A Role based Access Control for Web Services

Roosdiana Wonohoesodo Zahir Tari
RMIT University
School of Computer Science and Information Technology
GPO Box 2476V, VIC 3001 Australia
zahirt@cs.rmit.edu.au

Abstract

Web services are vulnerable to various types of security attacks. This paper addresses one type of attacks, where applications trying to access services to which they are not authorized. Existing access control for web services lack of support for global services. As such services are WAN-based, therefore access control needed to deal with various levels of web services, including global (for composite services) and local level (for web servers). This paper proposes two access control: SWS-RBAC (for single services) and CWS-RBAC (for global services). Instead of protecting the content of the service's parameters, these models protect the parameters themselves. The proposed approach introduces global roles which are used in the mapping to local roles of other service providers. To maintain the autonomy of roles between providers, an efficient role-mapping mechanism has been proposed accordingly.

1 Introduction

Web services are loosely coupled applications, which use well-known XML protocols (like WSDL, SOAP and UDDI) for representation and communication across the Internet. The open nature of the Internet makes such services vulnerable to security attacks. The fact that Web services send and receive requests through the HTTP port further contributes to the vulnerability of web services. This papers addresses one aspect of security vulnerabilities in web services, that is, whether or not the requester of the service is an authorized user and whether or not the user is indeed authorized for the service requested. This paper does not deal with the authentication of the service requester or the authenticity of the service request calls. The aim is to develop security policies and a framework to efficiently enforce access restrictions so that access is granted only to authorized users, suitable for web services, easily manageable and extendable as the services grow. As the scope of web services grows, the authorization rules and the users grow along with it, which could be a nightmare to manage if the management of these rules is not naturally efficient.

Several access control have been proposed to deal with the efficient management of access rights. These include DAC (Discretionary Access Control), MAC (Mandatory Access Control) and RBAC (Role Based Access Control). However such access controls are designed for LAN-based applications (such as banking applications), which make them not really appropriate for Internet applications, unless, of course, they are extended in an appropriate way to deal with the specific requirements of web services. Apart from the issue of finding suitable access control policies and adapting them (to work in the context of web services), the issue of implementation of security policies must be also taken into account. As web services are independent of external security tools (such as firewalls), access controls must not return its dependency to firewalls. Access controls must therefore be robust and efficient and not dependent on the firewall or the service's application code.

Probably the most significant work in relation to access control for web services is by Damiani in [4]. His proposal enforces access control policies on the request calls carried by SOAP. The proposed model exploits the XML structure of SOAP calls. Objects are path expressions of the request call's XML structure. Subjects are identified on the basis of locations from where the request is originated, user identifications, user groups and user roles. Unlike traditional access control models, where subjects are granted various modes of access to objects, this model only recognizes two types of access, allow and deny (symbolized by '+' and '-' respectively). By taking advantage of the XML structure, this model provides a simple yet efficient way to enforce access control. However, this model does not provide a solution for global services. Global services are one of the important features that are offered by web services, that is, the ability to request other services available on the Internet and integrate them to create composite services. Hence, this limitation is rather significant.

The contributions that this paper makes include proposing an access control for web services. This model is the adaptation of the traditional Role-based Access Control (RBAC). Unlike traditional RBAC [8], where objects are passive entities, objects in the proposed model are both active and passive entities. Active entities are services, and passive entities are the service's parameters. In a web services environment, both services and their parameters must be protected from unauthorized users. Therefore the proposed model enforces access control at both the service level and parameter level to ensure that users who have permission to call a service also have appropriate access on its parameters to successfully execute the service.

The proposed access control also provides a solution to some limitations of the model proposed in [4] by extending the basic RBAC model to address global services (which are services that are made up of services from different providers). In this extended model, we introduce a concept of global role. The user, when calling global services, uses global roles. Global roles are different from traditional roles, because they hold the information of roles from other providers that are mapped to them. The model provides an efficient mechanism of role mapping that does not require important maintenance.

This paper is organized as follows. Section 2 provides a summary of existing research works in access controls. The proposed RBAC models for both single services respectively global services are described in Section 3 and Section 4 concludes our paper.

2 Related Work

There has been a little work done in the area of access control for web services. Most of the work in the area focuses on the language to express access control policies in a uniform way. Extensible Access Control Markup Language (XACML) has just gained acceptance from the Organization for the Advancement of Structured Information Standards (OASIS). This standard is designed to alleviate the patchwork of access control policies that are written in proprietary languages specific to each device or application.

