Proceedings of the 5th WSEAS Int. Conf. on MULTIMEDIA, INTERNET AND VIDEO TECHNOLOGIES, Corfu, Greece, August 17-19, 2005 (pp5-10)

Unifying Private Registry and Web Service Access Control

KHALED KHANKAN, ROBERT STEELE, THARAM DILLON
Faculty of Information Technology
University of Technology, Sydney
PO Box 123 Broadway, NSW 2007
AUSTRALIA

Abstract: - Security is a key factor in any distributed computing environment and Web services-based environment is not an exception. The Security challenge includes, among other factors, registry and services access-control. In this paper we propose an access control approach that uses uniquely renamed per client service operation names and role based access control, which unifies registry and Web services access control mechanisms. The proposed approach is applicable in the wired as well as the wireless environments, and can support a wide spectrum of enterprises.

Key-Words: - Web services, registry, access control, role-based

1 Introduction

Despite the fact that Web services can enable new business models, securing these services is a prerequisite to wide acceptance of these new models.

Securing Web services can be divided into three inter-related categories: registry security, transaction security, and infrastructure linkage [1]. These categories involve the various Web services security aspects, namely, authentication, authorization, confidentiality, integrity, and non-repudiation. These aspects can be briefly defined as following: First, authentication is the process of identifying the client of a restricted resource or service. Second, authorization is the process of verifying the client's access rights to a particular resource or service. Third, confidentiality in the registry security context is allowing particular service requesters to access the contents of the registry. Fourth, integrity as mentioned by [2] is ensuring that the data has not been tampered with in transit or storage. Finally, non-repudiation means that business transactions are verifiable; which is particularly important for business partners to trust each other.

Registry security, as indicated earlier, is one of the three important parts in securing Web services and is mainly about the registry contents trustworthiness. In other words, the contents are authentic, authoritative, unmodified, current, and confidential as can be the case with private registries.

In this paper we focus on the authorization and confidentiality aspects of the registry security, simply who can do what, when, and how, through enforcing predefined security roles for the service clients. Despite that proposed approach is registry-oriented, it also provides authentication and authorization for service invocation.

The remainder of this paper is structured as follows: Section 2 briefly highlights the motivations behind the proposed approach. Section 3 reviews the related work and technologies that are relevant to Web services security and role-based access control. Section 4 presents a suggested architecture for the proposed approach realization. Sections 5, 6,7, and 8 discuss the various common scenarios. Finally section 9 summaries our proposed approach, and gives the conclusion.

2 Motivation

Among several motivations behind proposing a new approach for Web services access control are the limitations and performance considerations of the current approaches.

First, the current approaches of Web services access control [4,5,9] are considered to operate almost independently of the services registry access control. In fact this could be a source of confusion for some service clients when they successfully discover certain services at the registry then fail to be able to invoke some of these services because of the access control policy of the service provider. Hence, there is a need for integrating the access control of the registry and services.

Second, having an extra "SOAP proxy" [4] intermediary, which requires a client authentication followed by authorization processing such as via

Role Based Access Control (RBAC) [7] imposes performance overhead resulting from the expensive SOAP message processing.

Finally, these approaches might not be the ultimate choice in terms of usability and performance for highly demanding environments such as mobile computing environments.

3 Background

In this section, we briefly describe some of the technologies that are relevant to the proposed approach followed by a review of other approaches.

3.1 Related Work

Securing service registry as integral part of the Web services security is one of the hottest research topics in both the academia and industrial sectors. Brose [4] proposes a proxy-based SOAP security gateway as an implementation of the WS-Security specification [10]. That gateway has a client-side SOAP proxy to insert SAML [11] assertions or digital signatures in the outgoing messages and a server-side SOAP proxy to perform the required authentication and authorization by checking these SAML assertions or XML digital signatures contained in the message headers.

In contrast to this proxy-based implementation of the WS-Security specification, a library-based implementation is proposed by Microsoft under the name of Web Service Enhancements [9]. Again, this implementation is SOAP messages-oriented and relies on screening the SOAP messages sent from the client to perform the required authentication and authorization.

Despite the fact that these implementations can forbid unauthorized users from invoking Web services methods at runtime, they are SOAP messages-oriented as mentioned above which might have a performance impact. Moreover, they provide access control for the Web services but not for the Web services registry.

In terms of access control and security policies, extensive research has already been carried out on access control to XML documents and a number of access control models [3,5,8] were proposed.

