X-GTRBAC: An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control

RAFAE BHATTI, ARIF GHAFOOR, and ELISA BERTINO
Purdue University
and
JAMES B. D. JOSHI
University of Pittsburgh

Modern day enterprises exhibit a growing trend toward adoption of enterprise computing services for efficient resource utilization, scalability, and flexibility. These environments are characterized by heterogeneous, distributed computing systems exchanging enormous volumes of time-critical data with varying levels of access control in a dynamic business environment. The enterprises are thus faced with significant challenges as they endeavor to achieve their primary goals, and simultaneously ensure enterprise-wide secure interoperation among the various collaborating entities. Key among these challenges are providing effective mechanism for enforcement of enterprise policy across distributed domains, ensuring secure content-based access to enterprise resources at all user levels, and allowing the specification of temporal and nontemporal context conditions to support fine-grained dynamic access control. In this paper, we investigate these challenges, and present X-GTRBAC, an XML-based GTRBAC policy specification language and its implementation for enforcing enterprise-wide access control. Our specification language is based on the GTRBAC model that incorporates the content- and context-aware dynamic access control requirements of an enterprise. An X-GTRBAC system has been implemented as a Java application. We discuss the salient features of the specification language, and present the software architecture of our system. A comprehensive example is included to discuss and motivate the applicability of the X-GTRBAC framework to a generic enterprise environment. An application level interface for implementing the policy in the X-GTRBAC system is also provided to consolidate the ideas presented in the paper.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.2.1 [**Software Engineering**]: Requirements/Specifications; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*

General Terms: Design, Security, Theory, Management

Additional Key Words and Phrases: XML, role-based access control, secure enterprises

Authors' addresses: R. Bhatti and A. Ghafoor, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907; email: rafae@purdue.edu; E. Bertino, Department of Computer Sciences and CERIAS, Purdue University, West Lafayette, IN 47907; J. B. D. Joshi, School of Information Sciences, University of Pittsburgh, Pittsburgh, PA.

1. INTRODUCTION

The dynamic economic and technical environment surrounding modern day business operations requires today's enterprises to strive evermore to stay competitive in the marketplace [Kern 2002]. The fast paced advancements in information technology infrastructure has placed growing demands on the enterprise to rise above the competition by making its operations more expandable, responsive to external factors, and, on top of all, secure. This has motivated many enterprises to adopt enterprise computing (EC) services for efficient resource utilization, scalability and flexibility [Java Commerce]. Such enterprises are also known as computer-integrated enterprise (CIE). The CIE is characterized by heterogeneous, distributed computing systems exchanging enormous volumes of time-critical data, with varying levels of access control, through the use of EC technology. The EC framework highlights the durability and maintainability of the enterprise assets, and emphasizes upon the scalability and flexibility of system infrastructure to allow it to evolve with time. While adopting such strategies is vital to the success of enterprise's business operations, it poses some serious challenges in terms of ensuring an enterprise-wide secure interoperation among the various collaborating entities.

These challenges belong to various domains within the CIE, and each needs to be addressed in order to achieve the overall enterprise goals. The access control policy of a large enterprise has many elements and many points of enforcement [XML coverpages 2003a]. Elements of policy may be managed by the information systems department, human resources, the legal department, and the finance department. And the policy may be enforced by the extranet, mail, WAN, and remote-access systems. As most enterprises are migrating from legacy systems to newer infrastructure, the uniformity among the communication protocols across the heterogeneous systems is essential to interoperability, and hence timely and accurate execution of enterprise-level policies. Another concern faced by large enterprise applications is the number of users or clients accessing the enterprise resources, which runs in tens of thousands. Since these resources would typically have privilege levels associated with them, a mechanism needs to be provided to allow only authorized users to access the requested resource. Hence, exercising content-based access control is a significant challenge in securing large enterprises. Yet another dimension that complicates access control specification is the dynamic nature of context-based conditions attached with access decisions. An enterprise may grant/revoke access to resources to certain individuals based on their level of involvement in the current stage of product life cycle. The access constraints are also extended to sharing of information between various users based on their degree of relevance to resources at a particular time. To adequately satisfy these kinds of conditions in a dynamic environment, hence, constitutes another significant challenge. Our primary goal in this paper is to investigate these challenges and propose an XML-based access control specification language and enforcement mechanism that adequately addresses them.

Our specification language is based on generalized temporal role based access control (GTRBAC) model [Joshi et al. 2005]. GTRBAC is a generalized temporal extension of the widely accepted role based access control (RBAC) model proposed in the NIST RBAC standard [Ferraiolo et al. 2001]. RBAC model, because of its generality, can be used for defining a diverse set of access control policies. Another advantage of the RBAC model is that it simplifies authorization administration in large enterprises. RBAC models have also been shown to be policy-neutral [Sandhu et al. 1996], and can be used to represent a variety of security policies, including both DAC and MAC policies [Osborn et al. 2000]. Although several approaches have been presented in the literature based on RBAC to address various aspects of security administration within an enterprise, they have their own drawbacks that render them unsuitable for enterprise-wide access control (see Section 8). GTRBAC extends RBAC to allow a generalized mechanism to express a diverse set of fine-grained temporal constraints on user-role and permission-role assignments in order to meet the dynamic content-based context-aware access control requirements of an enterprise. Our framework augments the GTRBAC model with XML to allow for supporting the policy enforcement in a heterogeneous, distributed environment. Notable extensions with respect to GTRBAC include the support for credentials, which is essential when dealing with heterogeneity (see Section 3.2), and a comprehensive architectural framework, based on the use of modular policy documents, covering all modeling components of a typical RBAC model. Moreover, our framework is based on XML schemas, and not DTDs, and hence has more expressive power and support for data types as compared to DTD-based approaches [IBM].

The remainder of this paper is organized as follows. We first provide an introduction to the basic concepts related to XML and GTRBAC, since their understanding is deemed essential for having a good grasp on the concepts outlined in the paper. The discussion on RBAC and its extensions is also included in this section as part of introducing GTRBAC. We next provide the motivation behind the X-GTRBAC framework, and discuss the achievable goals of our system in context of enterprise security. The next section outlines the formal specifications of the GTRBAC model. The detailed X-GTRBAC specification language is then presented. The implementation architecture and system design is discussed next. A comprehensive example of a generic CIE is then presented to illustrate the applicability of the X-GTRBAC framework. The example not only serves to fortify the discussion on the model, but also brings together the various concepts explained throughout the paper. We then discuss our implementation experiences to highlight significant features of our framework. The paper concludes with a compendium of related work, and discussion on some future research goals.

2. PRELIMINARIES

2.1 XML

The eXtensible Markup language (XML) [XML 2000] evolved from a simple subset of SGML [ISO 1986], and is now hailed as the most promising technology for information interchange across heterogeneous, distributed domains

```
<enterprise>
 <depts>
  <engineering>
    <engg_manager job_id= "EM">
         <name>John</name>
         <level>5</level>
         <…> .. </..>
    </engg_manager>
    <product_engineer job_id="PE">
         <name>Paul</name>
         <shift>1</shift>
         <…> .. </..>
    </product_engineer>
  </engineering>
  <design>
    <design_manager job_id= "DM">
         <name>Adams</name>
         <grade>A</grade>
         <…> .. </..>
    </design_manager>
    <design_engineer job_id="DE">
         <name>Charlie</name>
         <skill>auto</skill>
         <…> .. </..>
    </design_engineer>
  </design>
 </depts>
</enterprise>
                                   (a)
```

```
<xs:schema>
 <xs:element name ="enterprise">
  <xs:complexType>
   <xs:element name = "depts">
    <xs:complexType>
     <xs:element name = "engineering">
      <xs:complexType>
       <xs:element name = "engg_manager">
        <xs:complexType>
         <xs:attribute name ="job_id" type="xs:string"/>
         <xs:element name = "name" type="xs:string"/>
         <xs:element name = "level" type="xs:string"/>
          …….
       </xs:complexType>
      </xs:element>
      <xs:element name = "product_engineer">
       <xs:complexType>
        <xs:attribute name ="job_id" type="xs:string"/>
        <xs:element name = "name" type="xs:string"/>
        <xs:element name = "shift" type="xs:string"/>
         …….
       </xs:complexType>
      </xs:element>
     </xs:complexType>
    </xs:element>
    ……
  </xs:complexType>
 </xs:element>
</xs:schema>                                         (b)
```

Fig. 1. (a) An XML instance document and (b) its schema.

[Web Reference]. XML is a metalanguage that lets users design their own markup language, and hence allows them to define an agreed-upon vocabulary for application-specific tasks. XML achieves this by offering an extensible set of markup tags to create custom documents, and a set of related technologies for their interpretation.

Each XML document has both a logical and a physical structure [Java Commerce]. Physically, the document is composed of units called entities. An entity may refer to other entities to include them in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. Additionally, elements may contain attributes as well.

The structure of the XML document is expressed through an XML schema [W3]. A schema itself is an XML document that defines the valid syntax of an XML instance document, where the term instance document denotes a document conforming to the said schema. Figure 1 illustrates an XML instance document and its corresponding schema. Here, the structure of the various XML tags in the instance document is governed by the schema definition. For instance, the second line in the schema definition declares "enterprise" as the root element of the document. The "depts" element is then added to the root as a child element. The hierarchy is similarly extended to incorporate all the desired elements. This extensible naming mechanism hence allows the creation of customized documents that capture the application-specific needs of any enterprise. For interested readers, the detailed specifications of XML and XML schemas can be found at XML [2000] and [W3].

2.2 RBAC and Its Extensions

As stated in the Introduction, the GTRBAC model is a generalized temporal extension of the RBAC model. We now provide a brief introduction to the RBAC model and its temporal and generalized temporal extensions.

2.2.1 *RBAC Model.* The RBAC model as proposed in the NIST RBAC standard consists of the following four basic components: a set of users `Users`, a set of roles `Roles`, a set of permissions `Permissions`, and a set of sessions `Sessions`. A user is a human being or an autonomous agent. A role is a collection of permissions needed to perform a certain job function within an organization. A permission is an access mode that can be exercised on objects in the system, and a session relates a user to possibly many roles. When a user logs in the system, he/she establishes a session and, during the session can request to activate some subset of roles he/she is authorized to assume. An activation request is granted only if the corresponding role is enabled at the time of the request and the user is entitled to activate the role at that time. If the activation request is satisfied, the user issuing the request obtains all the permissions associated with the role he/she has requested to activate. On the sets `Users`, `Roles`, `Permissions`, and `Sessions`, several relations are defined. The user-to-role assignment (*UA*) and the permission-to-role assignment (*PA*) relations model the assignment of users to roles and the assignment of permissions to roles respectively. A user can be a member of many roles and a role can have many members. Moreover, a role can have many permissions and the same permissions can be associated with many roles. The user function maps each session to a single user, whereas function role establishes a mapping between a session and a set of roles (that is, the roles which are activated by the corresponding user in the session). On `Roles`, a hierarchy is defined, denoted by $\geq$. If $r_i \geq r_j$, r_i, $r_j \in$ `Roles`, then r_i inherits the permissions of r_j. In such a case, r_i is a senior role and r_j a junior role. The following definition formalizes the above discussion.

Definition 2.2.1 (*RBAC Model* [*Ferraiolo et al.* 2001]). The RBAC model consists of the following components:

- Sets `Users`, `Roles`, `Permissions`, and `Sessions` representing the set of users, roles, permissions, and sessions, respectively;
- *PA*: `Roles` $\rightarrow$ `Permissions`, the permission assignment relation that assigns permissions to roles;
- *UA*: `Users` $\rightarrow$ `Roles`, the user assignment relation that assigns users to roles;
- *user*: `Sessions` $\rightarrow$ `Users`, which maps each session to a single user;
- *role*: `Sessions` $\rightarrow$ 2^{Roles} that maps each session to a set of roles;
- *RH* $\subseteq$ `Roles` $\times$ `Roles`, a partially ordered role hierarchy (written $\geq$).

The RBAC model differentiates itself from traditional access control models in that the permissions in RBAC are not directly associated with users, but with roles. Roles are created by the security administrators to reflect the various functional categories of users within the enterprise. Users are then assigned membership to roles, and these roles are in turn assigned permissions.

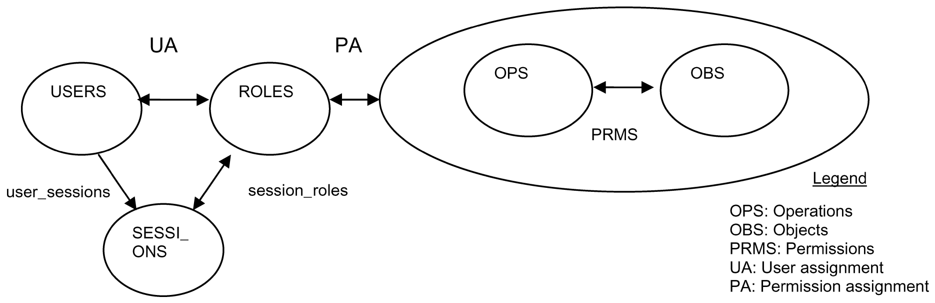


Fig. 2. Core RBAC elements and their relation (source: NIST RBAC standard [Ferraiolo et al. 2001]).