The model proposed by Damiani in [4] is probably the most significant work in relation to access control in web services. This model exploits the XML structure of SOAP calls. The elements of this access control include: authorization objects, authorization subjects, and authorization rules (which are expressed in XML syntax). This provides a simple yet efficient way to enforce access control policies directly to XML structure. However, it has several limitations. 1) The model uses the authorization rules to keep track of subjects' permissions or denials on objects. Storing authorization rules in this way is inefficient. If an authorization rule represents what objects a subject has permission to, an evaluation of all of the available rules is only needed when querying who has permission on an object. 2) The authorization filter allows requests to be rejected, passed as is or modified based on authorization rules. The fact that a request can be modified assumes that services support modified requests. If a service is configured in such a way that it supports optional parameters, such as overloading methods, the service will be able to carry out the request as normal. However, if the service does not support optional parameters, it will generate an error. Therefore, the fact this model makes an assumption that services support optional parameters and this assumption is not appropriate. 3) The model does not deal with composite/global services, which are fundamental for web services. Global services are services that exploit single services available on the Internet and integrate them to create added-value services.

Access control that has been given a lot of attention that may be relevant to web services is in XML. Damiani [3, 5] and Bertino [1, 2] proposed several solutions that restrict users from viewing part of an XML document that they are not authorized to see. The common approach to enforce such access control is by pruning elements of the XML document that the users are not authorized to see. However, these works are not directly relevant to web services.

The followings illustrate the differences between our approach and existing solutions:

- The proposed models are not concerned with language or mechanism for applying an authorization policy to request calls, rather they are concerned with how the elements of access controls are organized in order to efficiently meet the security requirements. The main security requirements for access policies for web services is the ability to ensure that only authorized users are allowed to access the functions provided in web services. As services offer their services by publishing them on the Internet, this means that everyone can see and send requests to them. Access control policies must be able to grant access to requesters that are authorized and deny access to unauthorized requesters.

- While XML structure is used to send and receive requests from/to web services, we believe the enforcement of access rights should based neither on XML structures nor on the content of such structures (as suggested in [4]). Instead, we ensure that the requester has appropriate permissions on both the services and the supplied parameters, before the request is permitted to pass through. In the proposed models, a request can only be rejected or accepted. We do not support modification of requests as in [4], since we do not make an assumption that all services support overloading methods. Hence, the XML tree pruning mechanisms [2, 1, 5, 3] are not relevant to our problem.

3 An Access Control for Web Services

This section proposes the adaptation of RBAC for web services. We divided the adapted RBAC into two models. The first model is the basic one, which only addresses single services. This is later extended to deal with global services.

3.1 RBAC Model for Single Web Services

This section discusses the details of the basic adapted RBAC model. We call such model, Single Web Services - RBAC (SWS-RBAC). SWS-RBAC enforces access control at two levels, service level and attribute level. Service level access control ensures only authorized consumers can use the service. Attribute level access control is applied only when the authorized consumers pass the enforcement of access control at the service level. Attributes are objects that are used by the service as parameters or returned values.

Services, which use attributes as parameters, must assign a minimum permission to each of the attributes. This minimum permission is used to enforce access control at the attribute level. In order for the service consumers to pass access control at the attribute level, they must satisfy the minimum permission requirement on every attribute used by the service. Permissions on attributes are represented by attaching access modes to the attributes. Although services use attributes as parameters, attributes are independent of services. This way, the same attribute can be used by different services.

3.1.1 Elements of SWS-RBAC Model

Figure 1 depicts the model of SWS-RBAC, the access control elements and the relationships between them. A role is assigned to one or many users. A role is granted one or many services, and granted access modes to one or many attributes. A service has minimum access modes applied to one or many attributes. One or many access modes can be applied to an attribute. An access mode can contain itself. A role can be derived from itself.

A user is the service consumer. A user is given a unique identity within a service provider. In order for users to be able to invoke any services provided by the provider, they have to be assigned at least one role. Roles maintain a list of services within a service provider that users, who are assigned the roles, have permission to execute. In addition, roles maintain a list of attributes the user has access to and the access types. For a role to be able to execute a service, it must be granted permission to the service. As discussed before, access controls are imposed at two levels, service level and attribute level. Having granted a service to a role does not guarantee that the role is able to execute the service. Once a role passes the access control at the service level, it must also pass the access control at the attribute level.

