UDDI ACCESS CONTROL FOR THE EXTENDED ENTERPRISE

Robert Steele, Juan Dai
University of Technology, Sydney, PO Box 123, Broadway, NSW 2007

Keywords: UDDI, access control, extended enterprise

Abstract: An Extended Enterprise is comprised of not only the enterprise itself but also the enterprise's suppliers, clients and other associated organizations. The Extended Enterprise, in response to business needs and decisions, can dynamically alter these interrelationships, for example possibly swapping out some partners and swapping in others. Web services are an appropriate technology choice to facilitate the Extended Enterprise via supporting interoperability. Furthermore, the UDDI Web service standard and in particular a private UDDI registry can enable partner organizations to lookup and discover services of their new partners. As such a private UDDI registry is well suited to allowing potentially regularly changing business partners in an Extended Enterprise to determine how to interoperate with each other. However, different partners, depending on their role, should see a different set of the available services in an enterprises' private UDDI registry. This is for security, business confidentiality and simplicity purposes. As such in this paper we propose how a role-based access control scheme for a private UDDI registry can be utilized to support the Extended Enterprise.

1 INTRODUCTION

Web Services are designed to provide easier business-to-business integration among enterprises. Although a public Universal Description, Discovery and Integration (UDDI) service registry provides centralized information storage and retrieval, due to security concerns more and more organizations prefer to build their own private UDDI registries in their corporate network. In contrast to public registries, private registries are hosted inside a secure intranet or extranet and are only accessible by invited business partners via some user authentication mechanisms.

Such a private UDDI registry provides an appropriate hub for the infrastructure of an Extended Enterprise, as it allows dynamic discovery and lookup of services, supporting the dynamic partnering changes that are part of the Extended Enterprise (Goldman, 1994; Whitman et. al., 1999).

Since an enterprise may only want certain partners in the Extended Enterprise to see certain services while hiding the other services, access control inside private registries becomes essential. As pointed out by Adams and Boeyen (2002), confidentiality is one of the registry security requirements as some content in the registry should only be available to a particular subset of service requesters. To do this it should be possible to specify access permissions for every entry inside a UDDI registry.

An access control-enable UDDI registry needs to support a number of scenarios so as to be an appropriate addition to the infrastructure of the Extended Enterprise. It must be able to support the case of a new partnering relationship being created, a partner being removed and modification to a partnering relationship.

Although XML SOAP gateways produced by commercial IT vendors these days may be configured to provide a form of access control for Web services method invocation at runtime, we argue that this has disadvantages to access control in the UDDI registry. SOAP security gateways normally understand the WSDL (Web Services Description Language) documents of the web services and therefore are able to understand the data within the SOAP requests (Brose). They are able to verify message senders by checking the XML digital signatures or security assertions embedded inside SOAP headers and communicate with the policy server for access decisions (Brose). A gateway will then either reject the request or forward the request to the Web service server for processing based on the predefined security policies about WSDL service methods.

Steele R. and Dai J. (2005).
UDDI ACCESS CONTROL FOR THE EXTENDED ENTERPRISE.
In *Proceedings of the First International Conference on Web Information Systems and Technologies*, pages 176-181
DOI: 10.5220/0001232601760181

This feature of SOAP security gateways forbids unauthorized users to invoke Web services methods at runtime. However, this checking by the gateway leads to a performance degradation in service calling. Furthermore, access control inside the UDDI registry is still essential to prevent unauthorized users from reading certain information in the first place. It is desirable for security, business confidentiality and interface simplicity purposes to present to users only the set of services they have permission to access.

UDDI defines a standard way for businesses to list their services and discover each other on the Internet. An entry in the UDDI registry contains references to the WSDL service interface description file and the URL of the Web service. Typically when a user searches the registry for a service, he will get the service access point and a tModel linking to the WSDL description file. The user is then able to interact with the service based on the access point and the WSDL description file.

Thus access control can be enforced on entries inside the UDDI registry to allow only authorized users to access information about a particular service access point and tModel. For easier maintenance and configuration, security policies may be defined in XML and we can use the existing XML-based access control specifications such as XACML to exploit XML's own ability to build access control in the UDDI.

