Applying Role-Based Access Control to Collaborative Web Portals

Steven A. Demurjian
University of Connecticut
371 Fairfield Way
Storrs, CT 06269-2155
steve@engr.uconn.edu

Haiying Ren
University of Connecticut
371 Fairfield Way
Storrs, CT 06269-2155
haiying.ren@pw.utc.com

Solomon Berhe
University of Connecticut
371 Fairfield Way
Storrs, CT 06269-2155
solomon.berhe@engr.uconn.edu

M. Devineni
Serebrum Cooperation
555 US Highway Route 1
Iselin, NJ 08830
mahitha@serebrum.com

S. Kopparti
Serebrum Cooperation
555 US Highway Route 1
Iselin, NJ 08830
surekha@serebrum.com

K. Polineni
Serebrum Cooperation
555 US Highway Route 1
Iselin, NJ 08830
krishna@serebrum.com

ABSTRACT

Collaborative portals are emerging as a viable technology to allow groups of individuals to easily author, create, update, and share content via easy-to-use web-based interfaces. Freeware and open source products (e.g., SourceForge's MediaWiki) as well as commercial solutions (e.g., Microsoft's Sharepoint) are prevalent in today's marketplace. From a security perspective, these products are often very limited and coarse grained in both their authorization and authentication. For example, in the case of Wikis, the security model ranges from anonymous users (no authorization and limited access) to the feature-rich system via registration with rigid roles, super users, and read-only browser. In the latter case, once access via registration is granted, that access is often the full range of actions and capabilities that are available in the Wiki. However, in practice, such full access may not be appropriate for all applications, particularly as the collaborative technology moves into other domains such as health care (which have stringent HIPAA requirements). In this paper, we report on our research and development effort of a hierarchical role-based access control for collaborative web portals that encompasses and realizes security at the application level, the document level (authoring and viewing), and the look-and-feel of the portal itself.

Categories and Subject Descriptors

H.4 [**Collaborative Web Portals,**]: Miscellaneous; D.2.8 [**Access Control Models**]: [Security Administration]Security and Protection

Keywords

RBAC, Wiki, Access Control Design

1. INTRODUCTION

Over the past decade, the World Wide Web (WWW) has come to the forefront as a viable means to allow individuals and organizations to collaborate. As a result, web portals have emerged as a means to facilitate these interactions, for many different purposes, ranging from information repositories to full-fledged authoring and document collaboration. For instance, Web sites like WebMD (www.webmd.com) and Wikipedia (www.wikipedia.org) are utilized by unregistered users to browse existing content on an array of different topics via easy-to-use web-based interfaces. At the other end of the spectrum, for registered users, these web portals provide a means to author, create, modify, and track documents of all types within a consistent framework or infrastructure. A registered user of Wikipedia has the ability to create new document content and modify existing content. In fact, freeware and open source products such as SourceForge's Mediawiki or a commercial solution such as Microsoft's Sharepoint allows any individual with sufficient expertise to generate their own web portal to meet specific purposes and needs.

However, from a security perspective, these products are often very limited in the level of protection that is offered to information content that is created and uploaded to these various sites. For example, a registered Wikipedia user could create and upload intentionally erroneous content (e.g., a document that says that the world is flat). Some of these web sites depend on the community of users themselves to monitor document content; as the volume of content at these sites grows, it becomes problematic to attempt to maintain this information in this fashion. In addition, many corporate and governmental users are hesitant to utilize such technologies for information content and collaboration, restricting their usage to an information repository. For example, consider health care, where there are stringent HIPAA (www.hhs.gov/ocr/hipaa/) requirements regarding the security of health care data. In order to utilize a web portal or Wiki to allow clinical researchers to collaborate on studying diabetes in a patient population, there would need to be much more rigorous security requirements than the coarse-grained authorization and authentication (user names/passwords) that are typically offered by Wikis/web portals.

In this paper, we report on our research and development effort of applying role-based access control (RBAC) to web portals as part of a funded research effort [19] for collaborative software requirements elicitation [21]. In this effort, the Axon Wiki has been prototyped with RBAC security at the application level, the document level (authoring and viewing), and the look-and-feel of the portal itself. Axon is a Java-based, Ajax (developers.sun.com/ajax) Wiki that is targeting enterprise adoption by offering a wide range of document authoring, collaboration, publishing, versioning, and other capabilities. The intent is to provide a full-capability Wiki that has fine-grained RBAC that meets the requirement in terms of security requirements, flexibility and administration so that it is more suitable than open source and other commercial products.

The remainder of this paper is organized into 5 sections. Section 2 explains background concepts on the Axon Wiki including its capabilities and architecture. Using this as a basis, Section 3 details security assumptions and concepts, that while specific to Axon, can be generalized to collaborative Web portals. Section 3 also includes a detailed discussion of the permissions that support RBAC/security at the application level, the document level (authoring and viewing), and the look-and-feel of the Wiki itself. Section 4 describes the set of relational database tables that are utilized to realize the permissions given in Section 3; these tables are consulted when a user starts a session with Axon to customize the Wiki capabilities and content based on role. Section 5 reviews related research efforts. Section 6 contains the conclusions and reviews ongoing research.