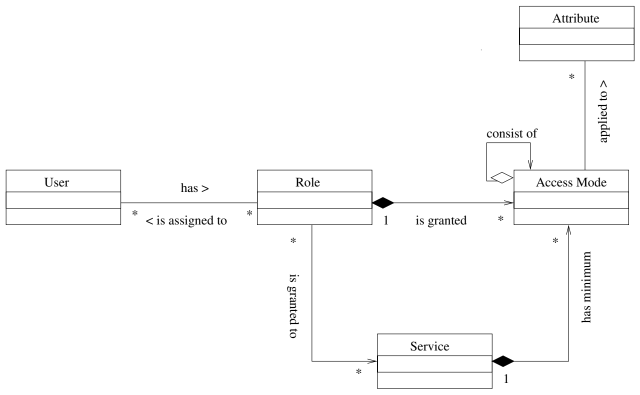

Figure 1. The elements of SWS-RBAC

Roles can be created and ordered in a hierarchical manner. Within a role hierarchy, a role can contain other roles. A role that contains other roles inherits all the services, attributes and access modes, which are assigned to the roles contained within it, as well as the services, attributes and access modes assigned directly to it. An example of multiple inheritance could involve Supervising Programmer which inherits from Testing Programmer and Analyst Programmer. Supervising Programmer must be able to perform the responsibilities of Testing Programmer and Analyst Programmer. Hence, even though Analyst Programmer and Testing Programmer may hold conflicting access modes on the same attribute, Supervising Programmer must have all of the access modes on the attribute in order to perform its subordinate responsibilities.

Table 1 shows examples of services, attributes and access modes assigned to roles. Testing Programmer is granted permission to call services create_‘project and change_title and access modes R (Read), W (Write), X (Execute) on attribute project and access mode M (Modify) on attribute title. On the other hand, Analyst Programmer has permission to call modify_project and access modes R (Read), W (Write), and D (Delete) on the attribute project.

Table 1. Examples of role capability lists.

Role	Service	Attribute:Access Mode
Testing Programmer	create_project, change_title	project:R&W&X, title:M
Analyst Programmer	modify_project	project:R&W&D

An access mode is a type of access a role can be granted to the services' attributes. For example, a role R1 is granted an access mode R, which stands for Read, on an attribute param1, which is used by service S1. This means that R1 is granted permission to read param1 used by S1 as its parameter or returned value. An access mode can be a composite access mode, where one or more access modes can be combined to form a new access mode. The SWS-RBAC model shown in Figure 1 depicts the access mode class as having aggregation relationship with itself. This means that an access mode can be part of other access modes. For example, an access mode M (Modify) is a composite access mode that contains other access modes, R (Read), W (Write) and Execute (X).

Roles that are granted a composite access mode are also granted access modes that are contained within it. For example, if a role, R1, is granted access mode M on attribute param1, R1 is also granted access mode R, W, and X.

Attributes are objects that are passed in to services or out of services. They refer to the service's parameters and returned values. Service's parameters can perform two functionalities, as a carrier of the information required for the service to perform its tasks, or as a carrier of the service's operation result. Attributes have access modes applied to them in order for them to be useful to the services that use them. The services, which use attributes as the parameters or returned values, must have some access to these attributes to make use of the information carried within the attributes. A service must at least have R (Read) access to its input parameters, W (Write) access to its output parameters and M (Modify) access to its input/output parameters.

Let us consider two attributes, namely *project* and *title*, with the following access mode: {R,W,X} (for project) and {R} (for title). If this assignment is applied to a role, the role will have access modes R, W, X on attribute project. On the other hand, if this assignment is applied to a service, the access modes R, W, X represents the minimum access modes required on attribute project to call the service.

Services are methods that are made available by the service provider so that roles, which are assigned the services, can call the methods to take advantage of the services it offers. Services use attributes as parameters or returned values because they need the attributes to bring in information to the service to complete the tasks or they need the attributes to carry the result of the service out. In order to make use of the attributes, services must have appropriate access to the attributes. Table 2 shows examples of service declarations; get_project uses the attributes title and project parameters: title and project requires a minimum access mode R and W respectively. On the other hand, service change_title uses the same attribute title. However, in this service, attribute title must have minimum access mode M.

Table 2. An example of services and their access modes.