2 BACKGROUND

Extensive research has already been carried out on access control to XML documents built upon XML-based security policies. A number of access control models (Bertino et. al., 2001; Damiani et. al., 2002a, Gabillon & Bruno, 2001) were first proposed by academia to regulate access to XML documents. For example Gabillon and Bruno have proposed an XML authorization model for pull-mode access control. Similarly Damiani et. al. have proposed a fine-grained access control system for pull-mode access control. Bertino et. al. from the University of Milan have proposed a java-based Author-X system in 2001 for both pull and push mode access.

These models introduce XML-based access control to XML documents at both instance and schema level. They provide a way for security administrators to define security policies in XML format. Access control is based on the definition of subjects, objects and authorization rules. Subjects are user identifiers such as a login name which may be used to access the system. Objects can be nodes inside an XML XPath tree and are referred in XPath language. Finally there are authorization rules defining the access permissions for certain subjects to access certain objects. The syntax of authorization rules differs slightly from one model to another. However they all define rules about who can access what resources under which mode.

These authorization policies support access control at different levels of granularity, from DTD (Document Type Definition) schemas to individual documents to elements within those documents. Security policies specified at the DTD level will be applied to all derived XML documents; On the other hand policies specified on an element inside an XML document may be defined in a recursive approach applicable to all its sub-elements or in a non-recursive approach only applicable to itself (Damiani et. al., 2002b).

Once receiving the request, these systems check for every element inside the requested document the access privileges the user has on the element by referring directly to the security policies as well as calculating the implicit rules by propagation options (Bertino et. al., 2001; Damiani et. al., 2002b). The system fetches all relevant policies from the policy base applicable to the element at both the instance and the DTD levels. The system then applies its conflict resolution policies to eliminate possible conflicts. The pruning algorithm recursively processes each secure policy and marks the appropriate elements with "+" or "-". Finally, as indicated by Bertino, the system presents the requester with a view of the requested documents by pruning all unauthorized elements and attributes from the original documents marked with "-" sign.

If a system has a large number of users, access control policies may become tedious for security administrators to manage. On the other hand quite often access permissions to resources inside an organization are normally determined by the role a user plays. As pointed out by Ferraiolo and Kuhn (1992), this role definition normally involves "*specification of duties, responsibilities and qualifications*". Hence Role Based Access Control (RBAC) was first introduced in 1992 by Ferraiolo and Kuhn. With the Role Based Access Control model, users will be assigned with security roles. Security policies will be defined in the form of roles, objects and permissions. Resources availability is then determined by the role a user plays in the interaction. By grouping users into security roles, the Role Based Access Control model is a lot easier to maintain.

Some of the above mentioned access control models support role based access control by offering ways for security administrators to define users and user groups. For instance Gabillon and Bruno (2001) proposed using a separate *XML subject sheet* (XSS)

to describe users and groups. Users and their corresponding groups are defined inside XSS files in a hierarchical structure.

Based on these access control models, OASIS (2003) proposed the extensible access control markup language (XACML) specification. XACML specification defines standard XML-based policy language and access control/decision language. It is a non-proprietary language that provides standard and generic means for policy-based access control.

Logically a typical XACML system comprises a Policy Enforcement Point (PEP), a Policy Decision Point (PDP) and a set of policies. As shown in figure 1, typically a request trying to access a resource is sent to system's PEP. Based on the requester's attributes, the requested object and type of operation, PEP sends a new request to the PDP. The PDP examines the request, checks the policies from the policy repository, makes a denial or grant decision and sends the decision back to the PEP. PEP then grants or denies access to the requester. All security polices are defined and stored in a repository. Policy Combining Algorithm and Rule Combining Algorithm are used to build complex policies and handle conflict resolution (OASIS, 2003).