2. BACKGROUND CONCEPTS

Axon is a full-function Wiki for content creation (WYSIWYG), document publishing (Web, PDF, RTF), document distribution (Email, Print, Fax), mobile access (limited via a BlackBerry), and role-based access control to allow collaboration among users that are sharing a particular task. As shown in Fig. 1, the Axon Wiki is loaded from a web server into a multi-framed structure that includes: a top bar of functions (i.e., Hide, History, Import, Export, Email, etc.) which includes full text search; three tabs for Topics, Documents, and Index, where the Topics tab is organized as accordions (e.g., US Travel, Project Brianstorm, etc.) with parent topics (Brainstorm), child topics (e.g., Project Drivers, Project Constraints, etc.), and grandchild topics (e.g., The Purpose of the Project); and, a main document window (xhtml) that is editable and stores past versions (Edit, History). The accordions and their Topic trees are customizable based on domain. Each topic (parent, child, or grandchild) can have a single xhtml document (in the main window) along with one or more other documents (PDF, Word, MPEG, etc.) that are attachable and accessible via the DOCS tab (see E in Fig. 1).

In addition, Axon has a number of other capabilities related to document creation, publishing, and viewing, as shown in Fig. 2. At the top of the figure, the WYSIWYG editor is shown that is very MS-Word like to allow the easy creation of content (the xhtml document in the main window). Documents that are created in this fashion can be viewed (and/or changed) by other authorized users; a detailed version history is maintained. At the bottom left, the documents of a topic can be assembled in various ways to publish a new combined document in different formats (PDF, RTF).

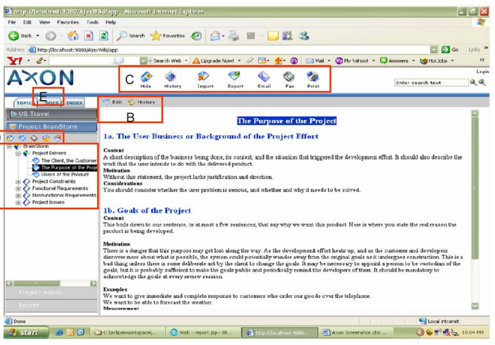


Figure 1: The Axon Wiki.

At the bottom right, Axon for a BlackBerry is illustrated, which allows a limited set of actions to both view and edit the application content.

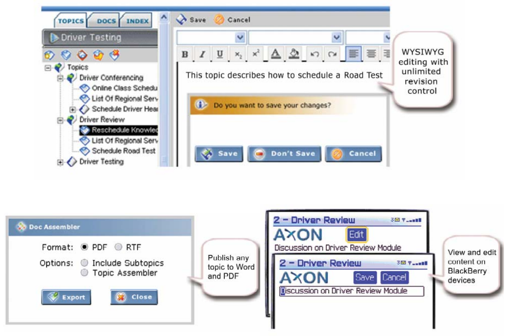


Figure 2: Axon Document Capabilities.

Lastly, in Fig. 3, the Axon architecture is shown. The involved technologies are indicated for the reader's interest. The clients can either be connected via workstations, laptops, or mobile devices. The Presentation Layer provides the typical means to access the underlying application. Brainstorm, which is the name of our software requirements elicitation effort [19], contains two grayed boxes (application specific, changeable) and two ungrayed boxes (representing core Axon functionality). The Application Layer embodies many of the various underlying technologies to support Axon core functionality. Lastly, the Data Layer allows Axon to be configured with any relational database as a backend.

3. SECURITY CONCEPTS AND PERMISSIONS IN AXON

In this section, we detail the Axon security assumptions and concepts as a means to define the relevant RBAC security permissions that Axon will support. The intent is to delineate the security granularity level in terms of the different portions of the Axon Wiki that need to be controlled. While the permissions that are presented are specific to Axon, the reader will note through the discussion that the concepts and

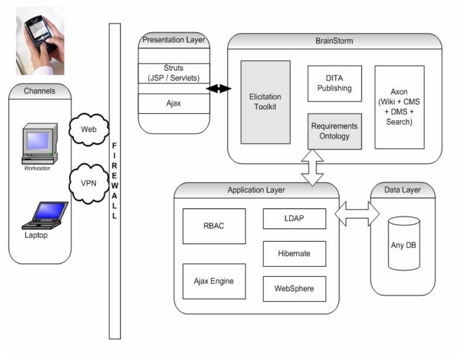


Figure 3: The Axon Architecture and its Components

permissions can be generalized to apply to a collaborative setting where there is a need to control, on a role-by-role basis, application content, access to documents, and GUI look-and-feel. To get started, in Fig. 4, we provide a table of permissions for Axon. Basic assumptions regarding roles and permissions are as follows:

- A User is identified by a UserName (unique), UserID (unique), and User duration (UserStartTime and UserEndTime that the User is active).