Permissions are actually composed of an object-to-operations mapping. The element relationships of Core RBAC model are illustrated in Figure 2.

RBAC approach naturally fits into an organizational context as users are assigned to organizational roles that have well-defined responsibilities and privileges. The RBAC model proposed by NIST, because of its generality, can be used to express a very wide range of security policies including discretionary and mandatory, as well as user-defined organizational specific policies [Ferraiolo et al. 1993, 2000; Osborn et al. 2000; Sandhu et al. 1996]. Many benefits of an RBAC approach include its support for security management and the principle of least privilege [Sandhu et al. 1996]. For example, we can easily manage a change in a user's responsibility or role within the organization by assigning him/her the new role and removing him/her from the old one. Furthermore, use of role hierarchies and grouping of objects into object classes based on responsibility associated with a role makes the management of permissions very easy. By configuring the assignment of the least set of privileges from a role set assigned to a user when he/she activates the role, inadvertent damage can be minimized in a system.

2.2.2 *Temporal Extensions to RBAC.* Because of its relevance and abovementioned benefits that it provides, the RBAC model has been widely investigated and several extensions to it have been proposed. A set of such extensions related to the temporal dimension of the model shall now be discussed. An initial temporal extension to RBAC has been proposed in the temporal RBAC (TRBAC) model [Bertino et al. 2001]. This has been motivated by the fact that in many organizations, functions may have limited or periodic temporal duration. Consider, for instance, the case of a component technician in a manufacturing enterprise and assume that any technician is to be authorized to work only when a supervisor is at work. If both the technician and supervisor are represented as roles, enforcing such a requirement entails constraining the enabling of a technician role only during the specified temporal interval when the supervisor role is enabled. TRBAC allows the specification of such temporal conditions. The main features it provides include the periodic enabling/disabling of roles and temporal dependencies among them expressed by means of role triggers, which are active rules that are automatically executed based on the enabling and/ or disabling of roles. Priorities are associated with both triggers and periodic enabling/disabling of roles to handle possible conflicts that can arise, when the simultaneous enabling/disabling of a role is required. In such cases, a combination of priority and disabling-take-precedence rule is used to resolve the conflicts. TRBAC further allows an administrator to issue run-time requests for enabling and disabling a role and restricted handling of role activations by a user. TRBAC, however, is inadequate to express a variety of useful temporal constraints. In particular, TRBAC does not include temporal constraints on (i) user-to-role and permission-to-role assignments, and (ii) activations of roles by users. In (i), TRBAC assumes that only roles can be transient, that is, only they are enabled and disabled at different time intervals. In this paper, we motivate the point that in a typical enterprise environment, roles, as well as users and permissions assigned to them, may also be transient. Because of (ii), TRBAC does not use a well-defined, separate notion of role enabling and role activation, and hence cannot enforce a fine-grained access control at the user level for role activation. The GTRBAC model distinguishes between the notions of role activation from that of role enabling to incorporate various activation constraints on role activations at the individual user level. It also extends the temporal constraint enforcement mechanism to user-to-role and permission-to-role assignments. It thus allows the specification of a more complete set of temporal constraints related not only to role enabling, but also to user-to-role assignment, permission-to-role assignment, and role activation. The notions of triggers and safety, as provided in TRBAC, are also accordingly extended to capture the enhanced language semantics. The X-GTRBAC framework hence builds upon the elaborate and consistent set of specifications laid out in the GTRBAC model.

3. MOTIVATION AND GOALS

We now present a motivation for adopting XML and GTRBAC for our X-GTRBAC policy specification framework, and highlight the achievable goals in the light of its application to enterprise-wide access control.

3.1 Motivation for XML

The use of XML is primarily motivated by the vast heterogeneity of collaborating entities exhibited by the distributed enterprise environment. The various functional units within an enterprise, connected through multiple media, and each comprising several computing systems ranging from old to new, are linked together by the EC technology. Hence the need arises for an interoperable mechanism to efficiently express and enforce the enterprise access control policy. XML provides a uniform, vendor-neutral representation of enterprise data, and allows a mechanism for interchange, sharing and dissemination of information content across heterogeneous systems. XML is, therefore, a natural choice as the basis for the enterprise policy specification language, due to the ease with which its syntax and semantics can be extended to accommodate the unique requirements of an enterprise, and the widespread support that it enjoys from all the main platform and tool vendors [XML Coverpages 2003a] which promotes interoperability.

The motivation for XML actually goes beyond the fact that it enjoys widespread support and is being used as the de facto communication standard in distributed environments. The interoperability advantage that XML provides has great consequences in the contemporary security arena. There are today a large number of evolving standards and most prominent amongst them are those designed for Web-based security (including authentication and access control). The design of Web-based security mechanisms is a major shift from the design of classical protocols since it faces the arduous task of securing inherently dynamic Web-based resources in open Internet environment, with the additional requirement that resources be accessible through very basic mechanisms and tools (such as Web browser). The support for interoperability in XML has therefore led to a significant interest in using XML for designing such protocols, since it would enable the deployment of interoperable, lightweight (i.e., browser based) security mechanisms in open environments. This present attempt at augmenting GTRBAC with XML is motivated by the emerging trend of migration of enterprise operations on the Internet, which necessitates sophisticated Web-based access control mechanisms. We discuss additional motivations for adopting XML in conjunction with GTRBAC at the end of next subsection.

3.2 Motivation for GTRBAC

The discussion on the challenges in Introduction would now be presented in greater detail. The motivation behind adopting the GTRBAC model for enforcing enterprise-wide access control is to incorporate within the access control model the following set of capabilities:

3.2.1 *Content-Based Context-Aware Access.* Information access within an enterprise may need to be restricted based on the information content and the contextual information obtained at the time the access requests are made. For example, an external client may not be allowed to access the product design manual from the enterprise document repository. Or it may be the case that only the authorized component technicians are allowed to access the plant inventory for a manufacturing enterprise, and such access would be restricted to the components related to the technician's job function. Hence the need arises to exercise content-based access control for all users accessing enterprise resources. The access control model should also capture security-relevant environmental context and incorporate it in its access control decisions. Access requests may be decided based on several context parameters, such as time or location. An example of location parameter is user domains, which are classified by IP addresses. In the manufacturing enterprise, the access control model could allow all users who submit an access request from within the design department Intranet to access the product design manual at any time for easy reference. This access may, however, be limited to be read-only for certain less privileged users. The time parameter needs to be incorporated in the model to express the time-dependent access constraints within an enterprise. For instance, in addition to the restriction that a particular component technician should be granted access to only the relevant components of the product, it is quite likely that such access is further restricted to only the times when the product is in the manufacturing stage. Another view of time-dependent constraints captures the periodic nature and associated duration of enterprise tasks. One example for such constraint could be a rule that the vendor contracts may only be allowed to be accessed by vendors in the second week of every quarter of every year, and need to be submitted within two weeks of that time. More complicated context conditions allow for expressing even more sophisticated constraints. As an example, it may be required for the manufacturing enterprise that the product engineering work should only start after the product design work has been completed. This requires for checking various context parameters to evaluate the stated condition, and allow for the requested or scheduled task to be executed. A mechanism for the resolution of these challenges based on a combination of the temporal framework of the GTRBAC model and the attribute-based constraint specification of X-GTRBAC shall be discussed in the paper, and illustrated through a comprehensive CIE example in Section 7.

3.2.2 *Heterogeneity of Subjects and Objects.* In a CIE, heterogeneity implies the diversity of users and resources across the component systems making up the enterprise. Object heterogeneity may exist in the form of different types of enterprise resources that need to be protected. These can range from purchase and marketing contracts, to design and engineering manuals, to various product assemblies and components. Furthermore, the information content therein can evolve with time as new resources are added and old ones removed or updated, introducing scalability problems in privilege management. Subject heterogeneity implies that users have diverse activity profiles, characteristics, and/or qualifications that may not be known a priori. Such activity profile is needed by the EC technology to dynamically confer privileges to authenticated users by upgrading their current role. This is needed because the trust level of the user may be elevated, as the product goes through the various stages within the enterprise, and access to more privileged resources may accordingly be allowed. For instance, a product technician with sufficient experience may be elevated to the role of product supervisor, and allowed to access the product assemblies component at the time of manufacturing of the product assembly. Subject heterogeneity complicates access control specification. The GTRBAC model uses the abstraction of roles to manage a large number of user and resource pools, which tackles the scalability aspect but not the diversity aspect of the subject heterogeneity problem. The issue there is that user identity cannot be assumed for role assignment, since the assignment conditions might depend on transient/dynamic properties of a user. We resolve this issue by using the notion of "credentials" in X-GTRBAC, as discussed in Section 5.1. Following a credential-based role assignment, the GTRBAC constraint mechanism can then be applied for privilege management. Object heterogeneity can similarly be addressed by associating relevant content-based constraints with resources.

The preceding discussion provides the motivation for using XML and GTRBAC toward the design of an access control scheme for a distributed enterprise environment. XML addresses the issues of interoperability between various protocols and across multiple enterprise units. GTRBAC addresses the content-based context-aware access control requirements and the subject and object heterogeneity challenges within a CIE. The need arises therefore to combine the features from these two models in an integrated framework that can be employed for meeting the aforementioned challenges of enterprise-wide access control. The use of XML in security protocols is not new. The area of XML security has gained significant momentum in the last few years, with standards such as Security Assertions Markup Language (SAML) [XML Coverpages 2003b] and XML-based Access Control Language (XACML) [XML Coverpages 2003a] recently been adopted. However, the unique requirements highlighted in the paper have not been addressed by any existing specification (see Section 8). We believe that our XML-based framework can contribute to the XML security research by providing an access control mechanism that not only meets these requirements, but is also interoperable with existing XML security standards. In fact, the authors have used this framework to propose an approach that integrates SAML with RBAC [Bhatti et al. 2004a]. Our work has been included in a recent announcement by the OASIS industrial consortium that hailed the emergence of Web-based specifications for RBAC security standard [XML Coverpages 2004]. Toward further advancement of such efforts, the X-GTRBAC framework presented in this paper outlines an XML-based specification language that focuses on encapsulating all the basic features set forth in the GTRBAC model. We next outline the formal specifications of the GTRBAC model, and in the subsequent section introduce the specification language for our X-GTRBAC framework.

4. GTRBAC MODEL

The GTRBAC model allows the specification of an elaborate set of temporal constraints on role enabling/disabling, activation/deactivation, and user-to-role and permission-to-role assignment/de-assignment. These constraints are composed of periodic-time expressions that capture the valid periods of time when the corresponding constraint may be satisfied. This expression has three parts: (i) a start-time expression, indicating a valid start time, (ii) an interval expression, indicating the interval within which the constraint may be satisfied, and (iii) a duration expression, indicating the valid duration for the constraint. Note that all parts of the periodic-time expression are optional, and the absence of any one of them means that there is no corresponding time restriction. The model additionally allows trigger-based temporal conditions to be specified. These conditions are captured through status expressions for roles and assignments.

Specifically, the following temporal constraints and events are allowed in GTRBAC:

1. *Temporal constraints on*
   (a) *Role enabling*: These constraints allow the specification of the valid time periods during which a role is enabled/disabled. Additional content and context conditions may also be supplied within the constraint.
   (b) *Role activation*: These constraints allow the specification of the valid time periods during which a role is activated/deactivated. Additional

content and context conditions may also be supplied within the constraint. Role activation only occurs as a run-time event.

(c) *User-to-role assignment*: These constraints allow the specification of the valid time periods during which a user may be assigned to a role. Additional content and context conditions may also be supplied within the constraint.

(d) *Permission-to-role assignment*: These constraints allow the specification of the valid time periods during which a permission may be assigned to a role. Additional content and context conditions may also be supplied within the constraint.

2. *Run-time events*: A set of run-time events allows a user or an administrator to dynamically initiate actions. One such example captured in our framework is activation/deactivation of a role. Note that activation requests for a role may only be supplied by the user, since role activation is done at the user's discretion.
3. *Triggers*: Triggers allow for expressing dependency among GTRBAC events, as well as capturing the past events and defining future events based on them. Triggers may not include any activation event in their head, for the reason cited above.

   In addition to the above temporal constraints, the GTRBAC model supports the following separation of duty constraints as per the NIST RBAC standard:

   (a) *Static separation of duty*: The semantics of static separation of duty require that no `n` roles that are part of a "static separation of duty role set" (SSD_Role_Set) be assigned to the same user, where `n` is any positive integer.

   (b) *Dynamic separation of duty*: The semantics of dynamic separation of duty require that no `m` roles that are part of a "dynamic separation of duty role set" (DSD_Role_Set) be simultaneously active in the same session of the same user, where `m` is any positive integer.

We reproduce below, for completeness, the formal expressions for the specification of periodic time and the temporal constraints and events in GTRBAC.

Definition 4.2.1 (*Periodic Expression* [*Bertino et al.* 1998]). Given calendars $C_d, C_i, \ldots, C_n$, and time occurrences $O_1, \ldots, O_n$, a periodic expression P is defined as

$$\mathtt{P} = \sum_{i=1}^{n} O_i.C_i \rhd x.C_d$$

where $O_1 = \mathtt{all}$, $O_i \in 2^{\mathrm{N}} \cup \{\mathtt{all}\}$, $C_i \sqsubseteq C_{i-1}$ for $i = 2, .., n$, $C_d = C_n$, and $x \in \mathbb{N}$.

