Security Considerations for SOA-based Multimedia Applications

Andrew Roczniak, Alexandre Miège and Abdulmotaleb El Saddik
Multimedia Communications Research Laboratory
University of Ottawa, Ottawa, Canada
Email: {roczniak, amiege, abed}@mcrlab.uottawa.ca

Abstract

Growing levels of digitalization and broadband access drives extremely fast progress in multimedia and networking technologies and allows consumers to create requirements at an accelerating rate. Producers' response is to emphasize the speed of delivery and upgradability of applications. Development of Web Services and Service-Oriented Architecture puts the emphasis on creation of specific services which then can be aggregated to achieve a particular goal. By substituting or adding new services, a particular application can be adapted faster to changing requirements. Any application thus composed must address security concerns, including security guarantees after a service substitution. Based on our framework for creating multimedia collaborative authoring applications from a set of standard services, we provide a security analysis of the resulting application and present a novel framework for ensuring uniform access control guarantees across different constituent services.

1 Introduction

The ubiquity of computers, networks and multimedia objects means that we are increasingly interacting with multimedia documents. Reflecting that fact, some business processes may include the need to collaboratively author a multimedia document, motivating in turn the development of suitable tools. Business processes are sometimes shaped by the available application, or equally probably, define the requirements on an application. This is usually a dynamic relationship where both the business processes and the applications supporting it, evolve. Applications are generally built on top of frameworks, and great care has to be deployed in order to make those frameworks responsive enough to inevitable changes. Frameworks in turn can be thought of as a collection of various services. Service-oriented architectures (SOA)[17] promises greater adaptability of business processes to new and evolving business realities. Conceptually, we can represent it as applications being built on top of services (and not frameworks), or alternatively as reversing the relationship between services and frameworks - services are now built on top of frameworks - allowing for specialized and highly-tuned services to do the work required by an application. Figure 1 illustrates the fact that services provided by frameworks can be supported by various network architectures.

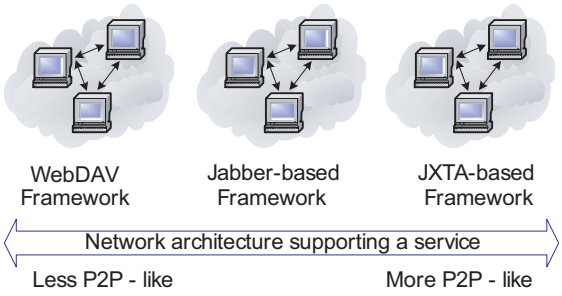

Figure 1. Each framework provides some native services. Frameworks can be implemented using various network architectures, including client/server and P2P

We have previously investigated the possibility of empowering everyday users of the Internet to quickly assemble their own collaborative authoring application from a set of standard services [18, 19]. In this tool, users provide their own content as needed, and that content does not necessarily survive the session: there is no central repository of documents that can be accessed at any time. As our resulting aggregation framework did not address security issues, in the following we provide answers to these two questions:

1. What kind of security with respect to authentication, confidentiality and integrity is offered in the current implementation?

2. Since services can be substituted for each other, this

presents a challenge for portability of consistent security policy. What kind of framework can be implemented to ensure that an application composed from services presents uniform security guarantees to its users across services?

This paper is structured as follows. Section 2 describes what is a service in the context of SOA and motivates the usage of Jabber and JXTA protocols. Section 3 presents the design and an example of a multimedia collaborative authoring application; Section 4 presents the security analysis of the application; Sections 5 outlines the approach taken to design and implement a security framework whose goal is to ensure that a certain security level is maintained across services; Related work is discussed in 6 and the paper concludes with Section 7.

