A Role-Based Authorization Model for Service-Oriented Architecture

Wen-Shin Hsu, Jiann-I Pan

1 Institute of Medical Science, Tzu Chi University, Hualien, Taiwan, ROC
95351114@stmail.tcu.edu.tw
2 Department of Medical Informatics, Tzu Chi University, Hualien, Taiwan, ROC
jipan@mail.tcu.edu.tw

Abstract

Service-oriented architecture (SOA) is widely recognized as an especially effective solution for integrating loosely coupled and distributed resources. One of the major challenges in developing SOA-based applications is the management of authorization requirements in distributed environments. This paper proposes a formal authorization model based on a role-based access control model to demonstrate the approach for authorizing service requesters, to access a particular service and information under specific permissions in an SOA. The proposed model defines the authorization relationships and constraints among users (service requesters), roles, permissions, and services according to first-order logic and set theory. Furthermore, role hierarchy and service hierarchy are discussed to fulfill the requirements for authorization hierarchy. Finally, we use a health care system developed in a service-based architecture to demonstrate the utility of the concepts of the proposed authorization model.

Keywords: Service-Oriented Architecture, Web Service, Role-Based Access Control

1. Introduction

Based on twenty-first-century Internet popularization, service-oriented architecture (SOA) has attracted attention as an optimal concept for integrating loosely coupled and distributed resources. Service-oriented architecture is modeled as a network of components connected through a set of services anchored in a particular source and component destinations. Service-oriented architecture has been widely reported to offer a flexible mechanism for weaving large-scale enterprise applications [1,2,3]. Hence, large-scale distributed enterprise applications can be composed using numerous standard application services [2,4].

In the past, using services was focused on easy, rapid broadband dissemination of varying information. Through the widespread use of the Internet, more companies and organizations have created Intranets in private networks to organize information using SOA [4,5,6]. Based on the infrastructure of the Internet, Intranets offer an effective approach for information access for hospital patients and also guard against the public Internet using firewalls. Moreover, hospital information systems (HIS) provide electronically accessible information. Patients and other users often access multiple sources with different characteristics in services, where each source has its own methods of access with different functionalities. The SOA provides an effective approach to organize and present services with different characteristics to consumers and providers in the HIS [7].

Although SOAs have attracted considerable attention from Intranet providers and brokers, the problem of authorization has not been widely investigated. Among Intranet providers and brokers, not all users are allowed to access all information, and users with various privileges have different restrictions for accessing the services. Although existing SOAs can provide fairly fine grained access control, they require some objects to administer centrally a particular provider, consumer. However, as the number of services to be shared and distributed grows, the need for effective administration to restrict access to specific consumers surely increases, and the administrative problems of authorization may limit the widespread usage of SOA.

Because the SOA service access level control mechanism does not clearly define the Web service [8,9,10], we do not know whether the services transmitted meet the requirements of its users. Following a general access control policy, this system adds the “role” component to the architecture of users and permission. A user accesses his permission through the role’s intermediary components, which reduces the number of changes in user access permission that must be made, thereby easing the
burden of the system administrator. By allocating responsibilities to roles, the database management of
the personnel and their responsibilities is streamlined. Role-based Access Control (RBAC) grants a
particular role an appropriate range of permission to access the information on a legitimate role.
Through RBAC, the transfer between the individual and the role is simplified, and the role and
permission allocation can readily be performed. In addition, through SOA, the exchange of service
information and the system maintenance are facilitated, and the security of the database and the system
are enhanced. The “role” is a concept that differs from the “group.” A role carries the concept of not
only users but specific access permission, role inheritance, and rights and responsibilities allocation, as
well as other interactive conditions.

By combining SOA with RBAC [11,12,13] to form role-based service-oriented architecture (RSOA),
this study defines the roles of services, assumptions, and scopes of authority. By adding a device
between the user and the scope of authority, the user can assess his/her scope of authority through this
intermediary device, and can thus simplify the process and greatly reduce the administrative burden.
Using RSOA can clearly define the roles of SOA in a distributed system environment.

The remainder of the paper is organized as follows. Section 2 discusses the related work and
presents drawbacks. This is followed by a short review of role-based access control in Section 3, which
introduces the concepts and advantages of role-based access control. In Section 4, we introduce the
structure and the elements of the proposed role-based authorization model. Moreover, we present
several new notations for authorizations and considerations of different types of services. In Section 5,
we propose the architecture of our authorization system based on the proposed authorization model.
Section 5 also introduces the concept of how to authenticate the identities of consumers and determine
the associated authorizations using credentials. Section 6 draws conclusions and suggests future work
for role-based SOA.