Periodic-Time Expression: Periodic time is represented by pairs <[`begin`, `end`], `P`>, where `P` is a periodic expression denoting an infinite set of periodic time instants, and [`begin`, `end`] is a time interval `I` denoting the lower and upper bounds that are imposed on instants in `P`.

The formalism for periodic expressions is based on the one used in Niezette and Stevenne [1992], and relies on the notion of calendars. A calendar is defined as a countable set of contiguous intervals,[1] numbered by integers called indexes of the intervals. In our discussion, we assume the existence of a set of calendars containing the calendars days, weeks, months, and years, where days is the calendar with the finest granularity, that is, it is the basic calendar.

Symbol $\triangleright$ separates the first part of the periodic expression that identifies the set of starting points of the intervals it represents, from the specification of the duration D of each interval in terms of calendar C_d. For example, $\texttt{all}.\mathit{Years} + \{3, 7\}.\mathit{Months} \triangleright 2.\mathit{Months}$ represents the set of intervals starting at the same instant as the third and seventh month of every year, and having a duration of 2 months. In practice, O_i is omitted when its value is `all`, whereas it is represented by its unique element when it is a singleton. $x.C_d$ is omitted when it is equal to $1.C_n$.

Event expression: A simple event expression has one of the following forms:
- (a) `enable` r or `disable` r, where $r \in$ `Roles`.
- (b) `assign` r `to` u or `de-assign` r `to` u, where $r \in$ `Roles` and $u \in$ `Users`.
- (c) `assign` p `to` r or `de-assign` p `to` r, where $p \in$ `Permissions` and $r \in$ `Roles`.

Role status expression: Role status expressions have one of the following forms:
- (a) `enabled` r or $\neg$ `enabled` r (or `disabled` r), where $r \in$ `Roles`.
- (b) `activated` r, where $r \in$ `Roles`.
- (c) `active` r `for` u or $\neg$ `active` r `for` u, where $r \in$ `Roles` and $u \in$ `Users`.

Assignment status expression: Assignment status expressions have the following forms:
- (a) `assigned` r `to` u or $\neg$ `assigned` r `to` u, where $r \in$ `Roles` and$u \in$ `Users`.
- (b) `assigned` p `to` r or $\neg$ `assigned` p `to` r, where $r \in$ `Roles` and $p \in$ `Permissions`.

Run-time request: A run-time request expression has one of the following forms:
- (a) a user's run-time request expression to activate a role has the form:

  s: `activate` r `for` u `after` Δt,or
  s: `deactivate` r `for` u `after` Δt

  where $r \in$ `Roles`, $u \in$ `Users`, s is the session attached to the request, and Δt is the duration.
- (b) an administrator's run-time request expression has the form:

$$E \texttt{ after } \Delta t$$

  where E is an event expression and Δt is the duration expression.

Triggers: A trigger expression has the form

$$E_i, \ldots, E_n, C_i, \ldots, C_k \rightarrow E \texttt{ after } \Delta t$$

where E_is are event expressions or run-time requests, C_is are role status expressions or assignment status expressions, E is an event expression such that $E \notin \{$`s:activate r for u`$\}$, and Δt is a duration expression.

Table I summarizes the temporal constraint types and expressions of the GTRBAC model. The GTRBAC model extends the safety notion of the TRBAC

[1]Two intervals are contiguous if they can be collapsed into a single one (e.g., [1, 2] and [3, 4]).

Table I. Temporal Constraints and Event Expressions in GTRBAC

Constraint Categories	Events	Expression
Enabling	Role enabling	(I, P,D, enable/disable r)
Activation	Role activation	<!--only occurs as a run-time event -->
Assignment Constraint	User-to-role assignment	([I, P, D], assign$_U$/deassign$_U$ r to u)
	Permission-to-role assignment	([I, P, D], assign$_P$/deassign$_P$ p to r)
Trigger	<!--any triggering event -->	$E_1, \ldots, E_n, C_1, \ldots, C_k \rightarrow E$ after Δt
Run-time Requests	*Users' activation request*	(s:(de)activate r for u after Δt))
	Administrator's run-time request	(assign$_U$/de-assign$_U$ r to u after Δt)
		(enable/disable r after Δt)
		(assign$_P$/de-assign$_P$ p to r after Δt)
		(enable/disable c after Δt)

model to show that there exists an execution model for it. Hence the consistency of our framework is implied by the consistency of GTRBAC model.

5. X-GTRBAC SPECIFICATION LANGUAGE

We now present the specification language for our X-GTBRAC framework. The specification is designed to serve dual goals. One, it attempts to model the RBAC elements and incorporate the functional specifications as per the NIST RBAC standard [Ferraiolo et al. 2001]. Secondly, it provides the syntactic and semantic constructs needed to enforce the temporal constraints as per the GTRBAC model, and in addition also provides an attribute-based credential and constraint specification framework to address the access control challenges outlined in Section 3.2.

5.1 Modeling RBAC Elements

Initial goal of the X-GTRBAC specification language is to model the five basic RBAC elements (as shown in Figure 2) and their associated set-relations. To represent the RBAC elements in XML, we generate schema definitions for "user," "role," and "permission." Note that schema definition is not necessary for "operation" and "object" elements because they are included in a "permission" definition as per the RBAC standard, and hence their relationship with other RBAC elements is captured in the "permission" schema. We introduce a BNF-like grammar, called X-Grammar, to present an overview of the specification language for the RBAC elements in an XML-syntax.

X-Grammar: The X-Grammar follows the same notion of terminals and nonterminals as in BNF, but supports the tagging notation of XML that also allows expressing attributes within element tags. The use of attributes helps maintain compatibility with XML schema syntax, which serves as the type definition model for our language. Since it follows BNF convention, X-Grammar can be accepted by a well-defined automaton to allow automatic translation into XML schema documents. This allows automatic creation of strongly typed policy schemas based on the supplied grammar specification. We choose to use X-Grammar syntax instead of directly working with XML schemas for ease of analysis (since existing compiler tools for BNF grammars can be applied) and better readability and presentation.

```
<!-- Definitions of Credential Types>
::=
<XCredType [xctd_id = (id) ] >
  {<!-- Credential Type Definition>}+
</XCredType>
```

```
<!-- Credential Type Definition> ::=
<CredType
cred_type_id = (id)
type_name= (type name) >
  <AttributeList>
     {<!-- Attribute Definition>}+
  </AttributeList>
</CredType >
```

```
<!-- Attribute Definition> ::=
<Attribute>
  <AttributeName
     usage = "mand | opt"
     type = (type)> (name)
</AttributeName >
```

Fig. 3. X-Grammar for XCredTypeDef sheet.

```
<!-- User Definitions >::=
<Users>
     {<!-- User Definition>}+
</Users>
```

```
<!-- User Definition>     ::=
<User user_id = (id)>
     <UserName> (name) </UserName>
     {<!--CredType>}+
     <MaxRoles>(number)</MaxRoles>
</User>
```

```
<!—CredType        > ::=
<CredType cred_type_id = (id)
          type_name= (type name) >
   <!-- Credential Expression>
</CredType>
```

```
<!-- Credential Expression>       ::=
<CredExpr>
   {<(attribute name)> (attribute value)
   </(attribute name)>}+
</CredExpr>
```

Fig. 4. X-Grammar for XUS.

The non-terminals in X-Grammar are expressed as <!–"non_terminal_name"> XML tags, and terminals as standard XML tags. Optional tags are placed within square brackets "[ ]". Group portions of a production are included in curly brackets "{}," with the repeat count indicated by a subscript. The default count is one. A "∗" and a "+" indicates a count of "zero or more," and "one or more," respectively, whereas a "–" is used to provide a range. A "|" indicates alternates within a production set, and exactly one can be chosen. Any data placed in parenthesis "( )" is not part of the terminal symbol, and shall be supplied by the security administrator. The X-Grammar has been adopted for a clear expression of the specification language constructs. The corresponding schemas for the XML sheets listed in the paper are provided in Appendix A in Bhatti [2003].

5.1.1 *Users and Credentials.* To evaluate the users being assigned to a particular role, the specification language uses the notion of credentials as discussed in Bertino et al. [1999a]. A "credential type" is created by the security administrator to group users based on their credentials, and hence enforcing a common set of attribute-value pairs for a given group. This set of attributes constitutes the "cred_expr" for the given credential type. A credential type definition schema (XCredTypeDef) is supplied as part of the specification language to facilitate the creation of new credential types.

With respect to the grammar for the XCredTypeDef sheet shown in Figure 3, mand indicates that the attribute is mandatory whereas opt indicates that it is optional. The credential information in XCredTypeDef sheet provides a vocabulary to express the credentials needed by the users of any organization in order for them to be considered for assignment to specific roles. Users and their credentials are expressed in the form of an XML document that we refer to as XML user sheet (XUS). The grammar for XUS is shown in Figure 4. The

```
<!-- XML Role Sheet> ::=
<XRS [xrs_id = (id) ]>
  {<!-- Role Definition>}+
</XRS>
```

```
<!-- Role Definition> ::=
<Role role_id = (id)
        role_name = (role name)>
   [<!--Attributes>]
   [<!--{En|Dis}abling Constraint>]
   [<!--[De]Activation Constraint>]
   {<SSDRoleSetID> (id) </SSDRoleSetID>}*
   {<DSDRoleSetID> (id) </DSDRoleSetID>}*
   [<Junior> (name) </Junior>]
   [<Senior> (name) </Senior>]
   [<Cardinality> (number) </Cardinality>]
</Role>
```

Fig. 5. X-Grammar for XRS.

```
<!-- Separation of Duty Definitions> ::=
<XSoDDef [xsod_id = (id) ]>
   [<!--SDRoleSets>]
   [<!--DSDRoleSets>]
</XSoDDef>
```

```
<!-- SSDRoleSets > ::=
<SSDRoleSets>
   {<!--SSDRoleSet>}+
</SSDRoleSets>
```

```
<!-- DSSDRoleSets > ::=
<DSDRoleSets>
   {<!--DSDRoleSet>}+
</DSDRoleSets>
```

```
<!-- SSDRoleSet>   ::=
<SSDRoleSet>
  {<SSDRole ssd_role_set_id =(id)
        ssd_cardinality = (number)>
         (role name)
    </SSDRole>}+
</SSDRoleSet>
```

```
<!-- DSDRoleSet>   ::=
<DSDRoleSet>
  {<DSDRole dsd_role_set_id =(id)
        dsd_cardinality = (number)>
         (role name)
    </DSDRole>}+
</DSDRoleSet>
```

Fig. 6. X-Grammar for XSoDDef sheet.

"max_roles" tag indicates the maximum number of roles that a user can be assigned to. As shall be elaborated in the next section, user credentials may be updated dynamically to capture the activity profile of the user.

5.1.2 *Roles.* Roles are also created by the security administrator. A role has an associated set of credentials that must be satisfied by the users who are assigned to that role. Roles and their associated information is expressed in the form of an XML document that we refer to as XML role sheet (XRS). The grammar for XRS is shown in Figure 5.

The "role_name" is a unique role identifier. The cardinality of a role is the maximum number of users assigned to it at any time. If none is explicitly supplied, it is assumed unlimited.

5.1.2.1 *Separation of Duty Constraints.* Each role definition contains optional "SSD_Role_Set_id" and "DSD_Role_Set_id" tags which refer to the set of roles that are collectively in static and dynamic separation of duty, respectively, as per the NIST RBAC standard. Each of SSD_Role_Set and DSD_Role_Set has a cardinality attribute that gives the maximum number of roles that a user may be assigned to or can simultaneously have active in his/her sessions from the set. The SSD_Role_Sets and DSD_Role_Sets are supplied in a separate XSoDDef sheet. The grammar for XSoDDef sheet is shown in Figure 6.

```
<!--{En|Dis}abling Constraint>  ::=
   <{En|Dis}abConstraint
       [op = {AND|OR|NOT}] >
     {<!--{En|Dis}abling Condition>}+
</{En|Dis}abConstraint>
```

```
<!--[De]Activation Constraint> ::=
   <[De]ActivConstraint
       [op = {AND|OR|NOT}]>
   {<!--[De]ActivationCondition>}+
</[De]ActivConstraint>
```

```
<!--{En|Dis}abling Condition>  ::=
<{En|Dis}abCondition
    [{pt_expr_id=(id) |
      d_expr_id=(id)}]  >
  [<!-- Logical Expression>]
<{En|Dis}abCondition>
```

```
<!--[De]Activation Condition> ::=
   <[De]ActivCondition
     [d_expr_id=(id)] >
   [<!-- Logical Expression>]
</[De]ActivCondition >
```

Fig. 7. X-Grammar for XRS constraints.