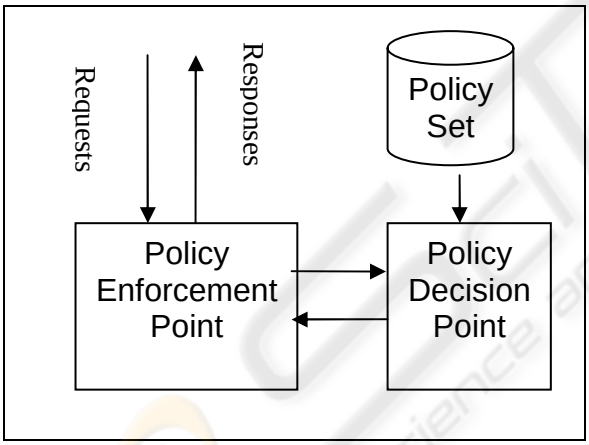

Figure 1: XACML system overview

To support role based access control, in February 2004, OASIS published the "XACML Profile for Role Based Access Control (RBAC)" specification, which defines the profile for use XACML for the Role Based Access Control. The specification defines normative profile to build role based access control in XACML as well as non-normative illustration of how to construct role based access control in XAXML (OASIS, 2004). Security roles can be expressed as attributes in XACML.

3 UDDI ACCESS CONTROL FOR IMPLEMENTING THE EXTENDED ENTERPRISE

The Extended Enterprise infrastructure architecture involves an access control-enabled private UDDI registry at each organization and a Partner Directory at each organization. This registry contains entries for all of the services provided by that enterprise.

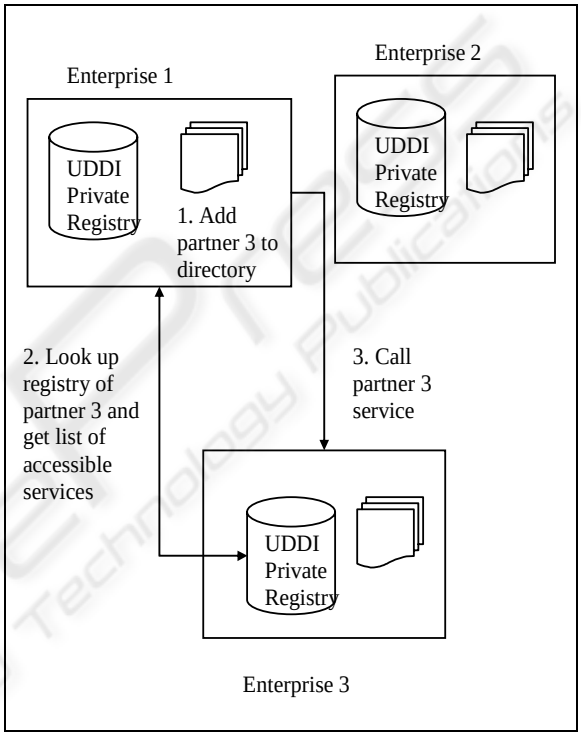

Figure 2: Extended Enterprise infrastructure

3.1 Partner Establishment

When an enterprise establishes a new relationship with another organization, the two organizations first record the relationship role the other organization has with it e.g. supplier, customer, sister company etc. This role is stored by each organization in its respective Partner Directory (see Figure 2). This Partner Directory will store the set of roles for each partner. The roles an organization and its partner have with each other may typically be complementary to each other.

The partners will now exchange URLs of their private UDDI registries, allowing each to now search and browse for services offered by the other. Based on the access control mechanism, each

partner only sees a certain set of the available services, based on their role.

The UDDI access control system will make use of the local Partner Directory in determining access. When a request to access the UDDI registry comes from a partner, the UDDI access control system will first consult the Partner Directory to determine the set of role(s) of that partner.

3.2 UDDI Access Control

Below we outline the Role Based Access Control Model that we propose to use for the Extended Enterprise infrastructure. More detail can be found in (Dai, Steele, 2005). This access control model is implemented with XACML, as XACML is a non-proprietary language that provides standard and generic means for policy-based access control and supports role-based access control. A typical role based access control model contains five basic elements: users, roles, objects, operations and permissions (OASIS, 2004). In our access control model, the objects to be protected are entries in a UDDI registry. The model will enable security administrators to define users, security roles and security policies all in XACML format.

Suppose suppliers are to have access to service *getSupplierQuote*. While suppliers may have access to the *getSupplierQuote* service, competitors should not. In this scenario the access control model can capture this functionality.