2. Related work

Two of the most challenging problems in managing SOA are the complexity of authorized
administration and modeling of services. The following is a brief review of previous related work.
In [14], the authors attached the new SOAP message, which is called the SOAP account. The
authors recorded the structure of the SOAP message elements, such as the header elements and signed
elements. The main idea of the SOAP account is to protect the messages from rewrite attacks. The
SOAP accounts are exposed on their own, the authors suggested that extra method checks are executed
on both the account and on the message to secure the absence of attacks. These models have several
common limitations. The first is that the service requestor and service provider usually come from
different domains; hence, they are strangers to each other and cannot create a trust relationship.
Another limitation is that service providers represent different business entities with complete control
over services. Finally, service providers and service requestors should confront numerous strange users
and cannot know the amount of authorization administration. Roles can combine with internal working
positions, including many advantages to finish tasks required for the corresponding positions.

The work in [15] not only discusses authentication but also inspects access control and identity
management at the service level. The authors proposed the architectural reference framework for
services, which are based on the approach. Therefore, the file is produced, which contains the
composed data that can be reused to authenticate the topic in the SOA system.

Because the SOA service access level control mechanism does not clearly define the Web service
[8], we do not know whether the transmitted services meet the requirements of its users. The RBAC
can facilitate the transition of roles played by an individual and the distribution of authority to each role.
The RBAC can be used to improve SOA for its exchange of services and maintenance management,
thus enhancing the security of the database and system. By combining SOA with RBAC to form RSOA,
this study defines the roles of services, assumptions, and scopes of authority. By adding a device in
between the user and its scope of authority, the user can assess the scope of authority through this
intermediary device, and can thus simplify the process and greatly reduce the administrative burden.

When restrictions are imposed on the roles for the service access levels, we must consider placing
restrictions on the persons who access the data. For example, a physician has authority to modify a
medical record, but the status of the patient is special. Thus, we must combine the binding service with
RBAC to verify the role of a person who accesses the data.
3. An overview of role-based access control

This section informally reviews the basic concepts of role-based access control that are used in this paper. We refer the readers to [11,16] for additional details related to role-based authorization models.

With RBAC [11,16], access permissions are based on the roles that individual users assume as part of the organization. Users accept assigned roles. The process of defining roles should be based on the thorough analysis of how the organization operates. Access rights are grouped according to role name, and the use of resources is restricted to individuals authorized to assume the associated roles. In many civilian governments and companies, the business or agency is the owner of information, and discretion on the part of the users may not be appropriate. Thus, RBAC is often described as a form of non-discretionary access control. However, RBAC is not focused on solving multi-level security problems because it is assumed within the existing standard for mandatory access control [17,18,19]. Moreover, many authorization models commonly provide groups of users as the access control unit. The major difference between user groups and roles is that groups are typically treated as the collection of users but not as the collection of permissions. The role is both the collection of users and collection of permissions [16].

In particular businesses, roles usually have overlapping responsibilities and privileges; that is, users belonging to different roles may need to perform common operations. In this situation, using the property of role hierarchies in RBAC would be efficient without specifying repeatedly the common operations for each role that is created. Role hierarchies can be established to provide for the natural structure of the enterprise or company.

Moreover, the features and concept of RBAC are policy neutral, supporting three security principles: least privilege, separation of tasks, and abstract permissions. The least privilege states that only the authorization required for the duties performed by the user in each role are assigned to each role. Separation of tasks states that when the user is authorized as the member of one role, then the user is not authorized as the member of another role [11,17]. That is, no single user can be allowed to perform all of the operations within a particular business function. Traditional operations may be read, write, and executed, but the abstract permissions provided by RBAC can be used to capture complex security-relevant details or constraints that cannot be determined using the simple mode of access. For example, the company may define an accounting supervisor role that is allowed to perform correction operations.

4. The role-based authorization model

This section formally redefines and extends some characteristics of the RSOA model. Five entities are shown: user (U), roles (R), session (S), expiration (E), and permissions (P). A user can belong to many roles, and a role can have many users. User assignment (UA) and permission assignment (PA) relationships; both are many-to-many and both are key to RSOA.

4.1. Basic and formal definitions on RSOA

Constraints are a powerful mechanism for enforcing higher level organizational policy. A common example is that of mutually disjointed organizational roles, such as those of a purchasing manager and an accounts payable manager.

Regarding RSOA, constraints can apply to the UA and PA relationships, user and roles for sessions, and expiration. When applied, constraints are predicates that return a value of “acceptable” or “rejection.”