- A role can be defined for any capability: Guest, Author, Manager, Admin in Fig. 4 are typical roles that would be available across application domains when using Axon. For each role, there is an indication of the access to topics.

- A user can be associated with one or more roles in Axon. When a user starts a session with a tool, then the user must be authenticated (user name, password), and once authenticated, the user is given a set of authorized roles to choose from. Once the user selects one of these roles, Axon customizes itself based on the chosen role using the permissions stored in the database. Axon also provides the means to change role during a session. A user may log on to multiple different separate sessions each with its own role.

- To isolate the user from the role, we introduce a group abstraction. Each User is the member of one or more Groups, where each Group is identified by: GroupName (unique), GroupID (unique), and Group duration (GroupStartTime and GroupEndTime that the Group is active). Users can be in multiple groups and users can also have multiple roles. Each group can have zero or more users.

- An active Session for a User limits the User to a particular Group <UserID, GroupID>. The idea is that for any active Session, a User is limited to being in a particular group from a permission perspective. A User may have multiple active Sessions (simultaneously open Axon Wiki sessions with independent logons) at any given point in time.

- Permissions (as given in Fig. 4) will be assigned to Roles, Roles will be assigned to Users/Groups, and Users/Groups will be assigned to Accordions.

In addition to these basic assumptions, there are other concepts that are relevant to define the security permissions as given in Fig. 4 with respect to Axon as given in Fig. 1.

- The Axon Wiki has a Project that contains multiple Accordions (i.e., accordions) such as US Travel, Brainstorm, EGuru, and Report in Fig. 1, and for each of these individual Accordions, a Topic Tree, a Document List, and an Index is maintained. As defined, each Accordion can have one or more Users, and each Accordion can have zero or more Groups (with each Group having zero or more Users as defined previously).

- The Axon Accordions are intended to represent different types or categories of topics that are to be maintained for each User. For example, in a University setting, they could be Accordions such as Faculty, Administrator, Grad Program Director which are very type oriented and shared by multiple individuals. The Faculty Accordion (corresponding to the Faculty Role) would have a set of topics for that Faculty Role; these topics would be shared by all individuals assigned the Faculty Role. This perspective is strongly type oriented, with the application data (for instance, the advisees for each faculty member) showing up in the task Topic Tree (e.g., an Advisee topic with child topics for each student such as Student1, Student2, etc.)

- The Topic Tree contains three levels of parent, child, and grandchild topics:
  - Each topic in this tree is associated with exactly one xhtml page.
  - Each topic in this tree is associated with zero or more documents of all types (Word, PPT, PDF, GIF, etc.).
  - The DOCS tab contains a list of documents. specifically, for the selected topic - all documents for the topic and its descendants are shown.

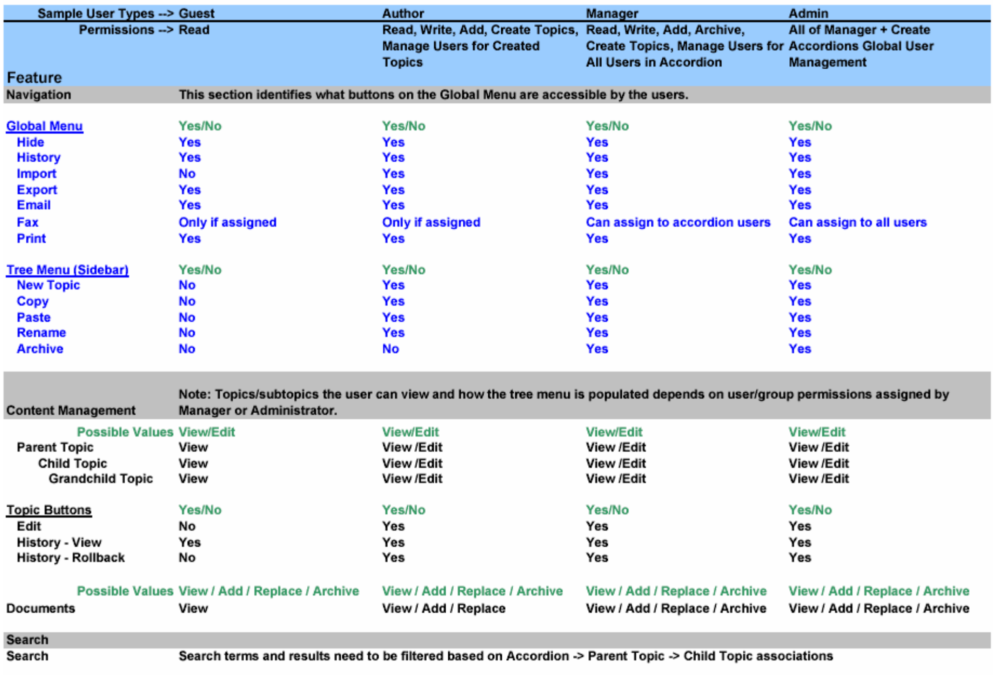
Lastly, given the discussion so far, there are many detailed permissions that can be defined to realize alternative look-and-feel and usage security of Axon on a role-by-role (user-by-user) basis. These are summarized as follows, with references to the different areas of Axon (A, B, C, D, and E) as given in Fig. 1. In terms of the topics in the Topic Tree there are three permission assignments that are maintained:

- Each Role can have one or more topics.