2 Services and Protocols

A *service* is some functionality that is packaged and offered in a specific way. Services are self contained and operate as black boxes; all necessary components to accomplish a specific functionality are encapsulated within the service and consumers are not concerned with the underlying implementation. Those characteristics should make a service independent of consumer context, allowing reuse in contexts not known at design time. Services expose their functionality through interfaces. The requirement on those interfaces is that they be *invokable* in a standardized way. The service itself may be within the same application or in a completely different system on the Internet but the interconnect scheme or protocol used to allow the invocation should however be open and standardized. This in turn requires open and well-understood way of publishing, finding and binding to services. The infrastructure components required to make the connection can be diverse since services may be implemented on a single machine or distributed across a set of computers on a network [17]. Services can be composed into other services - based on quantifiable quality of service, for example - and offered transparently to the consumer, as one service.

Traditionally, SOA implementation relies on the Web Services (WS) family of protocols to support interface definition and binding (WSDL), message exchange (SOAP), service discovery (UDDI) and composition techniques as presented in [9]. We have elected to use the Jabber and JXTA set of protocols instead of the WS protocols due to their relative simplicity and their closer focus on the functionality we are developing.

Jabber [22] is a set of streaming XML protocols that enable participants to exchange structured information in close to real time. Two of those protocols in particular are the extensible messaging and presence protocol (XMPP) defining messaging, presence, and request-response services between any two network endpoints, and instant messaging and presence protocol (XMPP-IM) defining instant messaging and presence functionality. These protocols usually depend on TCP/IP and are specified by the Internet Engineering Task Force (IETF). A Jabber framework usually follows a client/server architecture where each client registers with a Jabber server.

JXTA is a set of protocols that specify basic functionality required by peer-to-peer type applications. Functions such as peer discovery, peer communication and their associated management are performed by publishing and exchanging XML advertisements and messages between peers. JXTA architecture provides a standard service discovery framework, allowing 3rd party services to be offered. An example of such a service is the JXTA Content Management System (CMS) which is used to manage and transfer shared files.

To verify that the usage of Jabber (JXTA inherently uses the *publish*, *find* and *bind* paradigm) protocols is consistent with the general understanding of SOA, we are relying on compliance to the Reference Model for Service-Oriented Architecture as specified by OASIS [15].

As will be described in later sections, we are interested in three services produced by a Jabber server and accessed through the Jabber set of protocols. We make the basic assumptions that the server is willing to engage in service interaction and that there is a working communication path between the producer and consumer. The first two services are the capability to deliver a message to either a specific participant or to a group of participants. The implementation of those services hides the necessary naming, addressing, binding and routing functionalities. The third service is the capability to notify a participant about the availability of another participant. In contrast to the message delivery service to a specific participant, the group message delivery service may not be available. The user is responsible for obtaining information about the availability and type of this service directly from the server (service awareness). The Jabber set of protocols also define the valid input and output as well as error messages used in the process of interaction.

3 Aggregation Framework

At its basis, a collaborative authoring tool has at the minimum three main requirements: bulk transfer (to share files), group notification (to notify other users about one's actions) and document consistency (to ensure that the shared objects are consistent between users. Service State Support provides a facility to keep track of the interaction with the service, and may be needed in some cases. Service Interaction Interface is necessary to be able to find, invoke and interact with a given service. There may be numerous comparable services, which may be selected depending on

certain circumstances. Each service may be modeled as a resource that needs to have a notion of access control, at the minimum to differentiate between different sessions (we assume that service providers are not required to keep the state of the session). This conceptual view is shown in Figure 2.

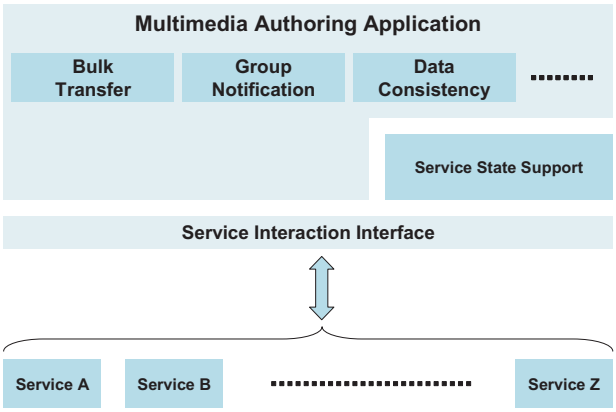

Figure 2. Conceptual view of fulfilling application's requirements by service composition