The Role Based Access Control (RBAC) was first introduced in 1992 by Ferraiolo and Kuhn. With this model, clients are assigned security roles; and resources availability depends on these roles. Security policies are defined in the form of roles, objects, and permissions.

Based on these access control models, OASIS proposed the eXtensible Access Control Markup Language (XACML) specification [6] that defines standard XML-based policy language and access control language. To support role based access control, in February 2004, OASIS published the XACML Profile for Role Based Access Control (RBAC) specification [14].

Adhering to these specifications, we looked at the extended enterprise registry access control in a previous work [12], and we extended this approach to secure Web services registries, including private UDDI registries, as presented in this paper.

3.2 Related Technologies

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.

Furthermore, a typical role based access control model contains five basic elements: users, roles, objects, operations, and permissions. Security administrators using XACML can define these elements as policies.

XACML systems include a Policy Enforcement Point (PEP), a Policy Decision Point (PDP), and a set of security polices stored in a repository. Typically, a request attempting to access a resource is sent to the system's PEP. The PEP sends a new request to the PDP based on the requester's attributes, the requested object, and the operation type. The PDP examines the request, checks the policies, makes a denial or grant decision, and sends the decision back to the PEP. Accordingly, the PEP grants or denies access to the requester.

Our approach to prevent unauthorized users from accessing particular services is to enforce a role based access control on entries inside the services registry, such as a private UDDI registry.

4 Web Services Registry and Service Provider Architecture

In this section we illustrate the proposed unified registry and service access control model.

This model assumes an access control-enabled private service registry and a 'Clients' directory that

keeps records of the assigned roles of the clients at each enterprise as illustrated in Figure 1.