Service	Attribute : Access Mode
get_project<[title, project],->	title:R, project:W
change_title<[title],->	title:M

The same attribute can be used by one or many services, with different minimum access modes. In the programming term, the term attribute here corresponds to the object type of a method's parameters.

3.1.2 Permission Granting

This section discusses how a user who nominates a role is granted permission to call the requested service. Users must be assigned at least one role before they can invoke any services offered by the service provider. In order for a role to be able to invoke a service on behalf of the user, it must be granted permission to the service and at least the minimum access modes on the service's attributes. The proposed model imposes access control at two levels; service level and attribute level.

The decision to grant permission to invoke a service is based on the following conditions: (i) Does the nominated role have the requested service granted to it?; Does the nominated role satisfy all of the minimum access requirements imposed on attributes used by the service?

The two above conditions must be satisfied in order for the role to be granted permission to invoke services. The services, attributes and access modes do not have to be assigned directly to the nominated role; they can be inherited from its subordinate roles. An example of a system employing SWS-RBAC is shown below. We consider two users, namely User01 and User01, with roles Manager and Employee respectively. Table 3 depicts examples of roles and their sub-roles. Role Employee in this table is a stand-alone role. It is not derived from other roles. Project_Member and Developer, on the other hand, are both derived from role Employee. Project_Leader is derived from both Project_Member and Developer. Manager is derived from Project_Leader. These roles have formed a role hierarchy where Manager is at the highest level and Employee at the lowest level in the same hierarchy line.

Table 3. An example of a role hierarchy.

Role	Immediate Base Roles
Employee	-
Project_Member	Employee
Developer	Employee
Project_Leader	Project_Member, Developer
Manager	Project_Leader

Table 4 shows examples of roles, which are assigned services and access modes to attributes. Role Employee is not granted any services, but it is granted R access on attribute title and W access on attribute project. On the other hand, Role Project_Member is granted services get_project and modify_project, but none on attributes.

Table 4. Examples of role capability lists.

Role	Service	Attribute:Access Mode
Employee		title:R, project:W
Project_Member	get_project, modify_project	
Developer	create_project	project:R&X
	change_title	title:M
Manager	allocate_resource	project:F

Table 5 depicts examples of services and the minimum access modes required on the service's attributes. Service get_project uses attribute title and project as its parameters. This service requires minimum R access on attribute title and W access on attribute project.

Table 5. Examples of services and their constraints.

Service	Attribute: Access Mode
get_project<[title, project],->	title:R, project:W
change_title<[title],->	title:M
create_project<[title, project],->	title:R, project:W
modify_project<[project],->	project:M
allocate_resource<[resource, project],->	resource:R, project:M

As depicted in Table 3, User01 is assigned a role Manager. This means that User01 can nominate either role Manager or Manager's subordinate roles, for example Project_Member or Employee, when invoking a service. If User01 nominates role Developer upon requesting service allocate_resource, User01 will be denied access. Service allocate_resource is indeed assigned to role Manager, which is assigned directly to User01. However, User01 chooses to nominate role Developer. Therefore, the permission is granted based on what role Developer has. Role Developer is granted services create_project and change_title. Service allocate_resource is not granted to either role Developer or its sub-role, which is Employee. Thus, role Developer is denied permission to call service allocate_resource.

The same user User01 with its nominated role Developer is granted permission to invoke service create_project. Role Developer has permission to invoke service create_project, which makes it pass the access control at the service level. User01 possesses M access for attribute title, which is more than the service's minimum requirement, R. Even though Developer only has access R and X on attribute project applied directly to it, it inherits access W from its subordinate role, Employee, which is equivalent to access M. Hence, that makes User01 pass the access control at the attribute level. Since, User01 passes both access controls at the service and attribute level, he/she is granted permission to invoke service create_project.

One of the advantages of breaking the access controls into two levels is preventing roles from invoking a service temporarily can be done easily by increasing the minimum access mode requirement and return it back to its original later as easily. If the access controls were imposed only on the service level, the service would have to be removed from the system and from the roles it is associated with, then recreated and re-assigned later, which could be tedious.