Definition 1 (User-Role Authorization)

\[ u \in U \] is the subject of the user.
\[ rs \subseteq R \] is the subject of the roles.
\[ \exists u, rs >\in E \] is the subject of expiration.
\[ UR \] set is the of U-R authorization tuples.
one to many assignment relationships.

The above tuple states that the user \( u \) has been made the member of role \( rs \). The user can be associated with one or more roles, and the role can have one or more users. Roles are created for various service positions and privileges in an SOA. The critical typical of roles is that the set of roles may have an expiration time; that is, the services must have a small probability of change.

**Definition 2 (Role-Permission Authorization)**

\( R-P \) authorization is a three-tuple \(<r, ps, os>\):

- \( r \in R \) is the role in the SOA,
- \( ps \subseteq P \) is the subset of permissions,
- \( os \subseteq O \) is the subset of services,
- \( RP \) is the set of \( R-P \) authorization tuples.

The \( RP \) tuple states that the role \( r \) has been granted the permission \( ps \) on the object \( os \). The permission can be for specific types of services that depend on the type of applications. For example, for patient data, history might be visible or hidden; for hospital information system, patient data can be visual or hidden from the user. Authorization can be derived in relation to any of the definitions. We use the concepts of implications between authorization rules. Therefore, according to Definition 1 and Definition 2, we present the derivation rule that can acquire the user authorizations from user-role authorizations and role-permission authorization. This is the first authorization implication rule.

- **Rule 1. (user permission authorization)**
  
  \[
  \exists u, rs >\in UR, \exists r, ps, os >\in RP \text{ and } r \in rs
  \]

  Rule 1 establishes that if the user is the member of one role, and the role has the authorization for the services, then the user can be granted the authorization to access the services \( os \).

**Definition 3 (Role-Hierarchy Authorization)**

Hierarchies are a typical means for structuring roles to reflect an organization’s lines of authority and responsibility.

\( R-H \) authorization is a two-tuple \(<r_1, r_2>\),

- \( r_1 \) is the parent role,
- \( r_2 \) is the child role,
- \( RH \) is the set of \( R-H \) authorization tuples.

This tuple states that role \( r_1 \) and \( r_2 \) have the forefather relationship. The immediate parent relationship can also be represented as an ordered pair, where \( r_1 \) is the immediate parent and \( r_2 \) is the child. The role \( r_1 \) inherits all permissions from the role \( r_2 \). The relationships can create some hierarchies. Role hierarchies are a natural way of organizing roles to reflect authority and responsibility. The hierarchies are partial orders. Therefore, the most potent roles are represented at the top of the hierarchies with the less potent roles being represented. For example, the primary-care physician role inherits permissions form the physician and health-care-provider roles. The primary-care physician and specialist physician both inherit permissions from the physician role, but each has different permissions directly assigned to it.

We present the rule that can derive some new authorizations from role-permission authorizations. The rule is defined as follows.

- **Rule 2. (role inheritance authorization)**
  
  \[
  \forall r_1 \in R, r_2 \in R, ps \subseteq P, os \subseteq O
  \]

  \[
  \exists r_1, r_2 >\in RH \text{ and } \exists r_2, ps, os >\in RP
  \]

  then \( <r_1, ps, os >\in RP \)

Rule 2 establishes that if one role has the authorizations of the services, all other roles that precede this role in the partial order hierarchies are granted the same authorizations.

4.2. The Authorization of RSOA

When the number of services in the SOA is extremely large, specifying authorization for each single application becomes ineffective. The approach to overcome the problem is to allow authorizations to be specified in the authorization of RSOA. The authorization of RSOA is the set of services for administrative purposes. Several services for supporting the SOA have been proposed,
supporting directories subject to the same domains. Directories provide an easy and intuitive approach to services. However, if several services belong to the same directory that is defined as the authorization of RSOA, they are granted the same authorizations without exceptions. If the services are from different sites, the administration approach ensures that authorizations are difficult to maintain and manage. Therefore, we propose the new authorization of RSOA. Because the services at each site in RSOA are constructed in the presentation tree, according to the features of the tree hierarchy, we can define the sub-tree of the presentation tree in the new authorization of RSOA. This approach is more appropriate for the protection of service in RSOA.

In the proposed model, the services specified for authorizations may be the non-authorization domains and authorization domains. Each authorization domain may be directories and sub-trees of each presentation tree in the RSOA. Each sub-tree is defined at some site. The authorizations are defined as follows.