5.1.2.2 *Role Hierarchies.* The optional "junior" and "senior" tags referring to junior and senior roles are used to capture hierarchical relationships in the RBAC model. The exact semantics that should be enforced on roles within a hierarchy are determined by the target enterprise. However, the notion of "authorized roles" and "authorized permissions" as stated in the NIST RBAC standard is supported by our framework. "Authorized roles" refers to the set of assignable roles for a user, including the roles that are directly assignable to the user based on his/her own credentials, as well as the "junior" roles of all such roles. "Authorized permissions" is the set of corresponding permissions for the authorized roles.

5.1.2.3 *Temporal and Nontemporal Context-Based Constraints.* We now discuss the tags of a role that capture the semantics of both temporal and nontemporal constraint specification and related information. All these tags are optional since their omission simply implies the absence of constraints in any given specification. This set of constraints is supplied in a separate XTempConstDef sheet. The grammar for XRS constraints specification is shown in Figure 7. The grammar for corresponding XTempConstDef sheet is shown in Figure 8.

The "Attributes" tag of the role contains a list of role attributes that may be parameters of the context conditions which need to be dynamically evaluated for any role enabling/disabling or activation/deactivation. The context conditions may be based on temporal (such as time) or nontemporal (such as system load) parameters, or on GTRBAC status expressions such as "whether role R has been enabled by user U." The temporal parameters are captured through GTRBAC temporal constraint expressions, whereas nontemporal context is captured using attribute-based X-Grammar constraint specification. The "(En|Dis)abling Constraint" and "[De]Activation Constraint" tags contain a set of conditions, where each condition is composed of possibly multiple logical expressions for specification of the respective constraints. The constraint tag has an optional op-code attribute that determines the evaluation logic of the expressions within the constraint. An op-code of (i) "AND" implies that all constituent expressions must be true for the constraint to be true, (ii) "OR" implies that at least one expression must be true for the constraint to be true, and (iii) "NOT" implies that none of the expressions must be true for the constraint to be true. The op-code defaults to "AND" if none is specified.

<!-- Definitions of Temporal Constraints> ::= <XTempConstDef [xtcd_id = (id)] > [<!—Interval Expression>] [<!-- Periodic Time Expression>] [<!-- Duration Expression>] </XTempConstDef>	<!-- Periodic Time Expression> ::= <PeriodicTimeExpr pt_expr_id = (id) [d_expr_id = (id)] [i_expr_id = (id)] > <!-- Start Time Expression> </PeriodicTimeExpr>
<!—Interval Expression> ::= <IntervalExpr i_expr_id = id)> <begin> (date)</begin> <end> (date)</end> <IntervalExpr>	<!-- Duration Expression> ::= <DurationExpr d_expr_id = (id)> <cal>{Years|Months|Weeks|Days}</cal> <len> (number)</len> </DurationExpr>
<!-- Start Time Expression> ::= <StartTimeExpr [pt_id_ref =(pt_id)] > [<Year>{all|odd|even} /<Year>] [<!--MonthSet>] [<!--WeekSet>] [<!--DaySet>] </StartTimeExpr>	<!--MonthSet> ::= <MonthSet> {<Month>{1|..|12}</Month>}$_{1-12}$ </MonthSet >
<!--WeekSet> ::= <WeekSet> {<Week>{1|..|4}</Week>}$_{1-4}$ </WeekSet >	<!--DaySet> ::= <DaySet> {<Day>{1|..|7}</Day>}$_{1-7}$ </DaySet >

Fig. 8. X-Grammar for XTempConstDef sheet.

Each condition tag may contain a "pt_expr_id" or "d_expr_id" attribute that refers to a periodic-time or a duration expression, respectively. These expressions are the XML representation of the periodic-time expression framework provided in the GTRBAC model, and bind the corresponding condition with the respective periodic-time expression. We give an XML representation for each of the start-time, interval, and duration expressions that together constitute the periodic-time expression. Following the notion of "calendars" used in the GTRBAC model, the start time expression consists of "calendar sets," where each calendar is a unit of time, for example, years, months, weeks, and so on. As an example, an event that occurs at the start of the second week of every first and eighth month of every odd year would be represented by using "{odd}" as the Year set, "{1, 8}" as the Month Set, and "{2}" as the Week Set. The optional "pt_id_ref" attribute indicates start time with reference to the provided periodic-time expression id. If it is supplied, then the start time is the same as that of the referenced periodic time. Note that a "pt_id_ref" is provided only when the calendar sets are not provided, and vice versa. Any new start time is always explicitly defined using new calendar sets. An interval is given by a (begin_date, end_date) pair, and a duration is specified as (calendar, calendar_length) pair. The semantics of the periodic time expression thus dictate that the associated event can only occur if the start time expression is satisfied by the time of request, and such time falls within the interval specified by the interval expression. The duration of the event, if it occurs, would be governed by the duration expression.

The "Logical Expression" tag contains a set of predicates, where each predicate may contain a context-condition expressed in terms of role attributes, or embed within itself another logical expression. Hence, the structure allows

```
<!-- Logical Expression> ::=
<LogicalExpr [op = {AND|OR|NOT}]>
  {<!-- Predicate>}+
</LogicalExpr>
```

```
<!-- Predicate> ::=
<Predicate>
   {<Operator> {gt|lt|eq|neq} </Operator>
      [<FuncParam>(function name)</FuncParam>]
      {<NameParam [type=(role|user|domain)]>
                  (parameter name)</NameParam>}+
      <ValueParam>(value)</ValueParam> }
      |
      < !--LogicalExpression>
</Predicate>
```

Fig. 9. X-Grammar for logical expression.

evaluation of nested conditions expressed by multiple logical expressions. The predicates are composed of contextual parameters, where the "ParamName" tag contains the name of the parameter to be evaluated, "FuncName" tag contains the name of the function used to evaluate the parameter, and the "RetValue" tag contains the expected return value that is to be checked according to the given "Operator." For instance, any attribute supplied as part of user credential expression may be compared for a prerequisite value needed for certain role assignment or activation by supplying the attribute name as "ParamName," the required values as "RetValue," and the comparison operator as "Operator." The simplest "FuncName" (as in this case) is the "isEqual()" (or equivalent) function. In more complex cases, an appropriate "FuncName" is used to evaluate the supplied parameter(s) through a system review function, such as the status expressions of GTRBAC model. Multiple parameter names may be passed to functions that evaluate multiple parameters, with the distinction among parameter types made with the "type" attribute. As an example of a complex predicate, we might evaluate status expressions for a role by supplying a status condition such as "`active` r `for` u" as "FuncName," the role name and the user id as two instances of "ParamName," and the value of either "True" or "False" as the "RetValue." In such situations where a Boolean output is returned, only "eq" operator is useful for comparison. The "Logical Expression" tag also has an optional op-code attribute that determines the evaluation logic of the predicates. On the similar lines as the constraint tag, an op-code of (i) "AND" implies that all constituent predicates must be true for the logical expression to be true, (ii) "OR" implies that at least one predicate must be true for the logical expression to be true, and (iii) "NOT" implies that none of the predicates must be true for the logical expression to be true. The op-code defaults to "AND" if none is specified. The grammar for logical expression specification is shown in Figure 9.

5.1.2.4 *Triggers.* The grammar for constraint specification is also used to capture the trigger mechanism of GTRBAC model. Since an X-Grammar constraint syntax can include both temporal and nontemporal contextual parameters, it allows for specification of context-based triggers in our X-GTRBAC framework. This set of triggers is supplied in a separate XTrigDef sheet. The grammar for XTrigDef sheet is shown in Figure 10.

The "Head" tag of the trigger has an attribute that indicates the target role or the permission on which the trigger action is performed. An optional "user_id" attribute is also supplied for triggers that need to perform the action with

```
<!--Triggers Specification> ::=
<XTrigDef [xtd_id = (id) ] >
  {<!-- Trigger>}*
</XTrigDef>
```

```
<!--Trigger> ::=
<Trigger [trig_id = (id) ] >
      <Head {role_name = (name) | perm_id = (id) }
            [user_id = (id)]
            action = {enable | disable |
                         assign | deassign | deactivate} >
     </Head>
     </Body>
       <!--Triggering Constraint>
     </Body>
</Trigger>
```

```
<!--Triggering Constraint> ::=
  <TrigConstraint
      [op = {AND|OR|NOT}] >
    {<!--Triggering Condition>}+
</TrigConstraint>
```

```
<!—Triggering Condition>  ::=
<TrigCondition>
  [<!-- Logical Expression>]
<TrigCondition>
```

Fig. 10. X-Grammar for XTrigDef sheet.

```
<!-- XML Permission Sheet> ::=
<XPS [xps_id = (id) ] >
  {<!-- Permission Definition>}+
</XPS>
```

```
<!-- Permission Definition> ::=
<Permission perm_id = id [prop= (prop op)] >
   <Object type= (type name) id= (id)>
        [<!-- Attributes>] </Object>
   <Operation> (access op) </Operation>
</Permission>
```

Fig. 11. X-Grammar for XPS.

respect to certain individual users. The triggering constraint in "Body" tag is semantically similar to the constraints discussed above, and is evaluated in an analogous manner. The action associated with the trigger is performed if the constraint evaluates to true.

5.1.3 *Permissions.* The permissions for a given system are defined in terms of "objects" and associated "operations." The "operations" component of the permission is typically system dependent, such as read, write, delete, create, operate, and so on. The security administrator creates the permissions that associate the objects in the system with corresponding operations. The set of permissions for a system is expressed in the form of an XML document that we refer to as XML Permission Sheet (XPS). The grammar for XPS is shown in Figure 11.

The "perm_id" is a unique permission identifier. An object in our framework can represent any system resource, such as documents, or inventory products, to which permission is being assigned. Each object is represented by a unique id and an associated type attribute. The access control requirements for various object types in an enterprise are therefore handled uniformly by our X-GTRBAC framework. The extent of the access is defined by the associated operation, indicated by an access opcode which is one of an enumerated set of values in the system. The "Attributes" tag of the object contains a list of resource attributes that may be used to compose content-based conditions for permission-to-role assignment. The resources in the system are modeled as XML, and the natural hierarchical structure of XML DOM is used to capture the physical object hierarchy. An object hierarchy could be composed of either documents, or document

Table II. The XML Sheets Comprising the XML Policy Base

Primary Policy Sheets	Policy Definition Sheets
XUS	XCredTypeDef
XRS	XSoDDef
XPS	XTempConstDef
XURAS	XTrigDef
XPRAS	—

elements (in case of XML documents), or a series of inventory products organized according to their order of assembly, or any other organization of system resources. A permission can, hence, have an optional propagation option, given by the "prop" attribute, which indicates whether or not it propagates down the object hierarchy. We allow the propagation options "no_prop", "first_level," and "cascade" [Bertino et al. 2001]. If no propagation option is explicitly supplied, it is assumed to be "no_prop," that is, no propagation. However, the security administrator can specify a different propagation option at the time of permission-to-role assignment if a role demands sufficient privileges.

5.2 Policy Administration

The information about users, roles and permissions, and the related credentials, separation of duty constraints, temporal constraints, and triggers, available from the corresponding XML documents are used in the process of policy administration. The security administrator uses these XML sheets to specify the policy base for the protected enterprise resources. The documents generated in this phase include an XML user-to-role assignment sheet (XURAS) and an XML permission-to-role assignment sheet (XPRAS). These assignments are specified through XML schemas. Keeping the user, role, and permission specifications separate from their assignments allows independent design and administration of the policy, and hence supports a modular implementation of the X-GTRBAC system.

The policy sheets in the policy base are summarized in Table II. The information from the policy base is used to enforce the authorization constraints. More specifically, the users are allowed access to resources based on their assigned roles per the XURAS and the associated permissions per the XPRAS. The grammar for the specification language for the generation of these assignment documents is presented below. The corresponding schemas are provided in Appendix B in Bhatti [2003].

5.2.1 *User to Role Assignment.* The grammar for XURAS is shown in Figure 12. Each "UserRoleAssignment" (URA) tag has an associated "role_name" attribute, and contains a set of "AssignUsers" tags containing the set of users who are to be considered for potential assignment to the specified role. Each such user is identified by the "user_id" attribute of the corresponding "AssignUser" tag. This tag also contains the assignment constraint for this particular user. The assignment constraint has a "cred_type" attribute that specifies the credential type that the user must possess in order to be considered for a potential role assignment. The remaining part of the constraint is semantically

```
<!-- XML User-to-role Assignment Sheet> ::=
<XURAS [xuras_id = (id) ] >
  {<!-- User-to-role Assignment>}+
</XURAS>
```

```
<!-- User-to-role Assignment>          ::=
<URA ura_id=(id) role_name=(name)>
<[De]AssignUsers>
     {< !--[De]Assign User>}+
</[De]AssignUsers>
</URA>
```

```
<!--[De]Assign User >   ::=
<[De]AssignUser
     user_id=(id)>
 <!--[De]Assign User Constraint>
</[De]AssignUser>
```

```
<!--[De]Assign User Constraint> ::=
<[De]AssignUserConstraint
   [op = {AND|OR|NOT|XOR}]>
   <!--[De] Assign User Condition>
</[De]AssignUserConstraint>
```

```
<!--[De]Assign User Condition> ::=
<[De]AssignUserCondition
   cred_type="type_name"
   [{pt_expr_id=(id) |
    d_expr_id=(id)}] >
   [<!-- Logical Expression>]
</[De]AssignUserCondition>
```

Fig. 12. X-Grammar for XURAS.

```
<!-- XML Permission-to-role Assignment Sheet> ::=
<XPRAS [xpras_id = (id) ] >
  {<!-- Permission-to-role Assignment>}+
</XPRAS>
```

```
<!-- Permission-to-role Assignment>      ::=
<PRA pra_id=(id) role_name=(name)>
<[De]AssignPermissions>
     {< !--[De]Assign Permission>}+
</[De]AssignPermissions>
</PRA>
```

```
<!--[De]Assign Permission >   ::=
<[De]AssignPermission
     [{pt_expr_id=(id) |
    d_expr_id=(id)}]
   {<PermId>(id)</PermId>}+
</[De]AssignPermission>
```

Fig. 13. X-Grammar for XPRAS.

similar to the constraints discussed above, and is evaluated in an analogous manner. The user is assigned to the specified role if the constraint evaluates to true. Similar logic applies to deassignment of users from roles. Note that a special user with user_id = "any" is recognized by the system as an unknown user, who may be required to supply additional assignment conditions in order to be assigned to a particular role. If no explicit conditions are specified, then any user could be assigned the particular role, which usually is the "guest" role in most enterprise applications.

