Role Based Access Control for UDDI Inquiries

Raymond Peterkin  
School of Information Technology and Engineering  
University of Ottawa  
Ottawa, Canada K1N 6N5  
Email: peterkin@site.uottawa.ca

Jinsuk Solomon  
School of Information Technology and Engineering  
University of Ottawa  
Ottawa, Canada K1N 6N5  
Email: jsolomon@ncct.uottawa.ca

Dan Ionescu  
School of Information Technology and Engineering  
University of Ottawa  
Ottawa, Canada K1N 6N5  
Email: ionescu@site.uottawa.ca

Abstract—Web Services are commonly used by organizations for B2B integration and enterprise application integration. UDDI registries are widely used as mechanisms for businesses to list and discover available services. Businesses may want to limit the information of their services to certain users and businesses. However, UDDI provides no mechanisms to limit data inquiries to a particular subset of requesters. This paper presents a role-based access control implementation of UDDI so businesses may safely publish data and limit the organizations who may perform inquiries against their information.

I. INTRODUCTION

With service-oriented architecture (SOA) an application’s business logic or individual functions are modularized and presented as services for consumer/client applications. SOA allows enterprises to plug in new services or upgrade existing services in a granular fashion, providing the option to make the services consumable across different channels, and exposes the existing enterprise and legacy applications as services, thereby safeguarding existing IT infrastructure investments.

Web Services can be used to implement SOA by allowing applications to communicate through a standard set of XML based protocols. Since communication protocols are standardized, Web Services can be used for several applications and purposes including business-to-business (B2B) integration, enterprise application integration (EAI) and connecting applications across numerous and heterogenous platforms across a network. Once a web service has been established, mechanisms must exist for users to discover and use the service. Those mechanisms must be standardized, otherwise the advantage to using Web services is significantly diminished.

UDDI (Universal Description, Discovery and Integration) [9] provides a standardized method for publishing and discovering information about Web Services. Using UDDI, businesses can publish information about themselves and search for information and services for other businesses. UDDI’s standardized integration and interoperability enables businesses to associate their Web services with those of various partners.

UDDI provides different security mechanisms to protect its data. Users are authenticated every time information is published to ensure that information is only saved by recognized and authorized businesses. Digital signatures may be used to ensure the data integrity and authenticity of published data. Users performing inquiries could filter information based on whether the information is digitally signed. Publishers would have assurances that they are not misrepresented by someone claiming to own a data entity. Policy support also exists for UDDI. Registries may be composed of multiple nodes and policies are statements of required or expected behavior. Therefore, policies for authorization models may be developed to account for the components of the registry. UDDI supports a security API with functions limited to acquiring and discarding authentication tokens.

There are no mechanisms in UDDI to authenticate users when performing an inquiry. All information published in UDDI may be retrieved by any user with knowledge of the registry’s contact information. Some businesses only want certain parties to have access to certain services because the services are of a sensitive nature and their resources must be allocated with great discretion. For example, a business may develop a web service for allocating bandwidth on a communications channel to make the process more efficient. The business may only wish for the bandwidth to be allocated to a specific set of users. If the Web Service is published in a UDDI registry, anyone with knowledge of the registry may use the Web service to retrieve bandwidth. This example illustrates a clear need for security mechanism when users perform inquiries to protect services and resources from unauthenticated users and unauthorized access.

Adams and Boeyen [2] outlines a framework for implementing security for Web Services. Security is discussed in terms of registry security (ensuring users receive trustworthy information), transaction security (ensuring that transactions are successfully executed) and infrastructure linkage (ensuring that all security infrastructure mechanisms are understood). Among the registry security requirements, it is stated that some UDDI information may be confidential as some content should only be available to a specific subset of service requesters.

Dai and Steele [6] proposes a role-based access control system in corporate UDDI registries based on the framework proposed in [2]. External methods to prevent unauthorized access, like XML SOAP [7] security gateways, are considered inadequate for the purpose of preventing unauthorized users from reading certain information. For full effectiveness, easier maintenance and configuration, users, security roles and security policies are defined in XACML [10] to build access control in UDDI. However, the XACML based system
provides a generic method for role and policy implementation and there are no implementation specific details for enforcing authenticated inquiries in UDDI.