Below we give the example of how this can be encoded based on Oasis XACML Profile for Role Based Access Control specification (OASIS, 2004). A security policy may be first defined as shown in Figure 3 about granting access to service *getSupplierQuote*. The policy Id (PolicyId) in this case is *permissions:for:supplier:role*. This policy will be hooked to security role *supplier* by referencing the policy Id in the *supplier* role definition as shown in Figure 4. Hence Figure 3 and Figure 4 together define that security role *supplier* has the permission to access service *getSupplierQuote*. XACML also allows the associating of a user or enterprise with various roles. For example *Enterprise 1* can be associated with the role, *supplier*. When *Enterprise 1* enters the system, the XACML engine will be able to assign the privileges of the *supplier* role to *Enterprise 1*.

The proposed access control model will do the following after an enterprise authenticates with the private UDDI registry:

```
<Policy PolicyId=
"Permissions:for:supplier:role"
RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:1.0:
rule-combining-algorithm:first-
applicable”>
<Target>
 <Subjects>
   <AnySubject/>
 </Subjects>
 <Resources>
   </AnyResource>
 </Resources>
 <Actions>
   <AnyAction/>
 </Actions>
</Target>
<Rule RuleId=”SupplierPolicies”
Effect=”Permit”>
 <Target>
   <Subjects>
    <AnySubject/>
   </Subjects>
   <Resources>
     <Resource>
      <ResourceMatch MatchId=
       ”&function;string-match”>
       <AttributeValue DataType=
        ”&xml;string”>
          getSupplierQuote
       </AttributeValue>
     <ResourceAttributeDesignator
        DataType="&xml;string"
          AttributeId=
            "getQuoteSupplier"/>
     </ResourceMatch>
    </Resource>
   </Resources>
   <Actions>
     <Action>
       <ActionMatch MatchId=
       “&function;string-match”>
       <AttributeValue DataType=
       “&xml;string”>access
       </AttributeValue>
     </Action>
   </Actions>
 </Target>
 </Rule>
</Policy>
```

Figure 3: Policy for getSupplierQuote

1. Role assignment

The system will read role definitions declared in XACML to find authorized roles for the enterprise. Once found, it will assign the security role to the enterprise

2. Policy collection

The system will look at the policy set associated with that role. By checking every policy ID declared inside the policy set, the system will locate all policies the role is granted to access. Different policies assigned to a single role can lead to conflicts.

Conflict resolution will occur as per XACML conflict resolution. This information can then be used in building the partial views of UDDI search results or browse results subsequently.

```
…
<PolicySet
PolicySetId="supplier:role">
  <Target>
   <Subjects>
    <Subject>
      <SubjectMatch MatchId=
      "&function;string-equal">
        <AttributeValue
         DataType="&xml;string">
         supplier
        </AttributeValue>
     <SubjectAttributeDesignator
        DataType="&xml;string"
        AttributeId="supplierrole"/
>
      </SubjectMatch>
    </subject>
   </Subjects>
   <Resources>
    </AnyResource>
   </Resources>
   <Actions>
     <AnyAction/>
   </Actions>
 </Target>
 <PolicySetIdReference>
   Permissions:for:supplier:role
 </PolicySetIdReference>
</PolicySet>
…
```

Figure 4: Define the supplier role

The UDDI access control implementation will work by maintaining two cached lists for each authenticated user: one for determined accessible services and one for determined inaccessible services (Dai, Steele, 2005). These lists will survive for the lifetime of the enterprises' partnership and it is assumed that the access control policies do not change during this time. The system proposal can be easily adjusted so as check for policy changes during the lifetime of the partnership.

3.3 Partner Removal

When a partner relationship of some form between two enterprises terminates, that role for a partner organization will now be removed from the Partner Directory. This means that new requests to search or browse the UDDI registry will not be granted.

However, the security problem remains that the services already known of by the partner will still be known of and callable. One approach to overcome this is to generate for any given service, unique endpoints (URLs) per partner enterprise. These unique URLs would be generated when the new partnership is formed and requests to the private UDDI registry are made. For example, these unique URLs might contain the service name with a random string of digits appended.

When a partner enterprise ceases to be a partner, these URLs will cease to function as endpoints (will not be recognized by the SOAP engine). As such, future requests will not be responded to. The enterprise that was previously a partner will still know the service names but will no longer be able to access the services.