The problem thus is reduced to selecting technologies that fulfill these three main requirements while being consistent with SOA philosophy. The *jabber.org* server (Jabber server) provides various services accessed through the Jabber set of protocols. There are three services that we will be using: a Jabber client can send short XML-based messages to any other connected client, Jabber servers usually provide chat rooms supporting multiple participants where any message issued to the chat room will be forwarded to all participants that joined it (a chat room can thus be viewed as a multicast group abstraction), and finally, since each Jabber client needs to establish a connection with a server, this *presence* information can be made available (depending on security and privacy policies in effect) to other clients. Note that we are not using certain functionalities such as chat room playback and potential server-side components as they are defined and controlled by the Jabber server administrator and are thus not standardized. We also use the JXTA set of peer-to-peer protocols to publish service advertisements at a Rendez-Vous (RV) peer, and to establish a direct connection between any two participants. Services are peer-based (RV acts as a broker), and we are mainly interested in the ability to exchange large amount of data between any two peers.

The group notification requirement can be met by Jabber's ability to send messages to individual participants as well as to the whole group, but we still need to address the bulk transfer and ownership management requirements. By design, Jabber servers do not handle transfers of large amount of data between any two peers in order to provide efficient and fair routing for all users, hence, we have implemented a HTTP-based transfer using a Web server and a more efficient solution for larger files, a JXTA-based *bulk transfer protocol* that distributes the cost of uploading between all participants (Extended CMS). The final requirement is met by leveraging the existing group notification service in conjunction with the presence service to implement a Jabber-based *locking protocol* to distribute ownership management over all participating users. This design view is illustrated in Figure 3.

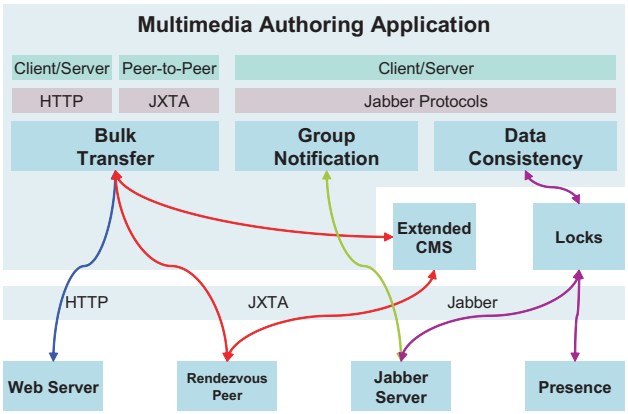

Figure 3. Design view of the multimedia authoring application

Figure 3 highlights the fact that the security of the whole application is highly dependent on the security features provided by the underlying technologies. From a wider perspective, and as shown in Figure 2, there are many services to choose from, each potentially implemented on different frameworks. The situation is further complicated by the fact that services can be substituted for each other. At a minimum, an application composed from services should be aware of the security guarantees it currently can offer to the users, but better yet, it would interact with service providers in order to establish a certain level of consistent security guarantees irrespective of which service it currently depends on. Those two issues are discussed in the following sections.

4 Security Analysis

In this section, we carry out a brief risk analysis in order to highlight the security issues and present potential solutions. The objective of this framework is to enable easily deployable and upgradeable multimedia collaborative au-

thoring application and as a consequence, any security solution should be as transparent to the users as possible.