There are also industry initiatives to provide more security to UDDI. The most comprehensive security alternative may be Microsoft UDDI which provides four different roles to users such as Users, Publishers, Coordinators, and Administrators. It defines the level of interaction in the registry by differential granting of authority. After the registry authenticates users by the native UDDI authentication, the users interact with information and data depending on their roles. However, access to the business information that companies may want to hide is not regulated.

This paper presents a design and implementation of a role-based access control system for UDDI inquiries. Section II provides an overview of Web Services, UDDI and how they interact. Section III summarizes the principles of role-based access control while Section IV describes the implementation in significant detail. Section V summarizes the results and Section VI concludes the publication.

II. Web Services and UDDI

Web Services interact with UDDI in the manner shown in Fig. 1. A Web Service acting as a provider will publish its information in the registry so other Web Services may retrieve information to invoke the service. Consumer Web Services perform inquiries against the registry to retrieve the necessary information and then communicate with provider Web Services to invoke the desired functionality. Web Services interact with UDDI through inquiry and publishing APIs that humans normally use indirectly with a GUI to perform similar tasks. Web Services and humans may also make administrative changes to UDDI depending on the privileges they possess.

Users interact with UDDI through SOAP based APIs. These APIs are divided into those that register information and those that search the registry for information. Registration APIs are used by publishers to save, update and delete information. Anyone using the publishers APIs must request an authentication token from the operator.

Search APIs are used by consumers to retrieve information from the registry. Consumers must provide search parameters but do not require an authentication token. Therefore, anyone with basic knowledge of the UDDI registry can execute the search APIs at their discretion.

UDDI data is stored in various data structure types with explicit relationships to associate information and avoid repetition. Fig. 4 illustrates the high level components of the UDDI data model and their relationships.

A business entity is considered the top level structure. It describes the business (or other entity) for which information is stored in the registry. All other data structures have a relationship to its corresponding business entity. The name and description of the service being published is contained in the entity 'Business Service' while the services access information (i.e. entry point address) is stored in the binding template. The tModel contains information uniquely identifying the service’s specification like a WSDL [4] file describing the interface. Relationships between business entities can be established with the 'Publisher Assertion' data structure. Typical relationships include indicating subsidiary or department links between business entities.

Each data structure contains several data fields but they are not shown in Fig. 4 to simplify the diagram. Data types include universally unique identifiers, strings and structures. Structures are compound data types containing multiple values and are used to represent a single data structure and many nested components.

Mechanisms to enhance the security of UDDI must specifically account for the data structure and their relationships. For
example, if inquiry access is prohibited to a service, then it should not be indirectly possible through any associated data structure. Publisher assertions also present a challenge because the association between two businesses may indirectly provide information to users that they are not authorized to receive.

III. ROLE-BASED ACCESS CONTROL

Access decisions are frequently determined by roles taken by individual users within an organization. Depending on the organization and its operations several roles may be defined with varying degrees of access to perform different tasks. A role-based access control (RBAC) policy bases control decisions on the functions allocated to a user through the roles allocated to the user. Fig. 3 illustrates the relationships between the various components of an RBAC policy.

A typical RBAC system consists of users, roles, operations, objects and permissions. Objects are the entities to which access are being restricted. An arbitrary set of operations may be performed on any set of operations. A collection of operations are represented in a single role and users are assigned to roles so they may perform the set of operations at their discretion. Permissions allow operations to be performed by some users while prohibiting others from doing the same thing. Users may be assigned to multiple roles and roles may access overlapping sets of objects. Although now shown in Fig. 3 relationships may exist between roles such that a set of operations may be shared or inherited between them.

Applying the concepts of RBAC to inquiry authorization for UDDI we see that the data structures shown in Fig. 4 can represent the objects of an RBAC system. Performing inquiries is the only operation of interest as publishing data already requires authentication. Therefore, an RBAC system can be implemented by defining roles whose sole operation is to allow inquiry access to various UDDI data structures. A given role would allow access to a particular set of data structures and any user not assigned to the role would not be able to perform an inquiry against information in those data structures. Assigning a user to the role and restricting permission would be tantamount to not assigning the user at all so long as the party managing the roles has full and exclusive discretion to assign access. Therefore, no specific mechanisms are required for setting permissions. Inquiry access to publisher assertions can be prohibited to ensure that no unauthorized access occurs.