- Each Group can have zero or more topics.

- Each Accordion can have zero or more topics.

Thus, when a User in a Group Logs onto Axon, the Accordions that are displayed are determined by the User being authorized to the Accordion. In addition, the topics that are displayed are determined by the topics assigned to the authorized Accordions (which each have a list of zero or more topics), the Groups that the User is a member of (which each have a list of zero or more topics), and the specific



Sample User Types -->	Guest	Author	Manager	Admin
Permissions -->	Read	Read, Write, Add, Create Topics, Manage Users for Created Topics	Read, Write, Add, Archive, Create Topics, Manage Users for All Users in Accordion	All of Manager + Create Accordions Global User Management
Feature				
Navigation	This section identifies what buttons on the Global Menu are accessible by the users.			
Global Menu	Yes/No	Yes/No	Yes/No	Yes/No
Hide	Yes	Yes	Yes	Yes
History	Yes	Yes	Yes	Yes
Import	No	Yes	Yes	Yes
Export	Yes	Yes	Yes	Yes
Email	Yes	Yes	Yes	Yes
Fax	Only if assigned	Only if assigned	Can assign to accordion users	Can assign to all users
Print	Yes	Yes	Yes	Yes
Tree Menu (Sidebar)	Yes/No	Yes/No	Yes/No	Yes/No
New Topic	No	Yes	Yes	Yes
Copy	No	Yes	Yes	Yes
Paste	No	Yes	Yes	Yes
Rename	No	Yes	Yes	Yes
Archive	No	No	Yes	Yes
Content Management	Note: Topics/subtopics the user can view and how the tree menu is populated depends on user/group permissions assigned by Manager or Administrator.			
Possible Values	View/Edit	View/Edit	View/Edit	View/Edit
Parent Topic	View	View /Edit	View /Edit	View /Edit
Child Topic	View	View /Edit	View /Edit	View /Edit
Grandchild Topic	View	View /Edit	View /Edit	View /Edit
Topic Buttons	Yes/No	Yes/No	Yes/No	Yes/No
Edit	No	Yes	Yes	Yes
History - View	Yes	Yes	Yes	Yes
History - Rollback	No	Yes	Yes	Yes
Possible Values	View / Add / Replace / Archive	View / Add / Replace / Archive	View / Add / Replace / Archive	View / Add / Replace / Archive
Documents	View	View / Add / Replace	View / Add / Replace / Archive	View / Add / Replace / Archive
Search				
Search	Search terms and results need to be filtered based on Accordion -> Parent Topic -> Child Topic associations			

Figure 4: Axon Security Privileges.

Role (which each have a list of zero or more topics) that the User is playing at logon time.

The permissions in Axon, given in matrix form in Fig. 4, are explained using Fig. 1 (which has boxed certain Areas A, B, C, D, and E) by reviewing permissions for a set of sample roles (Guest, Author, Manager, and Admin in Fig. 4). These four different roles are assigned typical permissions that would be given to different Wiki users: Guest is a user with very limited permissions; Author is a user able to create and manage topics and content, and perhaps even have limited capabilities for assigning permissions to other Users; Manager is essentially the Author with additional capabilities regarding topics and content and more wide scale User management; and, Admin is essentially a system administrator with access everywhere to all aspects of a Wiki. Specifically, for Axon:

- Permissions Related to the Topic Tree (see A in Fig. 1):
  - View Means that the User (via his/her Role) has permission to View the Topic and View the xhtml page associated with that topic.
  - Edit Means that the User (via his/her Role) has permission to modify, delete, update, etc., the xhtml page associated with that topic.
- Permissions Related to Topic Buttons (see B in Fig. 1):
  - Edit having a value of Yes means the Edit button is enabled. So, if the Topic Tree has a Permission of Edit for a Topic, then the permission for the Topic Button Edit should be set to Yes.
  - History View and History Rollback are assigned on a Yes/No basis to each Role.
- Permissions Related to Global Menu (see C in Fig. 1):
  - There are permissions on the Global Menu for Hide, History, Import, Export, Email, Fax, and Print.
  - These permissions are Yes/No assigned on a role-by-role basis. The assignment of No means that the associated ICON does not appear.
- Permissions Related to Topic Menu (see D in Fig. 1):
  - There are permissions on five icons for: New Topic to Create a new Topic; Copy to Make a Copy of an Existing Topic; Paste to Paste a Copy of an Existing Topic; Rename to Change the Name of a Topic; and, Archive to Store a new Version of the xhtml page associated with the topic - Archive the topic and all the associated documents related to the topic.
  - These permissions are Yes/No assigned on a role-by-role basis. The assignment of No means that the associated ICON does not appear.
- Permissions Related to Documents (see E in Fig. 1). Specifically, a User (via a Role) can have permission to a Topic but have No permission to see any Documents associated with a Topic, namely:
  - View: Open Document with associated desktop viewer but do not save changes. However, note

that it is possible to, within the capabilities of the associated document viewer (e.g., Word, PPT, Acrobat, etc.), to perform a "Save as" operation of the document to the local file system. This "Save as" operation has no impact on the Wiki and what is stored in the Wiki.