3.2 RBAC Model for Composite Web Services

Web services platforms provide the opportunity to create added-value services, called composite services. In this paper we will use the term global service to refer to the concept of composite service. In this way we can distinguish it from a local service, which is a service that completes its tasks locally. This section discusses the RBAC model for Composite Web Services, called CWS-RBAC, which is an extension of SWS-RBAC for local services

3.2.1 Overview of CWS-RBAC

CWS-RBAC adopts the same basic principle as SWS-RBAC. The access controls are imposed at two levels, service level and attribute level. The access control at the service level checks for the service assigned to the role. The access control at the attribute level checks if the access modes the role has on the attributes satisfy the minimum access modes, imposed by the services on attributes they use. Global services in CWS-RBAC consist of local services from other providers. When a user requests a global service, the global service performs its tasks on the behalf of the user, which means it uses user's privilege to perform its tasks. When a global service calls local services of other providers, it calls the local services on behalf of the user. Users have different roles on different providers. Therefore, upon calling local services from other providers, the global role, which is nominated by the user upon calling the global service, must be mapped to the roles.

The key objective of CWS-RBAC is to be able to map global roles to local roles (which are known to other providers with minimal dependencies between providers) and to call a service. This means that the provider must only give enough information on its roles to the provider, which consumes the service, to perform role mapping. The role information that is given to other provider must not be too detailed so they must be updated whenever the originated provider restructures its roles. The restructuring of roles belonging to a given provider must be transparent to other providers which use its services. To achieve autonomy of roles between providers, we propose that a provider, whose services are involved in other providers global services, allocates different sets of roles for different providers. These sets of roles must consist of the minimum roles required to call those services.

Allocating different sets of roles for different providers enables the owner of the service to determine where the request on behalf of the user originated from. Hence, the provider, which owns the services, can have full control of who can call its services and from where. If a user is allowed to call a service from a provider, the user will be assigned the minimum role to call the service, which is allocated to the provider. Since the roles allocated by other providers are the minimum roles for their supplied local services, a global role that is granted permission to call a global service is mapped to the local roles of the local services which are part of the global service. The role mapping will be discussed further, later in this section.

3.2.2 Elements of CWS-RBAC

Figure 2 depicts the elements of access controls involved in CWS-RBAC model and their relationships with each other. In this model, we introduce a new element called Provider since there is more than one provider involved in composite web services. We distinguish services into global services, and local services. A global service consists of local services or global services from other providers. Roles in this model are also distinguished into global roles and local roles. A global role consists of local roles or global roles from other providers.

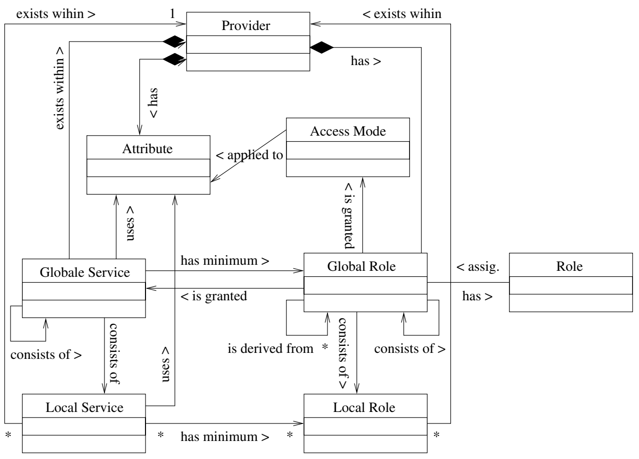

Figure 2. The RBAC Model for Composite Services.

In this model, a user is assigned one or more global roles. A global role is granted permission to call one or more global services and access modes applied to attributes that exist within the same provider. A global service uses attributes that exist within the same provider. Unlike SWS-RBAC, the global services do not apply minimum access modes directly to the attributes, instead they uses global roles, which possess the minimum access modes on the attributes of the global services. The reason behind using global roles to apply the minimum access is to achieve consistency in the mechanism of applying minimum access on both local services and global services.

Users are services consumers. They are active entities, which consume services provided by a service provider. Unlike its counterpart in SWS-RBAC, the user identification is not enough to identify the user. It is very unlikely for two providers to have identical user identification for the same user. Therefore, the user information that needs to be passed around must uniquely identify the user in a large community. It can be combination of user identifications, such as ABN number, Tax File Number, or ABN number and country name.

Providers are servers that provide services, which are either local services or global services, and publish them on the World Wide Web. They consist of global services, local services, global roles, local roles and attributes. In order for a provider to identify where the call originated, this model requires each of the providers, which is involved in a workgroup to carry out a global service, to allocate different set of roles for different providers in the workgroup. The same set of roles must not be assigned to more than one provider. A set of roles must be assigned exactly to one provider, so that when a provider receives a user-role activation request, it knows which provider sent the request on behalf of the user.