IV. IMPLEMENTATION

This section describes the high level design and implementation of the registry applying RBAC for additional security. The enhanced registry will be described as secure UDDI (sUDDI) to reflect the additional security and authentication features used to perform inquiries. A high level description of sUDDI is shown in Fig. 4.

sUDDI is based on jUDDI, an open source implementation of UDDI in Java. The latest version of UDDI is version 3. However, the latest release of jUDDI implements UDDI version 2. Therefore, features added to UDDI in version 3 (publisher assigned keys, digital signatures, policies, subscription APIs, etc.) are not available in sUDDI.

The standard UDDI database, publish API and SOAP processor remain unchanged as much as possible. The original inquiry API must be modified to authenticate the user attempting to search the registry. Two new components must be added to the standard UDDI registry. The UDDI access data component is the persistent data storage mechanism for role-based data and the admin API are a series of functions allowing a user to manipulate data for access control and user accounts. The following subsections describe the additional or modified components in further detail.

A. UDDI Access Data

Fig. 5 illustrates the database schema used to store RBAC related information in sUDDI. A role is uniquely identified by a role identifier and its creator (shown with a publisher identifier). In that manner, multiple publishers can create roles with the same name without conflict. Users are uniquely identified in a similar manner with user identifiers and publisher identifiers.

An arbitrary set of users, businesses, services, binding templates and tModels are associated with a role by storing the unique identifiers of the role and the related item in a separate table. UDDI data are uniquely referenced through
keys allowing access to data in the UDDI database shown in Fig. 4 whenever necessary.

B. Inquiry APIs

To enable inquiry authentication in sUDDI, the same authentication mechanisms used to execute the publishing APIs were applied to the inquiry APIs. Users must first request an authentication token before performing any publishing function. The authentication token must accompany any inquiry API function so a determination can be made in sUDDI if the user has sufficient privileges to execute the desired function. Once authentication is successful, the inquiry API function is executed as it would in a fully compliant UDDI registry.

Existing inquiry APIs were modified and no additional functions were added. The following functions in the UDDI inquiry APIs were modified to require an authentication token when executed:

find_binding: Finds and returns specific binding information.
find_business: Finds and returns information about one or more businesses. Acts as the main API for the initial search.
find_relatedBusinesses: Finds and returns businesses related to given business key.
find_service: Finds and returns specific services for a given business.
find_tModel: Finds and returns tModel information.
get_bindingDetail: Finds and returns full detail on binding information.
get_businessDetail: Finds and returns full details on specific business.
get_businessDetailExt: Performs same functionality as get_businessDetail but with extended information defined after UDDI v1.
get_serviceDetail: Finds and returns full details on specific service.

get_tModelDetail: Finds and returns full information on tModel.

C. Administration APIs

The UDDI specification does not define a standard set of APIs to perform administrative tasks. Therefore, the necessary APIs must be designed, implemented and integrated into sUDDI and its clients. Development of the APIs involves numerous tasks including defining and implementing several functions, data types and SOAP messages. Administration APIs for sUDDI are currently in development and fall into one of three categories.

The first category is related to publisher information. APIs must exist to add and remove publishers. Mechanisms must also exist to change publisher information including passwords and privileges. Only publishers with administrative privileges would be permitted to execute these tasks and sUDDI must be created with at least one administrative publisher.

Remaining categories of administration APIs are for modifying RBAC information in sUDDI. One such category is role management and it includes APIs for creating, removing and modifying roles and users. The second category deals with role access. Those APIs provide the facilities to add and remove the businesses, services, binding templates and tModels that can be searched through a given role. Users are also assigned and removed from roles through these APIs.

Any publisher may use the RBAC APIs to create and modify role related information. Publishers may only modify roles or users they created and allow inquiry access to businesses, services, binding templates or tModels that they have created. However, publishers are permitted to add any user defined in sUDDI to any role they have created.