  – Add: Be able to Import a Document. Documents have a different set of buttons. The attach or upload button is enabled if the user has add permissions for a document.
  – Replace: Be able to Substitute a new Document for an Existing Document This means to replace an existing document (say spec.doc) with an entirely different document (say impl.doc) that has nothing to do with the first document - so replace is really "Substitute this new document while saving all versions of the old one."
  – Archive: Archive means to transition a document to being "logically offline" as it exists at that point in time and remove it from the list of active documents. Users will not be able to view the archived documents. An Administrator has the authority to restore archived documents if required.

- Permissions Related to Search Tab: Search terms and results need to be filtered based on Accordion to parent topic to child topic associations

4. REALIZING RBAC AND SECURITY IN AXON

As shown in Fig. 3, the realization of security in Axon is achieved in the Application layer via a combination of LDAP and our own custom RBAC implementation. LDAP, the Lightweight Directory Access Protocol, is utilized to track directory information on users during interactive sessions. Our focus in this section is on achieving RBAC, to allow the look-and-feel of Axon to be customized according to the permissions defined in Section 3 on a role-by-role basis. Given the tight time constraints of the Phase I SBIR NSF grant (6 months), we have prototyped a basic RBAC and other security using relational database tables to capture permissions, and once captured, these same tables are consulted to customize Axon based on the chosen role of a user.

In arriving at this decision, we also considered other available technologies to support RBAC. First, XACML [20] the eXtensible Access Control Markup Language, is a web services policy constraint language that provides a standard infrastructure for security policy definition in a web context. There are many different implementations that have begun to emerge for XACML, for example, one open source implementation [24] was available, but in our timeframe (September 2006), the associated releases seemed premature and incomplete. As another example, there was the Bandit Role Engine [2], an open source RBAC solution based on the available RBAC standard from NIST [18] and XACML (Sun's implementation). However, it too had a to-be-announced Version 2 (no due date posted) that made it less attractive for our use. Thus, given our time constraint, and the fluidness of these products, we decided to not take the route of using an existing product, but to instead, develop a straightforward database model and associated implementation to realize RBAC for Axon.

The remainder of this section contains the set of relational database tables to handle the assumptions of Axon and its permissions as defined in Sections 3 (and as given in Fig. 1 and 4) in order to realize RBAC for Axon. To begin, we need to track the different Project configurations for Axon that contain the set of Accordions for each Project. Then, we can assign a particular Project configuration to a User. The ProjectInfo and AccordionInfo tables given below keep track of Projects and Accordions. ProjectAccordions maps Accordions into Projects. Start and End Times have been included for the ProjectAccordions table - that means that the Accordion is only visible for that time period. Basically, ProjectAccordions establishes the Accordions (e.g., in Fig. 1, US Travel, Project Brainstorm, etc.) that are in a particular Axon configuration for a given application.

```
ProjectInfo  <ProjectID, ProjectName>

AccordionInfo <AccordionID, AccordionName>

ProjectAccordions <ProjectID, AccordionID,
                   AccStartTime, AccEndTime>
```

Next, we need to model the Topics (parent, child, and grandchild), and associate Topics with Projects and Accordions to form a tree. Since a Project contains one or more Accordions, it makes sense to track the topics per Project/Accordions. Note that in this case, both ProjectID and AccordionID must be non-null. The idea is that regardless of permissions, for a given ProjectID, there are topics defined for each Accordion as identified by a AccordionID. The topics (and their subtopics and sub-subtopics) are all associated with a Accordion; all Topics (parents), SubTopic1 (children), and SubTopic2 (grandchildren) associated with a Accordion can be found by joining these three tables on TopicID and then selecting (or sorting) by AccordionID (or AccordionName if you join Topic and AccordionInfo tables). Thus, using Topics, SubTopic1, and SubTopic2, we are able to have a master list of all possible topics (and their subtopics and sub-subtopics) that are associated with each Accordion in a particular Project (ProjectID, AccordionID in combination).

```
Topic <TopicID, TopicName, ProjectID, AccordionID>

SubTopic1 <SubtopicID1, SubTopic1Name, TopicID>

SubTopic2 <SubTopicID2, SubTopicID1, TopicID,
                                  SubTopic2Name>
```

