments">
  <sch:rule context="Bank_RBAC_Model">
    <sch:assert test="3 > count(UserRoleAssignment[user/text()='TomK'])" diagnostics="Tom_Limit">Tom should be assigned a maximum of 2 roles
    </sch:assert>
   <sch:diagnostics>
     <sch:diagnostic id="Tom_Limit">The actual number of roles assigned to Tom is: <sch:value-of select="count(UserRoleAssignment[user/text()='TomK'])"/>
     </sch:diagnostic>
   </sch:diagnostics>
  </sch:rule
 </sch:pattern>
```

5. RELATED WORK

There are many efforts underway to develop XML-based frameworks for specification of access control information and each of them have limitations with respect to the type of policy constraints they can specify. The OASIS XACML [13] and IBM's XACL [14] are access control policy specification frameworks that are mainly geared towards securing XML documents and they do not provide support for representing traditional access controls such as DAC or MAC or enterprise access controls such as RBAC. Smith and Deng [15] have developed a DTD Schema of an RBAC Model and have left the verification of domain constraints to a separate administrative API. Further DTD Schemas in general have limitations with respect to representation of even structural constraints (e.g. number of occurrences of an element) for an RBAC model and they cannot be used for representation of even rudimentary domain-specific policy constraints.

6. SCOPE FOR FURTHER INTEGRITY IMPROVEMENT

We have illustrated an approach to validate enterprise access control data for satisfaction of both structural and domain-specific policy constraints by annotating an XML Schema of an Enterprise Access Control Model with Schematron language constraints. We intend developing mapping schemas that will translate the enterprise RBAC model elements to the actual access control elements in the various target platforms (like permission-bits of Unix and groups & ACLs of Windows NT platform) and then verify the integrity of the mapped access control data. This process will provide the integrity check for the class of software called "Agent Software" that are being increasingly deployed to translate enterprise access control data to various target platforms throughout the enterprise.

7. REFERENCES

[1] D.Ferraiolo, J.Cugini, and D.R.Kuhn. "Role Based Access Control (RBAC): Features and Motivations" Proc. 11th Annual Computer Security Applications Conference, December 1995.

[2] R.S. Sandhu, E.J.Coyne, H.L.Feinstein and C.E.Youman. "Role Based Access Control Models" IEEE Computer, vol 29, Num 2, February 1996, p38-47.

[3] XML 1.0, W3C Recommendation Feb '98, http://www.w3.org/XML/

[4] XML Schema Part 0: Primer W3C Recommendation, 2 May 2001 http://www.w3.org/TR/xmlschema-0/

[5] Document Object Model Technical Reports, http://www/w3.org/DOM/DOMTR

[6] R.Chandramouli, "Application of XML Tools for Enterprise-wide RBAC Implementation Tasks", Proc. Of 5th ACM workshop on Role-based Access Control, July 2000, Berlin, Germany.

[7] A.Schaad, "Role-based Access Control system of a European Bank: A Case Study and Discussion", Proc. Of 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001), Chantilly, VA, USA.

[8] Resource Description Framework (RDF), http://www.w3.org/RDF/

[9] Schematron - Pattern-based schema language, http://www.ascc.net/xml/resource/schematron/schematron.html

[10] D.Ferraiolo, R.Sandhu, S.Gavrila, D.R.Kuhn and R.Chandramouli, "Proposed NIST Standard for Role-based Access Control", ACM Trans. Inf.Syst.Security, Vol 4, Aug 2001, pp 224-274.

[11] http://www.xmlspy.com/download.html

[12] http://www.topologi.com/

[13] T. Moses, "The OASIS XACML Language Proposal", http://www/oasisopen.org/committees/xacml/docs

[14] M Kudo, S.Hada, "XML Access Control", "http://www.trl.ibm.com/projects/xml/xacl/xmlac-proposal.html, Oct 2000.

[15] Nathan N. Vuong , Geoffrey S. Smith , Yi Deng, Managing security policies in a distributed environment using extensible markup language (XML), Proceedings of the 16th ACM SAC2001 symposium on Applied computing March 2001, Las Vegas, Nevada, United States