The following functions have been implemented for the administration APIs:
create_publisher: Used to create a previously undefined publisher in sUDDI to save information related to businesses and services.
create_role: Used to create a previously undefined role in sUDDI for publishers to define who can perform inquiries against their web service data.
create_user: Used to create a previously undefined user in sUDDI to perform inquiries.
delete_publisher: Used to delete an existing publisher from sUDDI. Only publishers with administrative privileges may perform this action.
delete_role: Used to delete an existing role from sUDDI. All relationships associated with the role are automatically delete but not the underlying data itself.
delete_user: Used to delete an existing user from sUDDI and all relationships between the user and any roles that may exist.
modify_publisher: Used to change information associated with the publisher including the publisher name and its privileges.
modify_role: Used to change the description and relationships associated with a given role.
modify_user: Used to change the name associated with the user.

D. SOAP Processor

Messages sent between sUDDI and the client library must be defined by both entities and transported using the same protocol. While the vast majority of messages are already defined through the UDDI specification, data for RBAC must be separately defined and implemented into the SOAP processor and UDDI client. Two messages that must be defined are the role and the user.

Fig. 6 define the structure used to transport information about a role with sUDDI. The role has attributes to define the role identifier, publisher and description. There are five elements inside the role to hold references to multiple users, business entities, services, binding templates and tModels. Using a single data structure to represent all data associated with a role makes it possible to use this one form of data representation for all functions associated with roles.

The elements for business entities, services, binding templates and tModels are defined and used directly in the role data element. However data for the users must be defined and implemented separately. An XML representation of the user information transported between sUDDI and the client is shown in Fig. 7. The element contains three attributes for the user identifier, publisher and user name.

E. UDDI Client

Any application that can successfully send and receive SOAP messages from a UDDI registry is a client. Several UDDI clients exist as browsers and libraries that are fully compliant with the UDDI specification. However, sUDDI will not conform to standard inquiries due to the necessary authentication. Furthermore, support for the administration APIs must also be available to the client. Therefore, a library must be developed so users may successfully execute inquiries with authentication and administrative tasks.

UDDI4J [5] is an open source Java class library that provides an API to interact with a standard UDDI registry. Modifications have been made to UDDI4J for additional authentication and new APIs. Remaining modifications to UDDI4J will be made as the administration APIs are completed including defining new functions and SOAP messages.

V. RESULTS

sUDDI is currently in development but modifications to the UDDI client and registry have been successfully performed. Thus far, sUDDI has been implemented with limited RBAC mechanisms where publishers are the only users who may perform inquiries. Referring to Fig. 3, the only existing role is that of a publisher where users may be added or removed.

The GUI shown in Fig. 1 has been developed for human interaction with sUDDI and provides a front end to the modified UDDI4J library. Any web service using the same library can interact with sUDDI and invoke the same functionality. Therefore, successful use of the GUI is verification that web services can use sUDDI for B2B integration and other tasks with role-based access control for additional security.

Fig. 8 shows an image of the GUI after validating a user’s information. Authentication is required when data is searched or published. Therefore, validation of a user’s information is required before every task. A user ID and password must be verified against information in the sUDDI database. If the information is valid, an authentication token is granted. The token appears only for the purpose of testing at this stage in development. If the user information is invalid, an exception is thrown preventing access to the sUDDI registry.

Fig. 9 illustrates a screen where a user may search information in sUDDI after having their credentials successfully verified, as seen by observing the same authentication token in Fig. 8 and Fig. 9.
Users may list information based on the data structures in Fig. 2 and later refine their results. Fig. 10 demonstrates the results of a search where all the businesses for a publisher are listed by name. Other businesses in the registry are hidden and inaccessible to the user.

VI. CONCLUSION

This paper has presented a role-based access control system for UDDI. Publishers can limit the users searching their information through role-based access control. The registry, client library and GUI have been implemented for RBAC. Therefore, web services may use the same functionality to perform their tasks with increased security.

In addition to completing aforementioned modifications to sUDDI, future work includes full backward compatibility with standard UDDI registries where publishers can make information publicly available to anyone without any authentication required.

REFERENCES