The TopicVersion table tracks different versions of a topic (as related to the xhtml page that is associated with each topic). The two tables, Attachment and AttachmentVersions, track the various documents (PDF, Word, etc.) associated with a Topic and their versions. They are as follows:

```
TopicVersion <VersionPK, VersionID, TopicID,
              SubTopicID1, SubTopicID2, Author,
              Description, VersionDate, Attachment>

Attachment <AttachmentPK, TopicID, LevelID>

AttachmentVersion <VersionPK, VersionID, AttachID,
                   FileName, DocType, Author, Size,
                   AttachmentDate, Comments, RandomName>
```

Given these tables, we can proceed to define a set of tables to support permissions. For Users, Groups, and Roles, three tables are defined as below; all three have start and end times to delineate the duration of the User, Group, or Role. For authorizations there are two tables: one for User to Group authorization and a second for User to Role authorization. In this case, the start and end times are the durations of these authorizations and these times are constrained by the involved tables, e.g., the User to Role authorization is constrained by the start and end times of the User and of the Role. In addition, a table of PermissionInfo is defined, for the various types of permissions in Fig. 4 (e.g., View, Edit, Archive, Replace, etc.).

```
UserInfo <UserID, LastName, FirstName, UserStartTime,
                                           UserEndTime>

PermissionInfo <PermissionID, PermissionName>

GroupInfo <GroupID, GroupName, GroupStartTime,
                                          GroupEndTime>

RoleInfo <RoleID, RoleName, RoleStartTime, RoleEndTime>

UserGroupAuthorization <UserID, GroupID, UGStartTime,
                                             UGEndTime>

UserRoleAuthorization <UserID, RoleID, URStartTime,
                                             UREndTime>
```

Using these tables, we defined two possible options to model Topic authorization. First, in Option A, the tables TopicUserAuth, TopicGroupAuth, and TopicRoleAuth are defined, to allow permissions to be established with respect to Topics, by either UserID, GroupID or RoleID, respectively, for a given ProjectID/AccordionID (which are needed to clearly differentiate between the same Topic that may be defined in multiple ProjectID/AccordionID combinations). The idea is to utilize the three tables together to establish the Topics that are actually listed (under a Project/Accordions) for each User (as limited by Role and/or Group). The Topic table defined previously contains all Topics (subtopics and sub-subtopics) for all Accordions of a Project (a superset); the TopicUserAuth, TopicGroupAuth, and TopicRoleAuth customizes this superset to a subset (which may be the entire superset) of the Topics authorized to a User belonging to a Group and also playing a Role.

```
TopicUserAuth <UserID, PermissionID, ProjectID,
               AccordionID, TopicID, SubTopicID1,
                                     SubTopicID2>

TopicGroupAuth <GroupID, PermissionID, ProjectID,
                AccordionID, TopicID, SubTopicID1,
                                     SubTopicID2>

TopicRoleAuth <RoleID, PermissionID, ProjectID,
               AccordionID, TopicID, SubTopicID1,
                                     SubTopicID2>
```

The advantage of Option A is that it keeps the permissions logically and physically separate. For the future, if we decide to make changes to, for example, User permissions, such a change would not impact Group permissions.

Next, in Option B, we defined a generic TopicAuth table as:

```
TopicAuth <ProjectID, AccordionID, GroupID,
           UserID, RoleID, PermissionID,
           TopicID, SubTopicID1, SubTopicID2>
```

This table would be used as follows: for User authorizations to Topics, ProjectID, and AccordionID would be defined with GroupID and RoleID null; for Group authorizations to Topics, GroupID would be defined with UserID and RoleID null; and, for Role authorizations to Topics, RoleID would be defined with UserID and GroupID null. The advantage to Option B is that we have all permissions in a central location; the disadvantage is that we must keep track of all IDs (nulls) for all permissions and as a result, changes that we make impact all three types of permissions. In the short term for our Phase I effort, we selected Option B since it allows us to expand the authorizations to Accordions and Groups without introducing any new tables. For the longer term (Phase II and commercialization), Option A is preferred, or in fact, we may redo all of this security to use XACML or another technology that may have matured.

Lastly, for Wiki look-and-feel security, a set of tables are defined to identify the widgets (buttons, etc.) of the Wiki to be controlled, the privileges of those widgets, and then to define the Wiki look-and-feel security on a role-by-role basis. These are defined in the following three tables:

```
WikiLookandFeelAuthorization <RoleID, widgetID,
                                   widgetprivilegeID>

Widget <widgetID, widgettype, widgetcategory, widgetname>

WidgetPrivilegeType <widgetprivilegeID,
                                 widgetprivilegename>
```

Widget is being used to refer to a button, icon, link, or any other aspect of the Wiki GUI that needs to be controlled. To illustrate these tables, let's consider some actual tuples. First, there are all of the Widgets that are present in the Axon Wiki, which are uniquely identified (W1 to W6) and are all buttons (Table 1).