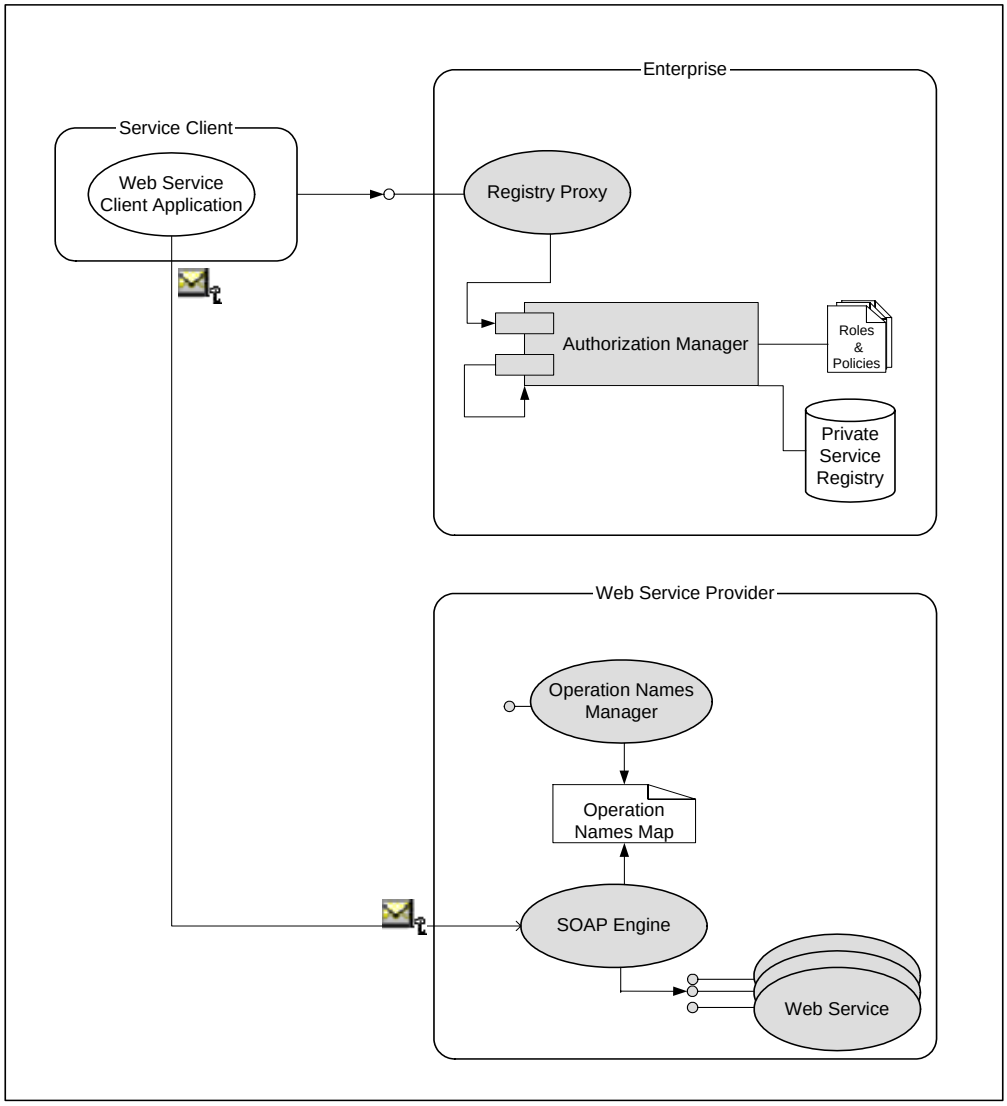

Fig. 1: Web Services Registry and Service Provider Architecture

The model also assumes a predefined set of XACML–encoded security policies that grants access rights to the various security roles. More details about these polices can be found in [12].

It is important to apply XML encryption [13] to protect the data exchanged between the client and the service provider against unauthorized interception.

5 Service Client Registration

Service clients that request to access the private UDDI service registry need to subscribe first with that service registry in order to be granted access rights.

The service client registration activity is illustrated in Figure 2. First, a record for this new client will be created in the 'Clients' directory to associate the clients' credentials to a security role e.g. 'marketing staff'. Next, the security manager finds the details of the authorized services in the service registry and gets their operation names. Next, the security manager replaces the operation original names with uniquely mangled names. Finally, the security manager publishes these mangled operation names to the relevant service providers through their Operation Names Managers, which in turn update the Operation Names Maps.

Upon successful registration and using valid credentials, the client can log on to the system for authentication and accessing the service registry as explained in details in sections 6 and 7.

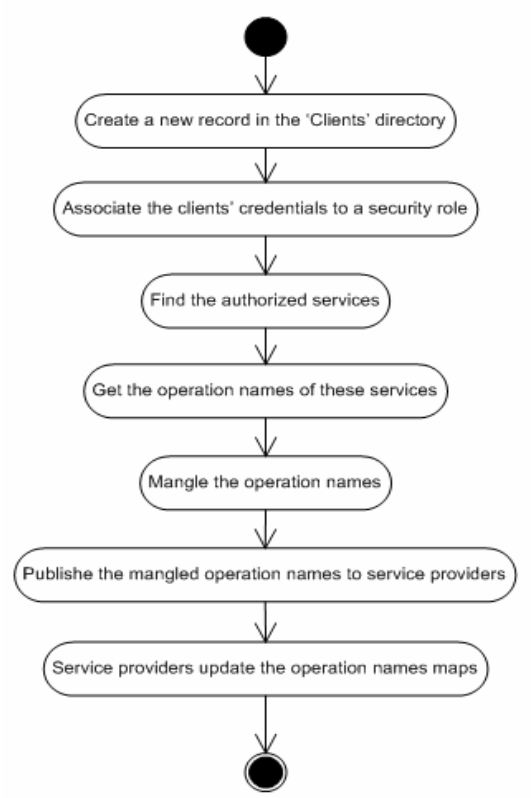

Fig. 2: Service Client Registration

6 Access Control-Enabled Service Discovery

When registered clients authenticate with the service registry Web interface, or query the registry through a client application, they only get a subset of the services based on their particular role. This approach of applying RBAC inside the private service registry has been discussed in details in [12].

The service discovery activity is illustrated in Figure 3. First, the service client sends his credentials for authentication to the service registry. Upon successful authentication, the Registry Proxy asks the Authorization Manager to define the authorized services. Next, the Authorization Manager uses the client credentials to lookup the 'Clients' directory for the security role of that client and looks up the policy set associated with that role to define the authorized services. Next, the security manager finds the details of these services in the service registry. Finally, after replacing the operation original names with the mangled names generated when the client registered with the registry, the Registry Proxy gets the authorized services details and responds to the Web service client.