5.2.2 *Permission to Role Assignment.* The grammar for XPRAS is shown in Figure 13. Each "PermissionRoleAssignment" (PRA) tag has an associated "role_name" attribute and contains a set of "AssignPermission" tags containing the set of permissions that are to be potentially assigned to the specified role. Each such permission is identified by a "PermId" tag within the corresponding "AssignPermission" tag. Note that the permissions would also be subject to temporal constraints, and hence we allow the option of specification of periodic-time expression for the permission assignment. The permission is assigned to the specified role if the constraint evaluates to true. Similar logic applies to de-assignment of permissions from roles.

6. SYSTEM ARCHITECTURE AND IMPLEMENTATION

In this section, we present the system architecture of X-GTRBAC. We first provide an overview of the system components and technologies, and then discuss

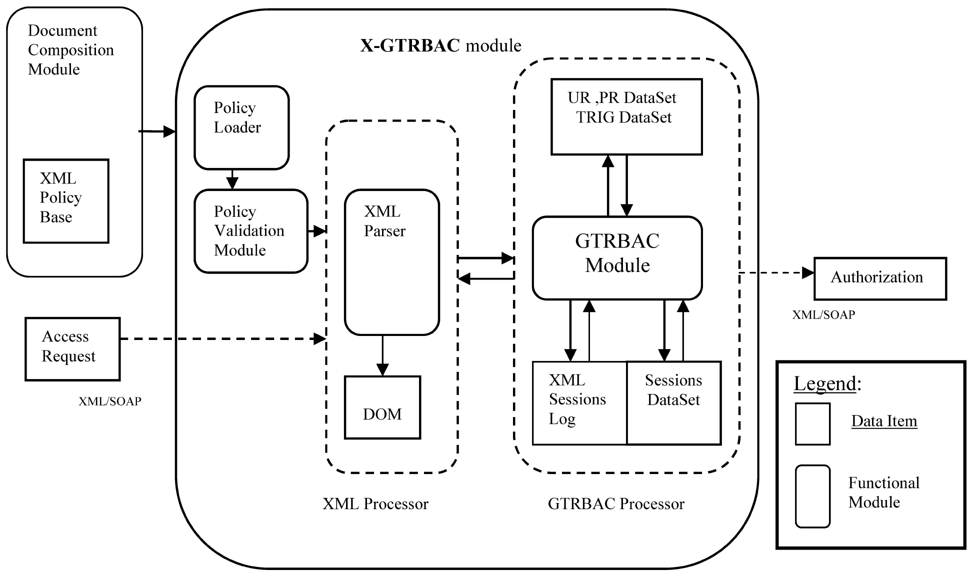


Fig. 14. X-GTRBAC system architecture.

the implementation details to illustrate the process of specification and enforcement of an enterprise's access control policy.

6.1 Overview

The X-GTRBAC framework allows the XML-based enterprise policies to be specified and enforced through a Java-based GUI-enabled application. The application code is readily integrated into a Web browser by an application-to-applet transformation mechanism provided by Java.

The overall system design is depicted in Figure 14. As indicated in the figure, the two main subsystems of X-GTRBAC module are the XML processor and the GTRBAC processor. The XML processor is implemented in Java using Java API for XML processing (JAXP). Custom modules have been designed to get the DOM instance of parsed XML documents and forward them on to the GTRBAC processor. The GTRBAC module then administers and enforces the policy according to the supplied policy information. The policy information is contained in the XML policy base. A document composition module external to X-GTRBAC is provided to compose the policy documents. This module composes the policy sheets listed in Table II. The policy sheets from the XML policy base are then loaded into the X-GTRBAC module by the security administrator. Since X-GTRBAC can act as both stand-alone and web-deployable application, it may be invoked from either the local system, or remotely through an XML-aware browser. Hence, the X-GTRBAC module seamlessly interfaces with an external client across distributed domains over an interconnect network (i.e., LAN, WAN, and so on). The client may submit an access request through any standard XML-based Web services messaging protocol, like SOAP [W3SOAP]. Similarly, the access authorization is returned via the same protocol.

6.2 XML Processor

The XML processor contains the XML parser and the DOM tree representations of the supplied XML documents. The X-GTRBAC system provides a policy loader to load the policy sheets for a given policy. As a next step, functionality is provided via a policy validation module to validate the policy sheets in terms of existence checking and type conformance. This means that all users, roles, and permissions referenced in XURAS, XPRAS, and XTrigDef sheet must exist in the corresponding XUS, XRS, and XPS, respectively. Also, all the referenced data must exist in the corresponding definition files. This means that (i) the credential types associated with the users in XUS must conform to the type definitions in the XCredTypeDef sheet, (ii) the separation of duty constraint sets referenced in the XRS must be present in XSoDDef sheet, and (iii) the periodic-time, start-time, interval, and duration expressions referenced in XRS must be present in XTempConstDef sheet. This validation support is provided by Apache Xalan XSLT engine built into JAXP. Once the policy sheets are validated, the corresponding DOM tree representation is generated and passed on to the GTRBAC processor. A facility is provided to display the instance of the DOM tree via the X-GTRBAC GUI.

6.3 GTRBAC Processor

The GTRBAC processor contains the GTRBAC module and associated data items generated by the GTRBAC module. It performs the policy administration and enforcement tasks.

6.3.1 *Policy Administration.* The GTRBAC module provides functionality to parse the DOM tree structures supplied by the XML processor and retrieves the relevant information into its internal data structures. The policy assignments are checked against the RBAC consistency rules, similar to those outlined in Gavrila and Barkley [1998], against violations of any SSD, DSD, or cardinality constraints. A consistent assignment means, for instance, that a user in question will be assigned by the GTRBAC module to the corresponding role because it satisfies all the required credential and consistency conditions. The permissions in the system are also assigned to roles under similar consistency notions. It may be noted that for all the users who have been assigned to roles, the actual role activation would occur when the user actually logs into the system and requests a role. The notion of role assignment in this context is of static type, that is, it implies that the user has been declared as assignable to the said role based on already supplied credential information. There can also be a dynamic role assignment for an unknown user based on his/her credentials supplied at the time of login. These static and dynamic policy assignments, together with the role activation and enabling rules and triggers information, create the complete internal representation of the XML policy base within the GTRBAC processor for enforcement of the policy. A collection of these policy information items are referred to as UserRole (UR) datasets, PermissionRole (PR) datasets, and TRIG dataset. A facility is provided to display the UR, PR, and TRIG datasets via the X-GTRBAC GUI.

```
<xas [xas_id= (id)] >
 <login login_id= (id)>
    [<!--CredType>]
 </login>
 ……….
 <xar xar_id= (id)>
    {<Object type= (type name) id= (id)/>}+
</xar>
<xas>
                                              (a)
```

```
<xss [xss_id= (id)] >
 <session>
  <session_id> (id) </session_id>
  <user_id> (user id) </user_id>
  <role_name> (role name) </role_name>
  <domain> (domain name) </domain>
  <login_time> (time) </login_time>
  <login_date> (date) </login_date>
  <duration> (duration) </duration>
  <active> {Yes|No} </active>
 </session>
</xss>                                        (b)
```

Fig. 15. X-Grammar for (a) XAS (b) XSS.

6.3.2 *Policy Enforcement.* The information from the internal data structures is then used by the GTRBAC module to enforce the policy and manage user sessions. The initial login into the system will create a default session for the user with a pre-specified "minimal" set of roles activated based on the supplied user credentials. The initial login can be the "user_id" from the XUS, if it is a known user, or a "user_id" of "any," as discussed above. In addition to the default set of activated roles, more roles can also be activated if the user credentials so allow. Any triggers associated with role activation or other events are handled by the GTRBAC module based on the information from the TRIG dataset. Access to resources is requested in the form of an XML access request (XAR) that specifies the "object type" and "object id" of the requested resource. An XAR could be submitted locally or remotely as an assertion in SOAP or similar XML-based messaging protocol. This access request is then evaluated based on the currently activated roles for this user. Only those resources may be accessed during a session for which the activated set of roles has associated permissions. Both the login information and XARs for a user are stored in an XML access sheet (XAS). The session-related information is contained in the sessions dataset within the GTRBAC processor. This information is extracted from an activity log maintained for every user by the GTRBAC module which we refer to as an XML sessions sheet (XSS). A session parameter is included in the XSS to record the domain from which the user is requesting access. In addition to the domain of the requesting user, the XSS also contains the attributes such as "login_time," "login_date," and "duration" of the session. These attributes are used to capture the activity profile of the user. Such information is constantly updated into the Sessions DataSet, where it can be dynamically processed, and incorporated into the access decisions. This feature is useful in certain situations where context information may be an important decision parameter, as discussed in Section 3.2. The grammar for a typical XAS and XSS is shown in Figure 15.

7. X-GTRBAC AND THE CIE

We now present a CIE application that is currently being implemented on our system, and discuss how the CIE specifications can be systematically mapped to our X-GTRBAC framework to highlight the latter's significance.

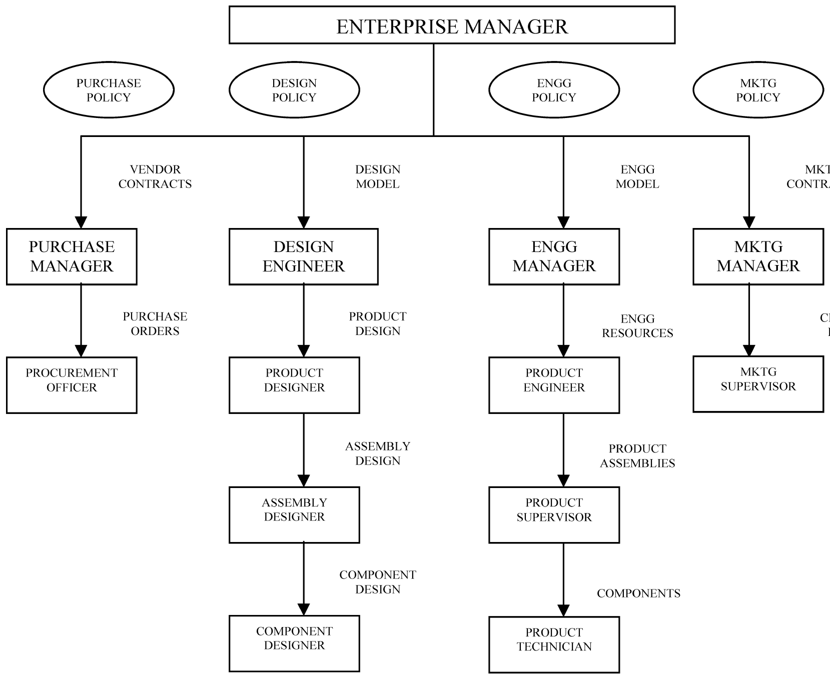


Fig. 16. The functional role hierarchy and accessed system resources at each level.

7.1 CIE Policy Specification

The access control policy for the CIE is essentially composed of the domain level policies described for each domain within the enterprise. These domain level policies capture the specifications of roles, users, permissions, and the related assignments for their respective domains. In essence, each such policy captures a minimal set of specifications that should include the following:

Functional Roles and Hierarchies. We let the roles in the CIE be represented by a functional role hierarchy that assigns, at each level of the system, a role that is needed to carry out the associated function. This role hierarchy captures the semantics of the top-down requirements interfacing[2] and bottom-up requests interfacing[3] within the enterprise [IIES]. In addition, it associates with the role at each level a set of responsibilities, and corresponding permissions to carry out those responsibilities. These responsibilities and permissions are captured in the domain level policies that are supplied according to the specific needs and requirements of the enterprise. The functional role hierarchy for the CIE in our application is shown in Figure 16. The corresponding policy name for each domain is placed in an oval above each column. The overall policy of the enterprise may then be composed of the combination of all domain level policies. Along the edges are placed the names of possible resources that are accessed by the respective roles at each subsequent level. Only a subset of relevant

[2]Interfacing requirements of management and staff to factory floor.
[3]Interfacing requests to management and staff from factory floor.