The advantage of having a provider allocate different sets of roles to different providers is that the provider, which owns the roles, still has complete control over its services. For example, if provider A is to deny access from provider B, it can simply remove any services associated to roles allocated to provider B. If provider A does not follow the role allocation rule and allocate the same role, for instance R1, to provider B and provider C, denying access to R1 would mean denying access to both provider B and provider C, which is undesirable.

Local services in CWS-RBAC are similar to services in SWS-RBAC. Local services are services that do not involve services from other providers to complete their tasks. Local services complete their tasks locally within the provider, in which they exist. The identification of local services within a provider must be unique among other services that exist within a provider. This means that no two services, which can be either local services or global services, can have an identical name within the same provider. However, two services can have an identical name provided that they exist on different providers.

Local roles are roles that hold permissions to call local services and access modes that the roles have on the local service' attributes. A local role in CWS-RBAC is the same as role in SWS-RBAC. Access modes in CWS-RBAC are the same as access modes in SWS-RBAC. In a similar way, attributes are objects that exist within a provider that are used by the services, either global or local, as their parameters or returned values. The name of an attribute must be unique within a provider. Two attributes can have an identical name as long as they exist within different providers.

Global services involve services from providers other than the provider where the global services exist. They are made up from either local services or global services of these providers and may or may not have attributes as their parameters or returned values. As services in SWS-RBAC, the global services perform their tasks upon user's requests. Hence, the operations within the global services are carried out within the user's context, using the user's privileges. The user may or may not have enough privileges to be used by the global services to complete their tasks successfully. Hence, the global services must enforce minimum privileges required from the user for the global services to be able to complete their tasks successfully.

A global service enforces the minimum privileges by having a global role assigned to it, which contains the permission to call the global service and the minimum access modes required on the global service's attributes. The reason behind using a global role to represent minimum permissions instead of access modes imposed on attributes (as in SWS-RBAC model) is to achieve some consistencies with the local services in CWS-RBAC (which use local roles to represent the minimum permissions required to call local services).

Table 6 depicts an example of a global service, its minimum global role and the local services from other providers that make up the global service. Service get_flight_quote is a global service because it calls service get_quote of provider AirlineAgent01 and request_quote of provider AirlineAgent02 to complete get_flight_quote tasks.

Table 6. An example of a global service.

Global Service	Minimum Global Role	Provider: Local Service
get_flight_qoute <[flight, quote_list],->	f_quote_min_role	AirlineAgent01: get_quote AirlineAgent02: request_quote

The global service get_flight_quote uses global role f_quote_min_role as the minimum global role required to call the global service. In order for the callers to be able to call get_flight_quote, they must have at least what global role f_quote_min_role has.

Global roles must be granted permissions to call global services for them to be able to use the global services. They can only be used to invoke global services; they cannot be used to invoke local services. Local roles must be used to invoke local services. Apart from having permission to global services, global roles also consist of access modes the roles have on the attributes that exist within the same provider as the global roles.

As in roles in SWS_RBAC, global roles can be ordered in a hierarchical manner, where a global role can be derived from other global roles. Global roles can only be derived from other global roles; they cannot be derived from local roles. Global roles, which are derived from other global roles, inherit all of the permissions that the sub-roles have, which include permission to global services and access modes to the attributes.

Unlike roles in SWS-RBAC, global roles must also consist of the local roles of other providers. Users have different roles with different providers. Upon requesting a global service, users nominate a global role. A global role is known only to the provider, which provides the global service. The other providers, whose services make up the global service, do not know the global role. Therefore, the global role must be mapped to the local roles that are known to their providers upon calling their local services.

Table 7 depicts an example of permissions assigned to a global role. Global Role f_quote_min_role is granted permission to call the global service get_flight_quote, which is defined previously in Table 6, and it has R access on attribute flight and W access on attribute quote_list.

Table 7. An example of a global role an its permissions.

Global Role	Global Service	Attribute:Access Mode
f_quote_min_role	get_flight_quote	flight:R,quote_list:W

A global service get_flight_quote is made up from the local service get_quote from AirlineAgent01 and local service request_quote from AirlineAgent02. Table 8 shows the local services and their corresponding minimum local roles of AirlineAgent01 and AirlineAgent02.