Obviously, different views of the service registry (i.e. different sets of WSDLs) are presented to different clients based on their particular role; consequently, and as an immediate advantage,

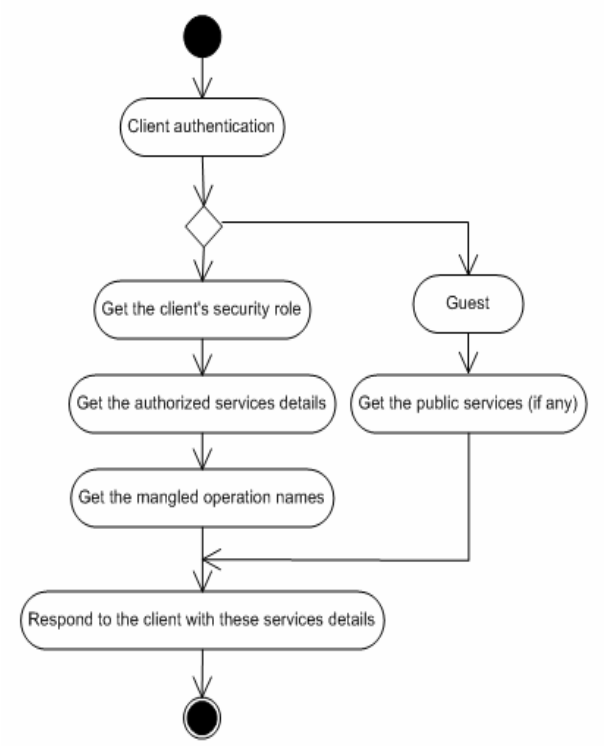

Fig. 3: Service Discovery Activity

clients can only browse and access the permitted set of services. Nevertheless, service providers can allow anonymous clients to get benefit from some public services, when applicable, through defining a 'guest' role for instance and keep the rest of services restricted to the pre-registered clients and based on their specific roles.

Furthermore, hiding the inaccessible services leads to a better user experience and productivity levels, especially for mobile computing environments, where client devices usually have limited resources.

7 Access Control-Enabled Service Invocation

The access control for services is unified with the registry access control.

In general, there are two mechanisms that the client can use for Web services discovery, namely, design time discovery and runtime discovery. Independent of the discovery mechanism, the service details retrieved from the registry will always have the 'real' endpoint URL of the service but with uniquely mangled operation names.

The service invocation activity is illustrated in Figure 4. First, after successful discovery of the authorized services, the client application selects the Web service to access and the operation to perform. Next, using encrypted SOAP messages (possibly

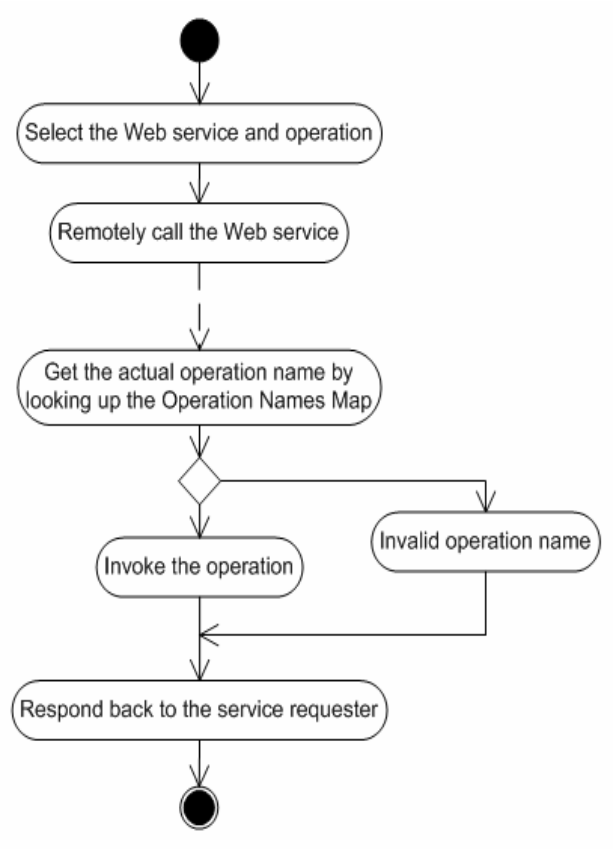

Fig. 4: Service Invocation Activity

with just the operation name encrypted for performance considerations), the client application remotely calls that particular operation in the required service. Next, the SOAP engine at the service provider site, after decrypting the operation name/message, gets the mangled name of the service operation and attempts to look up the Operation Names Map to get the service implementation method name. Once found, the SOAP engine calls that method of the specified service implementation. That is the method on the underlying service implementation/component is invoked by the SOAP engine. Finally, the SOAP engine will respond back to the service requester using encrypted message as well.

If an unauthorized client attempts to call a service, it will not know a valid mangled operation name and hence, service access will be denied. That is, there will not be an entry in the Operations Names Map for the invalid mangled operation name.

8 Service Client Deregistration

When service clients are deregistered for any reason, they shall no longer be able to access the registry or invoke the services they previously discovered.