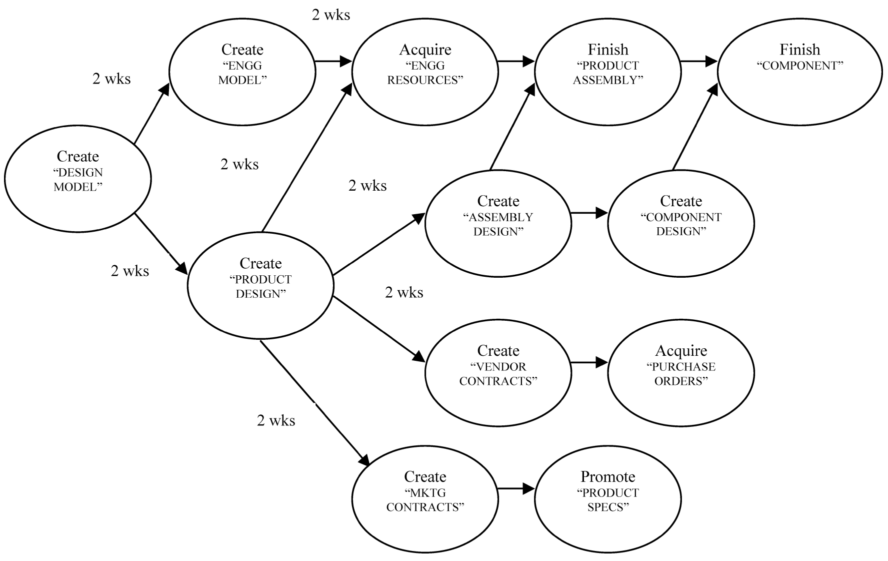


Fig. 17. The DAG representing the execution time-frame for a project within the CIE.

functional modules has been shown in the hierarchy to illustrate the applicability of X-GTRBAC policy specification framework. The hierarchy can accordingly be extended and new policies defined as per the need of a specific enterprise.

Role Enabling and Activation Constraints. The functional role hierarchy imposes a partial ordering on the timing, order, and extent of accesses by the various roles. This constitutes the temporal semantics of access control within the CIE, and could be captured by a directed acyclic graph (DAG), such as the one shown in Figure 17. This particular graph represents the execution time frame of a certain project scenario for the CIE being implemented in our system. The project requires the pooling of human, technical, commercial, and engineering resources from various domains within the CIE. The timing between two events is captured on the connecting arrows. Where it is not explicitly stated, the default duration is 1 week. Note that the total time to finished product according to the DAG is then 7 weeks. Based on the duration of the individual tasks in the project, the corresponding roles in the CIE need to be enabled and disabled. The enabled roles would have further constraints on activation in situations where there exist other activation constraints. The permissions for the enabled and assigned roles would also be constrained according to the involvement of the role in the current stage of the project. The temporal constraint specification mechanism provided by X-GTRBAC would be used to transform the temporal constraints specified by the DAG into XML policies for the CIE.

In the light of the preceding discussion on functional roles and tasks, we list in Table III a subset of constraints that is implemented in the CIE represented by the role hierarchy and DAG of Figures 16 and 17, respectively.

User Credentials. An enterprise would ordinarily supply the set of users who would typically assume one of the functional roles within the CIE, and their

Table III. A Subset of Constraints Derived from the Role Hierarchy of Figure 16 and DAG of Figure 17 for the CIE

#	Constraint Type	Role	Constraint Description
1.	Enabling	`Design Manager`	Is enabled only starting 1st week of every quarter of year 2003
2.	Activation	`Design Manager`	May be activated only by one user at a time
3.	Enabling	`Engg Manager`	Is enabled only: (i) starting 3rd week of every quarter of year 2003, and (ii) if `Design Manager` role is enabled
4.	Activation	`Engg Manager`	May be activated only if `Design Manager role` is activated
5.	Enabling	`Product Designer`	Is enabled only: (i) starting 3rd week of every quarter of year 2003, and (ii) if `Design Manager` role is enabled
6.	Activation	`Product Designer`	May be activated only if `Design Manager role` is activated
7.	Enabling	`Product Engineer`	Is enabled only: (i) starting 5th week of every quarter of year 2003, and (ii) if `Product Designer AND Engg Manager` role is enabled
8.	Activation	`Product Engineer`	May be activated only if Product `Designer AND Engg Manager role` is activated
9.	Enabling	`Purchase Manager/ Marketing Manager`	Is enabled only: (i) starting 5th week of every quarter of year 2003, and (ii) if `Product Designer` role is enabled
10.	Activation	`Purchase Manager/ Marketing Manager`	May be activated only if Product `Designer role` is activated
11.	Static Separation of Duty (SSoD)	`Purchase Manager` and `Marketing Manager`	Both these roles may not be assigned to the same user at any given time
12.	Dynamic Separation of Duty (DSoD)	`Product Designer` and `Product Engineer`	Both these roles may not be simultaneously active in the same session by the same user
13.	Trigger	All roles	All active roles are deactivated at the start of 8th week of every quarter of year 2003

associated set of credentials that may be used in determining their assignment to particular roles. Without loss of generality, we list in Table IV a subset of the users we consider in our example, along with their associated credentials. We assume for simplicity the convention that the credential types are named so as to reflect the current level of responsibility, or role, held by the user. In general, they may be named differently from the actual role name of the user. It should also be noted that the credential expression for a user with more than one different credential types (such as `george` in Table IV) is the union of the credential expressions of each of those credential types.

Role Assignment. The roles are assigned to users consistent with their supplied credentials. Such assignments may be constrained by temporal or

Table IV. A Subset of Users and Associated Credentials for the CIE

#	User Id	Credential Type	Credential Expression
1.	john	Product Designer	age = 39, level = B, qualification = MS
2.	nancy	Product Engineer	age = 36, experience = 15, qualification = MS
3.	george	Assembly Designer Product Supervisor	age = 29, level = D, experience = 5, qualification = BS
4.	carla	Product Supervisor	age = 28, experience = 5, qualification = BS
5.	smith	Procurement Officer	age = 32, level = B, region = northeast
5.	dorothy	Procurement Officer Marketing Supervisor	age = 34, level = C, experience = 20, region = midwest

Table V. A Subset of Role Assignments in the CIE

#	Role	User Id	Credential Type	Assignment Condition
1.	Design Manager	john	Product Designer	age>35 or level = A, qualification = PhD
2.	Engg Manager	nancy	Product Engineer	age>35 or experience>10, qualification = MS
3.	Product Designer	george	Assembly Designer	age>20 or level = B, qualification = BS
4	Product Engineer	george carla	Product Supervisor	age>20 or experience = 5, qualification = BS
5.	Purchase Manager	smith dorothy	Procurement Officer	age>30 or level = B, region = midwest
6.	Marketing Manager	dorothy	Marketing Supervisor	age>30 or experience>10, region = midwest

Table VI. A Subset of Available Permissions in the CIE

#	Permission ID	Object ID	Object Type	Allowed Operation
1.	P1	Design Model	Document	All
2.	P2	Design Model	Document	Read
3.	P3	Engg Model	Document	All
4.	P4	Engg Model	Document	Read
5.	P5	Product Design	Document	All
6.	P6	Product Design	Document	Read
7.	P7	Engg Resources	Material Equipment	Operate
8.	P8	Vendor Contracts	Document	All
9.	P9	Marketing Contracts	Document	All

nontemporal context conditions. Once again, without loss of generality, a subset of the role assignments considered in our example is listed in Table V.

Permissions. The available permissions within the CIE represent the set of operations that may be performed on the available enterprise resources by eligible roles. The specification of these permissions is system dependent. We list in Table VI a subset of the permissions assumed to be typically available in our example.

Permission Assignment. The permission assignment determines the extent of access of various roles within the CIE. The roles are assigned permissions consistent with their responsibilities within the CIE. Such assignments may be constrained by temporal or nontemporal context conditions. A typical set

Table VII. A Subset of Permissions Assignments in the CIE

#	Role	Permission ID	Assignment Condition
1.	`Design Manager`	`P1`	Is assigned starting 1st week of every quarter of year 2003 for 6 weeks
2.	`Engg Manager`	`P2`	Is assigned starting 3rd week of every quarter of year 2003 for 2 weeks
		`P3`	Is assigned starting 3rd week of every quarter of year 2003 for 4 weeks
3.	`Product Designer`	`P2`	Is assigned starting 3rd week of every quarter of year 2003 for 4 weeks
		`P5`	Is assigned starting 3rd week of every quarter of year 2003 for 4 weeks
4.	`Product Engineer`	`P4`	Is assigned starting 5th week of every quarter of year 2003 for 2 weeks
		`P6`	Is assigned starting 5th week of every quarter of year 2003 for 1 week
		`P7`	Is assigned starting 5th week of every quarter of year 2003 for 2 weeks
5.	`Purchase Manager`	`P2`	Is assigned starting 5th week of every quarter of year 2003 for 1 week
		`P8`	Is assigned starting 5th week of every quarter of year 2003 for 2 weeks
6.	`Marketing Manager`	`P2`	Is assigned starting 5th week of every quarter of year 2003 for 1 week
		`P9`	Is assigned starting 5th week of every quarter of year 2003 for 2 weeks

of permission assignments for the CIE considered in our example is listed in Table VII.

We capture the mapping of enterprise specifications to X-GTRBAC framework in Table VIII. The table lists the functions and tasks within the CIE and the corresponding component that is responsible for it in the X-GTRBAC system. We next outline the process of representing this CIE policy in our X-GTRBAC framework.

7.2 A CIE X-GTRBAC Policy

The specification language discussed in Section 5 can be used to compose the policy sheets for the CIE based on the mapping given in Table VIII. The policy specification is then loaded into our implemented system for enforcement. Presented below is a discussion of the composition and implementation of the policy in our X-GTRBAC framework.

7.2.1 *Policy Definition Sheets.* In order to supply the necessary information needed to enforce an access control policy, the security administrator of the CIE loads the basic policy definitions related to credential types, separation of duty constraints, temporal constraints, and trigger specification. The policy definition sheets containing this information are shown in Figures 18–21.

Table VIII. The Mapping of CIE Specifications to X-GTRBAC Framework

CIE Function/Task	Responsible X-GTRBAC Module	Related X-GTRBAC Data Element
Specify Users and Credentials	Policy Loader	XUS, XCredTypeDef
Specify Functional Roles and Hierarchy	Policy Loader	XRS, XSoDDef
Specify Available Permissions	Policy Loader	XPS
Specify Task Scheduling and Timing (DAG)	Policy Loader	XTempConstDef
Specify Task Dependencies	Policy Loader	XTrigDef
Specify User Eligibility for Functional Roles	Policy Loader	XURAS
Specify Permission Criteria for Functional Roles	Policy Loader	XPRAS
Validate the Enterprise Policy	Policy Validation Module	Entire XML Policy Base
Generate Enterprise Policy Documents	XML Processor	Internal DOM Tree Structure
Create User-to-role / Permission-to-role Mapping	GTRBAC Module	UR and PR DataSets
Create and Maintain User Sessions	GTRBAC Module	XSS, Sessions DataSets
Request Access to Enterprise Resource	External GUI (Local or Remote)	XAR
Enforce Access Control Policy	GTRBAC Processor	UR, PR, Sessions DataSets

XCredTypeDef:

```
<?xml version="1.0" encoding="UTF-8"?>
<XCredTypeDef xctd_id="CIE_XCTD">
 <CredentialType cred_type_id="cPD"
       type_name="Product Designer">
  <AttributeList>
       <AttributeName type="integer"
          usage="mand">age</AttributeName>
       <AttributeName type="string"
          usage="mand">level</AttributeName>
       <AttributeName type="integer"
          usage="mand">qualification
         </AttributeName>
  </AttributeList>
 </CredentialType>
  ................
</XCredTypeDef>
```

Fig. 18. Part of the XCredTypeSheet to define the user credentials specified in Table IV.

7.2.2 *Primary Policy Sheets.* The security administrator next creates the primary policy sheets related to the users, roles, permissions, user to role assignments, and permission to role assignments. These sheets refer to the supplemental information provided by the policy definition sheets to specify an elaborate set of temporal and nontemporal context-based constraints for the enterprise access control policy. As discussed in Section 6, the information from both these sets of sheets is read into the X-GTRBAC module to constitute a complete representation of the XML policy base for policy enforcement. The primary policy sheets for the CIE are shown in Figures 22–26.

XSoDDef:

```
<?xml version="1.0" encoding="UTF-8"?>
<XSoDDef xsod_id="CIE_XSOD">
 <SSD_Role_Sets>
  <SSD_Role_Set SSD_Role_Set_id="SSD1"
                SSD_cardinality="1">
     <SSD_Role>Purchase Manager</SSD_Role>
     <SSD_Role>Marketing Manager</SSD_Role>
   </SSD_Role_Set>
 <DSD_Role_Sets>
  <DSD_Role_Set DSD_Role_Set_id="DSD1"
                DSD_cardinality="1">
     <DSD_Role>Product Designer</DSD_Role>
     <DSD_Role>Product Engineer</DSD_Role>
   </DSD_Role_Set>
 </DSD_Role_Sets>
</XSoDDef>
```

Fig. 19. The XSoDDef sheet to define the separation of duty constraints specified in Table III.