Table 1: Widget

WidgetID	WidgetType	WidgetCategory	WidgetName
W1	Button	TopicLinks	HistoryView
W2	Button	TopicLinks	Edit
W3	Button	TopicLinks	HistoryRollback
W4	Button	GlobalMenu	History
W5	Button	TreeMenu	NewTopic
W6	Button	TreeMenu	Copy

Next, there are the privileges (permissions) for each widget - Table 2 shows buttons that can either be Yes or No (for buttons) or ActiveIcon or NoIcon (for icons like Email in Fig. 1).

Table 2: WidgetPrivilegeType

WidgetPrivilegeID	WidgetPrivilegeName
P1	Yes
P2	No
P3	ActiveIcon
P4	NoIcon

Lastly, on a role-by-role basis, for each role, we identify the widget and the allowable permission, as given by the tuples in Table 3 for role R1. To summarize, Widget is the list of all of the look-and-feel components of the Wiki that need to be controlled, WidgetPrivilegeType tracks the status of each widget, while the WikiLookandFeelAuthorization tracks the actual authorization by role to each <widget, type> pair. Once the user has been established with a role for the Wiki session, the customization of the Wiki screen is controlled by lookups and joins into these very basic tables.

Table 3: WikiLookandFeelAuthorization

RoleID	WidgetID	WidgetPrivilegeID
R1	W1	P1
R1	W2	P2
R1	W3	P1
R1	W4	P1
R1	W5	P3

5. RELATED WORK

Our work in this paper has been influenced by many different areas of research. To start, the role-based access control that we have designed for Axon has been influenced by our own past work [10], as well as foundational work in RBAC [23] and standards [12]. However, we have, for this initial version of Axon, kept the RBAC rather limited in its scope. In terms of security for collaborative computing, it is interesting to note that non-web-based computer supported cooperative work, proposed in the 1980s and explored into the early 1990s, addressed many issues on individual and group behaviors that are still relevant today[16]. This included: work in dynamic collaboration for a work group over space and time constraints [17]; multimedia communication systems that support widely distributed work groups [15]; and, our own work [9] on RBAC for collaborative computing environments. Much of this work has been supplanted by web-based research such as: an effort that seeks to construct a security architecture that is capable of being tailored for customer needs with a services-based approach for authentication, authorization, and RBAC [8]; a survey effort that seeks to identify the myriad of security issues that must be considered when multiple individuals collaborate and share information [1]; and, controlling access to information in a repository where there is a dynamic and unknown user population [22] using a trust-based approach. We are considering these and other efforts to fully understand the security issues that impact collaboration, in Axon, in our current work (software requirements elicitation) as well as in other critical domains such as health informatics.

From a more practical perspective, there have been a number of efforts that employ service-oriented architectures (SOA) or web-services as a basis for achieving security. For instance, [5] proposes X-RBAC, an approach that leverages XML as a means to first model RBAC, and then enforce defined security for an application by interacting with a security processor via SOAP (or other XML messaging). In a related effort [6], X-GTRBAC provides a larger-scale infrastructure based on X-RBAC for handling security for enterprise (business) applications. The underlying security infrastructure in both of these efforts must take advantage of SOA and Web services in order to offer guarantees in terms of security assurance, data integrity, confidentiality, etc. [4]. We agree with the idea that SOA and Web Services will be critical to achieve security solutions that easily operate in a web-and-collaborative setting such as Axon.

Lastly, as realized, our security for Axon is achieved in three dimensions: the application by controlling the accordions and topics that are available on a role-by-role basis; the documents (xhtml and other attachments) by controlling what is viewable/editable; and, the Wiki look-and-feel that enables select buttons, options, and commands. A fourth dimension that is of interest involves the xhtml document itself that is in the main window (see Fig. 1 again). Presently, this document is controllable by being viewable (or not) and editable (or not). However, in true collaboration, this document itself, being based on XML, may be partitioned into components, with RBAC utilized to control who can see/edit each of the components. In that regard, we have explored related efforts that involve security for the semantic web (in general) and controlling access to XML documents (in particular). For the semantic web, there has been an effort that essentially established a road-map for security for the semantic web by identifying the key security issues [25], as well as an effort that has focused the discussion of security for the semantic web from the perspective of web databases and services that are appropriate [13]. In terms of potential solutions to control access to information at the document level, there have been two efforts of note: [3] has proposed a model for access control of XML documents with the policy including credentials and privileges; and, [11] has extended the concept of security views so that they are applicable to XML DTDs in order to screen information for XML instances before they are displayed. Both of these efforts of our interest as we proceed to fine-tune our security in Axon to control access to the components of the xhtml documents associated with topics on a role-by-role basis.