**Definition 4 (Service Hierarchy Inheritance)**

An $S-H$ inheritance is a two-tuple $<o_1, o_2>$, where $o_1, o_2 \in S$ are the services in the SOA, and $o_1$ is the parent service, and $o_2$ is the child service. The $SH$ set is the set of $S-H$ inheritance tuples.

The definition is added to the RBAC model representing the structure of the information at a site in the SOA. An $SH$ tuple states that the service $o_2$ is a component of service $o_1$. That is, service $o_1$ may connect to service $o_2$ using a link or service $o_2$ is the basic service or specific service. Notice that service $o_1$ must be a node and service $o_2$ may be any type of information.

According to the aforementioned definition, we present a derivation rule that can glean a new authorization from role-permission authorizations. The derivation rule is defined as follows.

- **Rule 3. (service inheritance authorization)**
  For each $o_1 \in O$, $o_2 \in O$, $ps \subseteq P$, $r \in R$, and $os \subseteq O$,
  If $\exists <o_1, o_2> \in OH$, $\exists <r, ps, os> \in RP$, and $o_1 \in os$,
  Then $<r, ps, osa> \in RP$, where $osa = os \cup \{o_2\}$

In Rule 3, when the role is granted a specific permission on $o_1$, a user is granted membership in the role. In addition, the user can be granted the permission $ps$ on the service $o_1$, as well as ownership in the permissions for the service $o_2$, depending on the type of application in the service $o_2$.

**4.3. The authorization types and methods of RSOA**

The authorization types and methods specify the types of accesses that may be performed on different types of service. The permission is an approval for the particular method of access to one or more services in SOA. The characteristics of the permission depend largely on the SOA. For example, the services in the SOA involve operations such as bind, publish, and find. In the SOA, the protected services include relations, tuples, attributes, and views that are protected by services such as select, insert, update, and insert. Because the SOA comprises multiple services with differing characteristics of object and application, each service has its special method of access with differing functionalities. Hence, the authorization model for the SOA must provide different access types and methods that define the various ways subjects are authorized to achieve access.

The RBAC as an access control mechanism is the concept in which an operation may theoretically be anything. Therefore, our RSOA authorization model can define the different operational approval that is associated with different types of service.

When the new type of resource is joined to the SOA, based upon the feature and functionality of the resource we define the new permission. The consent for authorization for the different types of service is defined as follows.

**Definition 5 (Service-Permission Authorization)**

The authorization types and modes specify the types of accesses that may be performed on different types of service or object. The permission is an approval for a particular mode of access to one or more service in the SOA. The nature of the permission depends largely on the system and implementation.
An S-P authorization is a two-tuple \( <o, p> \), where \( o \) is the service, and \( p \) is the permission. The authorization changes the set of permissions \( P \) and the set of service \( O \) as follows. Extend the permission set \( P \), \( P = P \cup \{p\} \). In addition, if \( o \) is a new type of service, then extend the set \( O \), \( O = O \cup \{o\} \).

Authorization \( <o, p> \) states that the administrator can define the new permission \( p \) that is associated with the service \( O \).

### 4.4. the expiration authorization of RSOA

**Definition 6 (Role-Expiration Authorization)**

A R-E authorization is a three-tuple \( <r, ps, os> \), where \( r \in R \) is the role in the SOA, \( ps \subseteq P \) is the subset of permissions, and \( os \subseteq O \) is the subset of services. The RE set is the set of R-E authorization tuples.

An RE tuples states that the role \( r \) has been denied the permissions \( ps \) on the services \( os \). Our authorization mechanism is based on the assumption that some subjects have no authorization of the same services unless it is explicitly authorized.

- **Rule 4 (Service Expiration Authorization)**
  For each \( r \in R \), \( ps \subseteq P \), and \( os \subseteq O \)
  If \( \exists <r, ps, os> \in RE \),
  then it constrains the implication rule, Rule1.

Rule 4 establishes that if the role has negative authorizations on some services, then the positive authorization granted to the subject becomes blocked.

### 5. Component of authorization administration based on SOA

This section proposes the system framework that provides the role-based authorization model described in the previous section for the SOA. The framework integrates internal hospital functions and patient participation, presenting a number of security concerns, such as how to identify users and whether users have the proper authorization to access the mechanism. The fundamental obstacle, however, involves ensuring high-quality service because the system’s structure is an SOA. The administration of approval is complex. Traditional administrative policies are based on owner or centralized administration. Therefore, our authorization system involves two-level administrations, and provides non-authorization role-based access control. In our authorization system, responsibilities of administrators can be divided into central and local protection domains. The central protection domain has the system administrator, who is responsible for the creation of central protection policies. Each site in the SOA is the local protected domain. The administrator of each site manages the protection concerns. According to the authorization model discussed in Section 4, how the functions of authorizations and the inference of the derivation rules in our authorization model are embedded into the different administrators is described in the following subsections.