XTempConstDef:

```
<?xml version="1.0" encoding="UTF-8"?>
<XTempConstDef xtcd_id="CIE_XTCD">
 <IntervalExpr i_expr_id="Year2003">
       <begin>1/1/2003</begin>
       <end>12/31/2003</end>
  </IntervalExpr>
  <DurationExpr d_expr_id="SixWeeks">
       <cal>Weeks</cal>
       <len>6</len>
  </DurationExpr>
  ..........
  <PeriodicTimeExpr pt_expr_id="PTQuarterWeekOne"
       i_expr_id="Year2003">
    <StartTimeExpr>
       <Year>all</Year>
       <MonthSet>
           <Month>1</Month>
           <Month>4</Month>
           <Month>7</Month>
           <Month>10</Month>
       </MonthSet>
       <WeekSet>
           <Week>1</Week>
       </WeekSet>
    </StartTimeExpr>
  </PeriodicTimeExpr>
 ............
</XTempConstDef>
```

Fig. 20. Part of the XTempDefSheet to define the constraints specified in Table III.

In particular, note that (i) the credential expression shown in XUS of Figure 22 captures the credential expression #1 in Table IV for user `john`, and (ii) the activation and enabling onstraints on `Design Manager` role in the XRS shown in Figure 23 capture the constraints #1 and #2 of Table III. Similarly, the reference to the dynamic separation of duty role set in `Product Designer` role captures the constraint #12 of Table III. Also note that (i) the role assignment shown in XURAS of Figure 25 captures the assignment condition #1 in Table V for user `john`, and (ii) the permission assignments shown in XPRAS of Figure 26

XTrigDef:

```
<?xml version="1.0" encoding="UTF-8"?>
<XTrigDef xtd_id="CIE_XTD">
 <Trigger trig_id = "disableAll">
   <Head role_name = "all"
        action = "disable" >
   </Head>
   <Body>
    <TrigConstraint>
      <TrigCondition
       pt_expr_id="PTQuarterWeekEight"/>
    </TrigConstraint>
   </Body>
 </Trigger>
</XTrigDef>
```

Fig. 21. The XTrigDef sheet to define the trigger specified in Table III.

XUS:

```
<?xml version="1.0" encoding="UTF-8"?>
<XUS xus_id="CIE_XUS">
<User user_id="john">
   <UserName>John</UserName>
   <CredType cred_type_id="cPD"
        type_name="Product Designer">
     <CredExpr>
        <age>39</age>
        <level>B</level>
        <qualification>MS</qualification>
     </CredExpr>
   </CredType>
   <MaxRoles>2</MaxRoles>
  </User>
 ..........
</XUS>
```

Fig. 22. Part of the XUS to define the users specified in Table IV.

capture the assignment conditions #1 and #4 in Table VII for `Design Manager` and `Product Engineer` roles, respectively.

7.3 Implementation Experiences

This section discusses our implementation experiences with the CIE example on our working prototype system.

The policy sheets are loaded into the X-GTRBAC system through the policy loader module. The XML processor loads the policy sheets as DOM, and the GTRBAC module stores the policy information from the DOM into internal system data structures. Based on this information, some of the policy assignments effected by the GTRBAC processor are shown in Figure 27. In particular, we note the following:

(i) `john` has not been assigned the `Product Designer` role since he does not have the required qualification (`PhD`).

(ii) `nancy` has been assigned the `Engg Manager` role.

XRS:

```
<?xml version="1.0" encoding="UTF-8"?>
<XRS xrs_id="CIE_XRS">
  <Role role_id="rDM" role_name="Design Manager">
   <Junior>Product Designer</Junior>
   <Cardinality>1</Cardinality>
     <EnabConstraint>
      <EnabCondition
         pt_expr_id="PTQuarterWeekOne"/>
    </EnabConstraint>
    <ActivConstraint>
      <ActivCondition>
        <LogicalExpr>
        <Predicate>
         <Operator>eq</Operator>
         <NameParam type=role>Design Manager
           </NameParam>
         <FuncParam>activated</FuncParam>
         <Value_param>false</ValueParam>
        </Predicate>
       </LogicalExpr>
      </ActivCondition>
    </ActivConstraint>
  </Role>
  <Role role_id="rPD" role_name="Product
         Designer">
    <DSD_Role_Set_id>DSD1</DSD_Role_Set_id>
    <Senior>Design Manager</Senior>
    <Junior>Assembly Designer</Junior>
     <EnabConstraint>
      <EnabCondition
         pt_expr_id="PTQuarterWeekThree"/>
    </EnabConstraint>
  </Role>
  .......
</XRS>
```

Fig. 23. Part of the XRS to define the roles illustrated in the role hierarchy of Figure 16, and capture the constraints on them specified in Table III.

XPS:

```
<?xml version="1.0" encoding="UTF-8"?>
<XPS xps_id="CIE_XPS">
 <Permission perm_id="P1">
        <Object object_type="Document"
         object_id="DesignModel"/>
        <Operation>all</Operation>
 </Permission>
 ...........
 <Permission perm_id="P7">
        <Object
         object_type="MaterialEquipment"
         object_id="EnggResources"/>
        <Operation>operate</Operation>
 </Permission>
 ...........
</XPS>
```

Fig. 24. Part of the XPS to define the permissions specified in Table VI.

(iii) `dorothy` is assignable to both `Purchase Manager` and `Marketing Manager` roles. However, she can be assigned only to one of them (a policy validation rule) because of the static separation of duty constraint #11 in Table III.

XURAS:

```
<?xml version="1.0" encoding="UTF-8"?>
<XURAS xuras_id="CIE_XURAS">
 <URA ura_id="uraDM" role_name="Design
    Manager">
  <AssignUsers>
   <AssignUser user_id="john">
    <AssignConstraint>
      <AssignCondition cred_type="Product
    Designer">
       <LogicalExpr op="AND">
        <Predicate>
         <LogicalExpr op="OR">
           <Predicate>
             <Operator>gt</Operator>
             <NameParam>age</NameParam>
             <ValueParam>35</ValueParam>
           </Predicate>
           <Predicate>
             <Operator>eq</Operator>
             <NameParam>level</NameParam>
             <ValueParam>A</ValueParam>
           </Predicate>
         </LogicalExpr>
        </Predicate>
        <Predicate>
         <Operator>eq</Operator>
          <NameParam>qualification</NameParam>
          <ValueParam>PhD</ValueParam>
        </Predicate>
       </LogicalExpr>
      </AssignCondition>
    </AssignConstraint>
   </AssignUser>
  </AssignUsers>
 </URA>
.......
</XURAS>
```

Fig. 25. Part of the XURAS to define the role assignments specified in Table V.

(iv) `george` has been assigned to both `Product Designer` and `Product Engineer` roles, however, he may only have one of them activated at any given time due to the dynamic separation of duty constraint #12 in Table III. This follows from the notion of role assignment and activation as treated in our framework.

(v) `smith` cannot assume the role of `Purchase Manager` since his region is not `midwest`.

(vi) `nancy` has been authorized the permission `P7` by virtue of being senior to the `Product Engineer` role.

Note that the permissions of the roles within the CIE are constrained according to the policy specification by including the duration expression within the permission assignment constraints. The fact that all roles need to be disabled after the specified project duration expires will be handled by the `disable` action trigger that would fire at the start of the 8th week to disable all roles. In case a role is activated up to the end of specified duration, the semantics

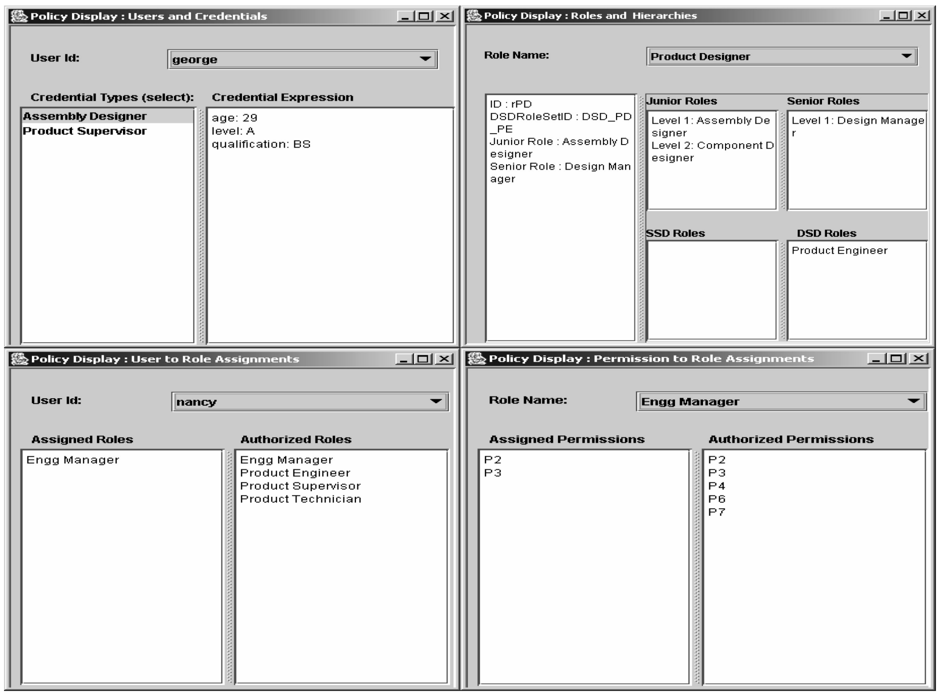
XPRAS:

```
<?xml version="1.0" encoding="UTF-8"?>
<XPRAS xpras_id="CIE_XPRAS">
 <PRA pra_id="praDM" role_name="Design
    Manager">
  <AssignPermissions>
   <AssignPermission d_expr_id="SixWeeks">
    <PermId>P1</PermId>
   </AssignPermission>
  </AssignPermissions>
 </PRA>
.......
<PRA pra_id="praPE" role_name="Product
    Engineer">
  <AssignPermissions>
   <AssignPermission d_expr_id="TwoWeeks">
    <PermId>P4/PermId>
   </AssignPermission>
   <AssignPermission d_expr_id="OneWeek">
    <PermId>P6</PermId>
   </AssignPermission>
   <AssignPermission d_expr_id="TwoWeeks">
    <PermId>P7</PermId>
   </AssignPermission>
  </AssignPermissions>
 </PRA>
.......
</XPRAS>
```

Fig. 26. Part of the XPRAS to define the permission assignments specified in Table VII.

of GTRBAC model require that the trigger first deactivates the role, and then disables it. It may be mentioned that a role may also have explicit duration constraints if it so requires. Also indicated in the figures are the authorized roles and permissions that are acquired by virtue of the role hierarchy.

The policy administration process thus creates a complete internal representation of the specified enterprise policy. The policy enforcement phase then uses this information to allow the users to create sessions and access permitted resources. The various context conditions supplied within the activation constraints are then evaluated to make access control decisions, as discussed in Section 6. We invoke individual sessions for the users, and apply a three-level security mechanism to effectively enforce the access control policy: (i) the user may only activate a role if he/she already meets the assignment criteria for it, and this restriction is imposed through the X-GTRBAC GUI by allowing only the assigned roles to appear in a drop down list of roles to choose from; (ii) the role activation goes through only if the role is enabled at that particular instance; (iii) when in an activated role, the user is restricted to request access to only those resources that the activated roles for the user have permission on, and this restriction is also imposed by allowing a selection from a drop down list of available accessible resources corresponding to the assigned permissions of the activated role. Hence, the three stage security mechanism ensures the enforcement of access control policy by restricting user access to only his/her available set of resources, and preventing any possibility for even requesting access to any unauthorized resource.


Policy Display : Users and Credentials

User Id: george

Credential Types (select): Assembly Designer, Product Supervisor

Credential Expression:
age: 29
level: A
qualification: BS

Policy Display : Roles and Hierarchies

Role Name: Product Designer

ID : rPD
DSDRoleSetID : DSD_PD_PE
Junior Role : Assembly Designer
Senior Role : Design Manager

Junior Roles: Level 1: Assembly Designer; Level 2: Component Designer

Senior Roles: Level 1: Design Manager

SSD Roles:

DSD Roles: Product Engineer

Policy Display : User to Role Assignments

User Id: nancy

Assigned Roles: Engg Manager

Authorized Roles: Engg Manager, Product Engineer, Product Supervisor, Product Technician

Policy Display : Permission to Role Assignments

Role Name: Engg Manager

Assigned Permissions: P2, P3

Authorized Permissions: P2, P3, P4, P6, P7

Fig. 27. Snapshots of policy display, clockwise from top left: (i) User credentials for george, (ii) information for product designer role, (iii) user to role assignment for nancy, and (iv) permission to role assignment for engg manager role.

The updated documentation, XML schemas and downloadable releases of X-GTRBAC system can be accessed at our project website at http://shay.ecn.purdue.edu/∼iisrl/ccgtrbac.html.