The approach of having UDDI-based access control as opposed to having a SOAP gateway check every incoming SOAP message for access permissions, has performance advantages. There will not need to be an access permission check on every incoming SOAP message, but rather permission evaluation is done just once, at UDDI lookup time.

3.4 Relationship Modification

A third case exists of partner relationship modification – that is, a change of Role. This case is treated like partner removal followed by a partner establishment but under a new Role. More efficient approaches are possible, where common services are maintained and only differing services added or disabled, but such approaches are not considered further here.

4 FUTURE WORK

Future work of interest includes extending the UDDI access control to be more fine-grained: that is providing access control on a per-method basis. The current system allows access or denies access to a whole service, which is described with a single WSDL file and which typically might have a number of methods. However, cases will occur where a user should have access to some of a service's methods and not others. In this case, one option is for a UDDI lookup to potentially "pruned" WSDL files – service description files showing only accessible methods and omitting others.

In this paper we have concentrated on access control for UDDI lookup activities. UDDI also has an API for publishing. That is, a set of services for

uploading or deleting services from the UDDI registry. This aspect of UDDI functionality also requires access control. For example, partner organizations of sufficient closeness may be able to publish new services or have permission to publish in only certain UDDI categories. Other organizations should not have write permissions to the private UDDI registry.

5 CONCLUSION

Businesses today are facing the increasing need for business collaboration among different partners through time. Web services provide a solution for easier B2B interaction and private UDDI registries inside corporate networks are used for efficient interaction between business partners. However since an organization may only want the right business partners to see only the right service information they are entitled to, some kind of access control mechanisms inside the private registry are required. In this paper we described how access control-enabled private UDDI registries can be used to provide an appropriate infrastructure for the Extended Enterprise. This approach to providing access control for Web services has some advantages over approaches based on SOAP gateways.

REFERENCES

Adams, C. & Boeyen, S., 2002. UDDI and WSDL Extensions for Web Services: A Security Framework, *Proceedings of the ACM workshop on XML security*, Session 2, p. 30-35.

Dai, J., Steele, R., 2005. UDDI Access Control. In *Proceedings of the 2005 International Conference on Information Technology and Applications*, Sydney, Australia.

Bertino, E., Castano, S. & Ferrari, E., 2001. Securing XML Documents with Author-X, *IEEE Internet Computing*, Vol. 5, Iss. 3, pg. 21-31

Brose, G. Securing Web Services with SOAP Security Proxies [Online], Available:
http://www.xtradyne.com/documents/whitepapers/Xtradyne-WebServices_Security_Proxies.pdf (Accessed September 2004).

Damiani, E., Vimercati, S., Paraboschi, S. & Samarati, P., 2002. A Fine-Grained Access Control System for XML Documents, *ACM Transactions on Information and System Security*, Vol. 5, No. 2, pg. 169-202

Damiani, E., Vimercati, S., & Samarati, P., 2002. Towards Securing XML Web Services. *Proceedings of the 2002 ACM workshop on XML security*, Session 4, p. 90-96

Extensible Access Control Markup Language Version 1.0 [Online], 2003, Available: http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf

Ferraiolo, D. & Kuhn, R., 1992. Role-Based Access Control. *Proceedings of 15th National Computer Security Conference*, pg 554-563

Gabillon, A. & Bruno, E., 2001. Regulating Access to XML Documents. *Proceedings of the fifteenth annual working conference on Database and application security*, pg. 299-314

Goldman, S.L., 1994. Co-operating to Compete: From Alliances to Virtual Companies, in *CMA*. p. 13-17.

Oasis, 2003. A Brief Introduction to XACML [Online], Available: http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html (Accessed September 2004)

Oasis, 2004. XACML Profile for Role Based Access Control [Online], 2004, Available: http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf

Whitman, L., Krishnan, K., Agarwal, R., Bhandare, P., 1999. Engineering the Extended Enterprise. *Proceedings of the 4th Annual International Conference on Industrial Engineering Theory, Applications and Practice*, Nov 17-19, 1999, San Antonio, Texas, USA.