Table 8. Examples of providers' local services and local roles.

Provider	Local Service	Local Role
AirlineAgent01	get_quote <[flight],quote>	quote_min_role
AirlineAgent02	request_quote <[flight,quote],->	request_quote_min

Based on the information in Table 6, 7 and 8, the mapping between a global role and local roles can be built. Table 9 shows the result of the role mapping. The role mapping is built from the fact that the global role is assigned global service get_flight_quote. The global service get_flight_quote is made up of local services get_quote from provider AirlineAgent01 and request_quote from AirlineAgent02, which have minimum local roles quote_min_role and request_quote_min respectively. Therefore, the global role f_quote_min_role is mapped to quote_min_role from AirlineAgent01 and request_quote_min from AirlineAgent02. If a user nominates global role f_quote_min_role to call global service get_flight_quote, the global role f_quote_min_role can be translated to quote_min_role when calling local service get_quote and request_quote_min when calling local service request_quote.

Table 9. An example of role mapping.

Global Role	Provider : Local Role
f_quote_min_role	AirlineAgent01:quote_min_role AirlineAgent02:request_quote_min

3.2.3 Permission Granting

Permission to execute a global service is granted in the same way as in SWS-RBAC model. Users must nominate a global role before they can execute global services. As in SWS- RBAC, access controls are enforced at two levels, service level and attributes level. A global role must pass both the enforcement points before it can be granted access to invoke the requested global service.

Unlike SWS-RBAC, this model uses roles to represent the minimum permissions required to call a service. The services in this model have a minimum role attached to them. These simplify the authorization process, which checks for the following conditions:

- If the nominated global role is the minimum global role required to call the requested global service, then the authorization is passed.

- If the minimum global role of the requested global service is a sub-role of the nominated global role, the authorization is also passed.

If neither of the above conditions is met, the nominated global role must be evaluated against the minimum global role. The nominated global role must have at least what the minimum global role has. It can have more but it cannot have less than what the minimum global role has.

Although the nominated global role satisfies the minimum global role and is granted permission to call the global service, it does not guarantee that the global service would be able to complete its tasks. This is because the global service calls local services from other providers. Once the request is sent to the local services on other providers, the control is passed on to the other providers. The local roles attached to the request sent to the local services are the minimum local roles required to call the local services. The users, who call the global service, may or may not be assigned roles that satisfy the minimum local roles on the other providers. The users must satisfy all of the minimum local roles on every provider, which the global service calls the local services from, for the global service to be able to complete its tasks successfully.

4 Conclusion

Adaptations to role-based access control (RBAC) policy have been proposed to work in Web services environments. We have proposed two models of RBAC. The first model is the basic RBAC model for single web services, called SWS-RBAC. This model captures the relationships between elements of access control in SWS-RBAC. The enforcement of access control in this model is performed at two levels, service level and attribute level. The access control for composite services, called CWS-RBAC, deals with the specification of global security policies (by using global roles and appropriate mapping to local permissions). We have implemented the proposed models for web services. In the implementation, we have combined both models into a single implementation, which offers both single services, which we called local services, and global services. We also proposed to exploit the concept of handler supported by Apache Axis, the SOAP platform we used for the implementation. In enforcing the access controls we have modeled, we use two handlers, called Authorization Handler, which is responsible to perform authorization, and Role-Mapping Handler, which is responsible to perform the mapping of roles.

In the future, when a SOAP request call can carry more than one invocation of services, these models must be modified to support multiple role nominations. Consequently, dynamic separation of duty must is also considered.

Acknowledgments

This project is fully supported by ARC (Australian Research Council) and SUN Microsystems. The ARC Linkage project number is LP0347217.

References

[1] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-x: A java based system for XML data protection. In *IFIP Workshop on Database Security*, pages 15–26, 2000.

[2] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for XML document sources. *World Wide Web*, 3:139–151, 2000.

[3] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Design and implementation of an access control processor for XML documents. *Computer Networks*, 1999.

[4] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Fine grained access control for SOAP e-services. In *Proc. of 10th Int. Conf.on World Wide Web (WWW)*, pages 504–513, 2001.

[5] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security (TISSEC)*, 5:169–202, 2002.

[8] Z. Tari and S. Chan. A role-based access control for intranet security. *IEEE Internet Computing*, pages 24–34, 1997.