8. RELATED WORK

The specification of security policies for enterprises has recently emerged as an active research area. The advent of the RBAC model has generally been hailed as a promising sign in the industry and research community for its potential to simplify authorization administration in large enterprises. Related work has broadly spanned the aspects of both presenting system architecture and implementation of RBAC-based technologies for enterprise security administration, and complementing the RBAC model to introduce extended policy specification frameworks. In the following, we summarize the efforts in both these directions, and then highlight the significance of our particular work.

Several approaches have been presented in the literature to address various aspects of security administration within an enterprise. Applications of RBAC policies to workflow systems [Bertino et al. 1999b] and Web services [Bhatti et al. 2004b] have been proposed. An XML-based approach to specify enterprise RBAC policies has been reported in Vuong et al. [2001]. Ferraiolo et al. [1999] use the RBAC model to address access control needs of enterprise computing environments. They have attempted to present an RBAC-based approach as an alternative to ACL-based access control scheme used within a corporate intranet, and have illustrated its use through a reference implementation. Kern [2002] presents an ERBAC model for enterprise-wide role-based access control. Their model uses the notion of enterprise roles with parameterization and is reported by the authors as being helpful in reducing the administration effort required to maintain users and their access rights in large enterprises. They have augmented their work with the discussion of a commercial security administration tool. All these schemes, however, are not suited to enterprises with dynamic content- and context-aware access control requirements, such as the one illustrated through the example in the paper, since they provide no explicit mechanism to support evaluation of dynamic user credentials and context conditions. The discussion on implementation in Kern [2002] talks about interfaces to consolidate information from constituent subsystems of the enterprise in a common security repository using a system independent conceptual model. This repository may be similar to our XML policy base of Section 5. However, the design of interfaces would actually determine the degree to which their system would actually be able to support the heterogeneity of the subsystems within a large CIE.

There has been an effort in the research community to complement the RBAC model with additional features to allow extended policy specification frameworks. A temporal extension to RBAC has been presented in the TRBAC model [Bertino et al. 2001]. TRBAC supports the specification of temporal constraints on role enabling, and the dependencies among them, and hence provides a mechanism to enforce time-dependent access control. A generalized GTRBAC model presented in Joshi et al. [2005] is capable of providing a wider range of temporal constraints, including periodic as well as duration constraints on user-to-role assignments, permission-to-role assignments, and role activation. The GTRBAC model extends the syntactic structure of TRBAC; however, the notion of user credentials is not supported in both these schemes to allow the assignment of authenticated users to a particular role.

Some recent work has been reported in XML-based security and context-aware access control. Two prominent security specifications emerging from the industrial community are the Security Assertions Markup Language (SAML) [XML Coverpage 2003b] and the XML Access Control Language (XACML) [XML Coverpage 2003a]. While SAML is directed primarily as an authentication mechanism, XACML is intended for Web-based authorization. XACML [XML Coverpage 2003a] specification is based on an extension of XML to define an access control specification that supports notions similar to those of user credentials and context-based privilege assignments. It, however, does not directly support the notion of roles, and hence lacks the essential features as separation of duty constraints, role hierarchy, and cardinality. The absence of roles also prohibits the provision of a comprehensive mechanism to supply and evaluate sophisticated temporal constraints on assignments of users to privileged tasks, since direct user-to-permission assignments violate the very principles of scalability and manageability that motivated the use of GTRBAC (see Section 3.2). The OASIS model for active security presented in Bacon et al. [2002] addresses the context-aware access control requirements within large scale systems. It is an extension of the RBAC model with parameters based on first order logic, and allows fine-grained evaluation of dynamic user credentials and context conditions. The parameter-based evaluation of context predicates is similar to our notion of evaluating logical expressions with predicates and parameters explicitly supplied via the XML policy sheets. The discussion on environmental predicates closely resembles that of the temporal constraints provided in our framework. The paper emphasizes on the formal logic-based semantics of the model with its own merits, but it does not detail any implementation architecture to enforce the same. The implementation, however, is related by the authors to a middleware architecture that supports asynchronous events, and requires a mechanism that allows the communicating systems to acquire support for asynchronous operations. Although this scheme is designed to be scalable and manageable for distributed environments, the fact that it relies on extending the client's capabilities to make them able to communicate with the OASIS server adds significant overhead to its wide-scale deployment. One such scenario would be to configure it to allow heterogeneous, distributed systems to interoperate in the afore-referred generic CIE environment. Since our framework is entirely XML based, our approach allows for adopting the XML-based middleware architecture [W3 SOAP] that is emerging as a widely accepted standard for communication among distributed applications, and thus significantly reduces the burden attached thereto. Our system hence captures the combined semantics of both XACML and OASIS models and achieves secure enterprise-wide interoperation with dynamic fine-grained access control. To the best of our knowledge, an XML-based GTRBAC approach to address enterprise-wide access control similar to the one we have presented in this paper has not been previously investigated.

9. CONCLUSIONS

In this paper, we have highlighted the challenges for enterprise-wide access control and presented X-GTRBAC framework, an XML-based specification language based on the GTRBAC model and its implementation, which addresses them. Our specification language provides compact representation of access control policies for a generic CIE, while incorporating the security relevant features that have been motivated in this paper to allow content-based, context-aware access control. The language conforms to the GTRBAC model, and hence incorporates the features from the NIST RBAC model and the temporal extensions proposed thereupon. We have emphasized separation of language schemas to provide efficient specification of definitions of RBAC elements, user-to-role and permission-to-role assignments, hierarchical and separation of duty constraints, and an elaborate set of temporal constraints. Such separation allows for an extensible design of the enterprise access control policy. Our specification language additionally augments GTRBAC with attribute-based credential and constraint specification mechanism which allows capturing contextual information is user-to-role and permission-to-role assignments. The language can be used to specify GTRBAC policies for securing heterogeneous, distributed enterprise resources, and allows dynamic evaluation of user credentials and context information to provide fine-grained access control. An implementation based on Java has also been presented, and the system architecture has been illustrated to highlight its salient features. Our framework hence allows an enterprise's access control policy to be expressed in XML, and enforced through the X-GTRBAC system module. A comprehensive example has been discussed to motivate and illustrate the applicability of the model to a generic CIE. We have provided the specifications for an enterprise access control policy, and a mapping thereof to our framework. Our implementation experiences have also been presented on our working prototype system.

The complexity of policy administration done in a centralized manner can be significantly high in a large enterprise. We have recently proposed a decentralized administration model for the X-GTRBAC framework [Bhatti et al. 2004c]. The decentralized model uses the notion of administrative domains to decentralize the policy administration tasks. We plan to incorporate support for the administrative concepts outlined in that work in our existing X-GTRBAC prototype.

A major direction for future will be extending our X-GTRBAC framework to distributed interenterprise environments. This poses several challenges, the key amongst them being semantic heterogeneity management. Each individual system of a multienterprise environment can have its own access control policy at a local level, and the integration of these local policies entails various challenges regarding reconciliation of semantic differences between local policies, secure interoperability, containment of risk propagation, global level policy management, and so on [Joshi et al. 2001]. When local policies of individual enterprises are integrated to generate a global policy or metapolicy that governs the rules for access mediation within a multienterprise environment, semantic differences and inconsistencies among the local policies must be resolved in order to ensure secure interoperation. Efficient administration and management of global level policy becomes challenging, particularly as local policies evolve with time. We plan to explore the promise of our XML-based GTRBAC framework for its support for information management and access control in distributed systems to handle the semantic heterogeneity challenges posed by such environments.

REFERENCES

Bacon, J., Moody, K., and Yao, W. 2002. A model of OASIS role-based access control and its support for active security. *ACM Transactions on Information and System Security*, *5*, 4 (Nov.).

Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. 1998. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems 23*, 3 (Sept).

Bertino, E., Bonatti, P., and Ferrari, E. 2001. TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security 4*, 3 (Aug.).

Bertino, E., Castano, S., and Ferrari, E. 2001. Securing XML documents with author X. *IEEE Internet Computing 5*, 3 (May-June).

Bertino, E., Castano, S., Ferrari, E., and Mesiti, M. 1999a. Controlled access and dissemination of XML documents. In *Workshop on Web Information and Data Management*, Kansas City, MI, Nov. 2–6.

BERTINO, E., FERRARI, E., AND ATLURI, V. 1999b. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security 2*, 1 (Feb.).

BHATTI, R. 2003. X-GTRBAC: An XML-based policy specification framework and architecture for enterprise-wide access control. Masters Thesis, Purdue University. Available as CERIAS Technical Report 2003-27.

BHATTI, R., BERTINO, E., AND GHAFOOR, A. 2004a. *Towards Improved Federated Identity and Privilege Management in Open Systems*. CERIAS Technical Report 2004-32.

BHATTI, R., JOSHI, J. B. D., BERTINO, E., AND GHAFOOR, A. 2004b. XML-based RBAC policy specification for secure Web-services. *IEEE Computer 37*, 4 (Apr.).

BHATTI, R., JOSHI, J. B. D., BERTINO, E., AND GHAFOOR, A. 2004c. X-GTRBAC admin: A decentralized adminstration model for enterprise wide access control. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, Yorktown, Heights, NY, June 2–4.

FERRAIOLO, D. F., BARKLEY, J. F., AND KUHN, D. R. 1999. A role based access control model and reference implementation within a corporate Intranet. *ACM Transactions on Information and System Security 2*, 1 (Feb.).

FERRAIOLO, D. F., GILBERT, D. M., AND LYNCH, N. 1993. An examination of federal and commercial access control policy needs. In *Proceedings of NISTNCSC National Computer Security Conference*, Baltimore, MD, Sept. 20–23.

FERRAIOLO, D. F., SANDHU, R., GAVRILA, S., RICHARD KUHN, D., AND CHANDRAMOULI R. 2001. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security 4*, 3 (Aug.).

FERRAIOLO, D., SANDHU, R., GAVRILA, S., KUHN, R., AND CHANDRAMOULI, R. 2000. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of the 5th ACM Workshop on Role-Based Control*, Berlin, Germany, July 26–28.

GAVRILA, S. I. AND BARKLEY, J. F. 1998. Formal specification for role based access control user/role and role/role relationship management. In *Proceedings of the 3rd ACM Workshop on Role-Based Access Control*, Fairfax, VA, Oct. 22–23.

IBM. Why XML schema beats DTDs hands-down for data. http://www-106.ibm.com/developerworks/xml/library/x-sbsch.html.

IIES. Purdue reference model for computer integrated manufacturing. http://iies.www.ecn.purdue.edu/IIES/PLAIC/PERA/ReferenceModel/index.html.

ISO. 1986. Standard Generalized Markup Language (SGML). ISO 8879. Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML).

JAVA COMMERCE. XML tutorial. http://www.javacommerce.com/tutorial/xmlj/intro.htm.

JTENENBG. Overview of enterprise computing. http://faculty.washington.edu/jtenenbg/courses/455/s02/sessions/ec_overview.ppt.

JOSHI, J. B. D., BERTINO, E., LATIF, U., AND GHAFOOR, A. 2005. A generalized temporal role based access control model (GTRBAC). *IEEE Transaction on Knowledge and Data Engineering 17*, 1 (Jan.). Also available as CERIAS Technical Report 2001-47.

JOSHI, J. B. D., GHAFOOR, A., AREF, W., AND SPAFFORD, E. H. 2001. Digital government security infrastructure design challenges. *IEEE Computer 34*, 2 (Feb.).

KERN, A. 2002. Advanced features for enterprise-wide role-based access control. In *Annual Computer Security Applications Conference*, Las Vegas, NV, Dec. 9–13.

NIEZETTE, M. AND STEVENNE, J. 1992. An efficient symbolic representation of periodic time. In *Proceedings of 1st International Conference on Information and Knowledge Management*, Baltimore, MD, Nov. 8–11.

OSBORN, S. L., SANDHU, R., AND MUNAWER, Q. 2000. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security 3*, 2 (Feb.).

SANDHU, R., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. 1996. Role based access control models. *IEEE Computer 29*, 2 (Feb.).

VUONG, N. N., SMITH, G. S., AND DENG, Y. 2001. Managing security policies in a distributed environment using eXtensible markup language (XML). In *Symposium on Applied Computing*, Las Vegas, NV, Mar. 11–14.

W3SOAP. Simple object access protocol (SOAP) 1.1. http://www.w3.org/TR/SOAP/.

W3. W3C XML schema. www.w3.org/XML/Schema.

Web Reference. Web services XML's role. http://www.webreference.com/js/tips/011028.html.

XML. 2000. eXtensible Markup Language (XML) 1.0 (Second). W3C Recommendation 6 October 2000. http://www.w3.org/TR/REC-xml.

XML Coverpages. 2003a. XACML 1.0 specification. http://xml.coverpages.org/ni2003-02-11-a.html.

XML Coverpages. 2003b. SAML 1.0 specification. http://xml.coverpages.org/ni2003-05-27-b.html.

XML Coverpages. 2004. OASIS RBAC announcement. http://xml.coverpages.org/ni2004-04-05-a.html.

XPath. 2002. XML Path Language (XPath) 2.0. Working Draft 16 August 2002. http://www.w3.org/TR/xpath20/.

Received April 2003; revised September 2004; accepted September 2004