Our authorization system consists of two administration components: home care server (the Internet) and hospital care server (integrating Intranets). The responsibilities of the two administration components are introduced in details as follows.

1. Home care server (Internet): the home care server in the SOA is a unique server that generates all roles in the SOA. The home care server assigns the authorization of users as members of roles.
   - **Creating roles:** first, the home care server administrator generates all roles in the SOA. All roles are composed of the role set \( R \). The role is the set of actions and responsibilities associated with the particular organization function.
   - **Constructing role hierarchies:** the home care server administrator can grant the role-hierarchy authorization in the SOA. The RH set maintains the relationships among roles in the SOA, and constructs the relationships into role hierarchies. The role hierarchy defines roles that have unique attributes that may contain other roles. One role can be granted some permission on some services. In addition, the role may include the operations that are associated with other roles that are lower in role hierarchies. Because roles have overlapping responsibilities and privileges, the administrator can grant common operations to the bottom roles. This reduces the
cumbersome processes of specifying repeatedly the common operations for each role. This contributes to the efficiency of the administration.

- Granting user-role authorization: When the new user registers into the SOA, according to the user who determines the roles or owns the access rights in the SOA, the home care server authorizes the user as the member of roles by granting operation for user-role authorization. Meanwhile, the home care server assigns an account and gives the credential to the user. The credential contains information about user roles, user identity, and the validity period.

In addition, the home care server administrator must be concerned with constraints for granting users as members of appropriate roles. Constraint is an essential aspect of RBAC. The main constraint involves mutually complete roles, because some roles in the hospital may be mutually disjointed. This constraint is the principle of separation of duties. Finally, the home care server transmits the role set $R$ and the RH set to the hospital care server at each site.

2. Hospital care server (integrating Intranets): the hospital care server of each site can grant permission to those roles on the services stored at the site; it has also the mechanism for confirming authorization when the user requests to have access to the services at the site. The details are described as follows.

- Constructing/modifying services: first, the hospital care server administrator organizes the information stored at the site into the presentation tree. When the services at the site must be added, deleted, and modified, only the hospital care server administrator and authorized users possess the advantage. General end users are not granted the authorizations.

- Creating/modifying authorization domain: the hospital care server administrator may create and change authorization domains by authorizing operations for service-hierarchy inheritance respectively, and generates the SH set. The SH set presents the inheritance relationships among services stored at the site, and constructs them in service hierarchies.

- Creating new permissions: when the new type of resource is joined to site in the SOA, the hospital care server administrator must consider the feature and function of the application, define the new approval using service-permission authorization.

- Presenting/repealing role-service authorizations: according to the security policy of the SOA, the hospital care server administrator may present or repeal approval to roles on services using the operations for role-service authorization.

- Authenticating users: the hospital care server establishes user authentication using the credentials of the user, and confirms whether the validation period is expired. The purpose is to authenticate the legal identity of the user. If the user does not accredit the authentication process, then the user has no chance to enter the system.

- Service expiration: when the user requests access to services stored at the SOA, the credentials of the user must be verified to confirm the user’s identity. The service situation maintains itself in favorable condition.

6. Conclusions and future work

This study proposes a role-based authorization model for the SOA. We extended and redefined some characteristics with definitions for the RBAC model, adequate for the protection of information in the SOA. The characteristics include the service inheritance, consideration of different types of service, and negative authorizations for denying roles to access services.

In contrast to other authorization models [11,20], our authorization model and system for the administration of authorizations have the following advantages and features.

1. We define the roles by services, assumptions, and scopes of authority. By adding the device between the user and its scope of authority, the user can assess the authority through this intermediary device, and can simplify the process and greatly reduce the administrative burden.

2. If the permission for services and roles in the SOA are altered, only the related server must modify the authorizations.

3. We provided the different access permissions for various types of information in SOA. We consider the features of the various types of information to provide the function of defining the new permission.
4. Using our model, the transfer between the individual and the role is simplified, and the role and permission allocation can readily be performed. The exchange of service information and system maintenance are facilitated, and the security of the database and system are enhanced. Further works in process ensure that our authorization model and system is more complete. The development of the SOA approach for RBAC configuration design and analysis is the first concern. Another concern is that several practical protocols in our work should be integrated.

7. Acknowledgment

This project is supported by the National Science Council, Taiwan (grant number NSC99-2221-E-320-005).

8. References