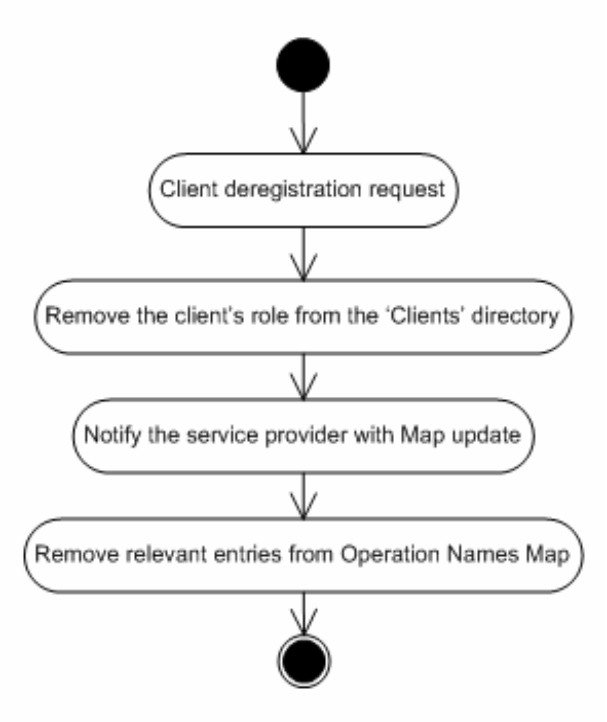

Fig. 5: Service Client Deregistration Activity

As illustrated in Figure 5, to deregister a client we remove the role assigned to that client from the 'Clients' directory, consequently, any further requests from that client to access the service registry will not be successful. However, because former clients already know the access points and operation names of the permitted services in the registry and may be able to bypass the authentication process and access these services, the UDDI registry pushes map update to Operation Names Manager at the service provider site to remove the entries associated with this client. Hence, when a SOAP request comes, the SOAP engine looks up the Operation Names Map as normal, however, it fails as the mangled operation name no longer exists in the map, and hence the SOAP engine blocks this no longer authorized call.

9 Authorized User and Network Sniffing

Another possible threat arises when someone attempts to sniff the network during the service access session of an authorized client. This threat is eliminated by encrypting the SOAP messages (or at least the operation name part) exchanged between the service client and service provider.

10 Conclusion

Securing and controlling the access to Web services has its own challenges, and having a SOAP gateway checking every incoming SOAP message for access permissions has performance and responsiveness disadvantages. In this paper we have presented an access control approach for securing registry and Web services based on unique per client service operation names and role based access control enforcement.

References:

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

[2] Agrawal, M., Padmanabhan, H., Rao, H.R. & Upadhyaya, S., A Conceptual Approach to Information Security in Financial Account Aggregation, *Proceedings of the 6th international conference on Electronic commerce*, 2004, pp. 619-626.

[3] Bertino, E., Castano, S. & Ferrari, E., Securing XML documents with Author-X, *IEEE Internet Computing*, Vol.5, No.3, 2001, pp. 21-31.

[4] Brose, G., Securing Web Services with SOAP Security Proxies, *Proceedings of the ICWS 03 International Conference on Web Services*, 2003, pp. 231-234.

[5] Damiani, E., De Capitani Di Vimercati, S., Paraboschi, S. & Samarati, P., A fine-grained access control system for XML documents, *ACM Transactions on Information & Systems Security*, Vol.5, No.2, 2002, pp. 169-202.

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

[7] Ferraiolo, D. & Kuhn, R., Role Based Access Control, *15th National Computer Security Conference*, 1992.

[8] Gabillon, A. & Bruno, E., Regulating access to XML documents, Database and Application Security XV, *IFIP TC11/WG11.3 Fifteenth Annual Working Conference on Database and Application Security*, 2002, pp.299-314.

[9] Microsoft, Web services enhancements, [online] http://msdn.microsoft.com/webservices/building/wse/default.aspx

[10] OASIS 2003, Web services security: SOAP message security. Working Draft 10, [online] http://www.oasis-open.org/committees/wss/documents/WSS-SOAPMessageSecurity-10-0223-merged.pdf

[11] OASIS 2003, Assertions and protocol for the OASIS Security Assertion Markup Language (SAML). Committee Specification 01, [online] http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf

[12] Steele, R. & Dai, J., UDDI Access Control for the Extended Enterprise, *Accepted to the International Conference on Web Information Systems and Technologies*, 2005

[13] W3C 2002, XML Encryption Syntax and Processing, W3C Recommendation, [online] http://www.w3.org/TR/xmlenc-core/

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