6. CONCLUSIONS AND ONGOING WORK

In this paper, we have presented our work on role-based access control (RBAC) for collaborative web portals, as attained within the Axon Wiki. The intent was to expand the security capabilities of portals/Wikis from beyond simple user and password authorization to a more robust set of privileges that encompasses and realizes RBAC at the application level, the document level (authoring and viewing), and the look-and-feel of the portal itself. To accomplish this, in Section 2, we presented the Axon Wiki capabilities and architecture. With this, in Section 3, we detailed assumptions of Axon in order to provide a detailed treatment of permissions (see Fig. 4 again) and the attainment of these permissions in the various Axon components. To meet strict timeframe constraints of the funded effort [19], we prototyped the RBAC as a set of relational database tables, as given in Section 4; these tables are consulted when a user playing a role initiates a session in order to customize Axon based on the privileges that are authorized to that user. Lastly, we placed our work into context by considering related efforts in Section 5. The work as presented herein, while specific to Axon, can be generalized to other collaborative web/portal settings.

In terms of ongoing efforts, in addition to this RBAC, we have also recently completed a second SBIR Phase I grant from the Department of the Navy to explore the ability of a Wiki like Axon to support mandatory access control/multi-level security. This also included exploring issues related to security assurance in order to attain Evaluation Assurance Level-6 (EAL6) [7]. In addition to this effort, we have an ongoing effort with the University of Connecticut Health Center on the feasibility and utility of the Axon Wiki in supporting various aspects of clinical research and healthcare informatics. As separation of duty and constraint delegation of permission are essential in an emergency case, these features will be addressed in our future work as well. As Axon supports the idea of sessions, it is possible to create an role authority component, which tracks the current roles logged in. Thus, separation of duty can be implemented. (Constrained) Delegation of permission is more complex to implement, as a role must have administrative right to change the permissions of another role. Nevertheless, we'll address this as part of our future work along with the enabling Axon to work in an distributed environment.

7. REFERENCES

[1] M. Attalah. Security Issues in Collaborative Computing. In *Computing and Combinatorics COCOON 2006.* D. Chen and D. Lee (eds.), Springer LNCS, Vol. 4112, 2006.

[2] BANDIT, www.bandit-project.org/index.php/Role_Engine

[3] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM Trans. Inf. Syst. Secur.*, 5(3), 2002.

[4] E. Bertino and L. Martino. Security in SOA and Web Services. *Proc. of 2006 IEEE Intl. Conf. on Services Computing (SCC 2006).*

[5] R. Bhatti, et al. Access Control in Dynamic XML-based Web Services with X-RBAC. *Proc. of Intl. Conf. on Web Services, ICWS '03*, 2003.

[6] R. Bhatti, et al. X-GTRBAC: An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control. *ACM Trans. Inf. Syst. Secur.*, 8(2), 2005.

[7] Common Criteria, http://www.commoncriteriaportal.org/public/files/ccusersguide.pdf

[8] Y. Demchenko, et al. Security Architecture for Open Collaborative Environment. In *Advances in Grid Computing - EGC2005.* P. Sloot, et al. (eds.), Springer LNCS, Vol. 3470, 2005.

[9] S. Demurjian, et al. User-Role Based Security for Collaborative Computing Environments. *Journal of Multi-Media Review*, 4(2), Summer 1993.

[10] S. Demurjian, et al. A user role-based security model for a distributed environment. In J. Therrien, ed., *Research Advances in Database and Information Systems Security.* Kluwer, 2001.

[11] W. Fan, et al. Secure XML Querying with Security Views. *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 2004.

[12] D. Ferraiolo, et al. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3), 2001.

[13] E. Ferrari and B. Thuraisingham. Security and Privacy for Web Databases and Services. In *Advances in Database Technology - EDBT 2004.* E. Bertino, et al. (eds.), Springer LNCS, Vol. 2992, 2004.

[14] S. Foley and J. Jacob. Specifying Security for Computer Supported Collaborative Working. *Journal of Computer Security*, 3(4), 1995.

[15] E. Francik, et al. Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems. In [26].

[16] J. Grudin. CSCW Introduction. In [26].

[17] H. Ishii and N. Miyake. Toward an Open Shared Workspace: Computer and Video Fusion Approach to TeamWorkStation. In [26].

[18] National Institute of Standards and Technology, http://www.csrc.nist.gov/rbac/

[19] NSF Award #0611053 to Serebrum, Inc., SBIR Phase I: BrainStorm - Collaborative Customer Requirements Elicitation for Distributed Software Development.

[20] Organisation for the Advancement of Structured Information Standards, www.oasis-open.org/committees/xacml/

[21] P. Pia, et al. BrainStorm: Collaborative Customer Requirements Elicitation for Distributed Software Development. *Proc. of 31st Annual Software Engineering Workshop*, March 2007.

[22] I. Ray and S. Chakraborty. A Framework for Flexible Access Control in Digital Library Systems. In *Data and Applications Security XX*, E. Damiani and P. Liu (eds.), Springer LNCS, Vol. 4127, 2006.

[23] R. Sandhu, et al. Role-based access control models. *IEEE Computer*, 29(2), 1996.

[24] Sun's XACML Implementation, http://sunxacml.sourceforge.net/

[25] B. Thuraisingham. Security Issues for the Semantic Web. *Proc. of the 27th Intl. COMPSAC 2003.*

[26] Special Issue on Collaborative Computing. *Comm. of the ACM*, 34 (12), 1991.