4.1 Risk Analysis

We envision that the context in which our application is to be used is consistent with the following assumptions. First, we assume that the participants know each other; that there is a limit on the number of users; and that they are both reliable and genuinely willing to cooperate. Second, the system is not designed to support collaboration on highly confidential information, however access to shared objects should be access-controlled. Third, we assume that Jabber messages are always received within a finite amount of time. Threats can be divided into accidental events and malicious actions. In our context, accidental events are for example problems related to the network state and users' computers and as such are out of our scope. The following malicious actions are identified: (1) eavesdropping on Jabber communication, (2) eavesdropping on file transfer service, (3) illegal access to the system that stores the shared objects and (4) illegal access to the chat room which would enable the attacker to disturb the service by sending misleading messages. The sensitive data and information are: (a) the Jabber messages used for data consistency maintenance (integrity), (b) shared objects should not be modified outside of the valid execution of the protocol (integrity), and (c) the data transfer corresponding to the upload/download actions of the shared objects (confidentiality).

The framework thus is open to the following risks: (i) an attacker could issue Jabber messages to some users in order to compromise the data consistency maintenance, (ii) an attacker could enter the chat room and transmit misleading messages with the same objective as before, (iii) an attacker gains access to the server that stores the shared objects and succeeds in downloading or modifying them, (iv) an attacker eavesdrops on the upload or download of a shared object.

If a Web server is used to ensure the bulk transfer (see Figure 3), the solution addressing risks (iii) and (iv) is well-known and is composed of HTTPS or SHTTP to avoid eavesdropping during data transfer, and a login/password mechanism to control the access to the server. In the following sections we discuss the security features provided by Jabber and JXTA that can be leveraged in order to protect the framework from the identified risks.

4.2 Jabber Security

The XMPP Core protocol used by Jabber specifies how to achieve both communication confidentiality and mutual authentication between a client and a server and between two servers. Confidentiality is provided with Transport Security Layer (TSL) encryption while peer's authentication is achieved through the use of the Simple Authentication and Security Layer mechanism (SASL). If TSL and SASL are enforced, all Jabber packets are encrypted and thus it cancels the risk (i) identified in section 4.1. These two protocols can easily be enforced in our application transparently to the users, however TSL and SASL are optional for servers. This implies that users should either find or install and manage their own TSL and SASL enabled server. An end-to-end encryption for confidentiality and integrity independent of the servers is discussed in [10], but no implementation has been proposed. Jabber protocols support creating password-protected or members-only chat rooms. In both cases one of the members is in charge of creating a room. In the first case, this user will transmit the password to other participants, while in the second case a member *whitelist* has to be communicated to the Jabber server. Assuming SASL usage for user authentication, the access control over the room enables protection against risk (ii).

4.3 JXTA Security

We focus on two security mechanisms provided by JXTA: the peer group services and the TSL transport binding. One of the objectives of the creation of peer groups is to implement secured environment in which confidential documents can be shared. Furthermore a specific *Membership service* is used to grant or revoke access to members. JXTA implements TSL and so is able to provide confidentiality between peers. These two mechanisms address risks (iii) and (iv) identified in section 4.1.

4.4 Discussion

We showed in this section that some simple built-in security mechanisms can be leveraged in order to greatly reduce the risks associated with the infrastructure. Since most of the security mechanisms are transparent for the members and do not generate any significant overhead, the collaboration remains simple from the users' point of view. However, various services can be used for the same purpose with the consequence that security requirement must be expressed in a generic way. In the next section, we suggest an original way to handle this issue by defining the security requirements using a formal framework.

5 Service Security Framework

We propose to use a formal access control model in order to allow a client to specify the policies and security requirements which a producer agrees to implement within the context of a contractual agreement for a specific service. We first present an overview of the Or-BAC model and then

show how it can be applied in the context of the multimedia authoring application.

5.1 Or-BAC Model

Or-BAC [1] was designed to provide a robust modeling solution to address access, network and usage control policies in traditional IT systems. Based on the flexible concept of organization, Or-BAC is suitable for collaborative work between entities from different organizations. Furthermore, Or-BAC allows the specification of contextual policies: specification of security rules can depend on concrete circumstances such as for example time, location and previous actions carried out by users [7].

Traditional access control policies consist of a set of triples $< user,\ action,\ object >$ that define which user is allowed to perform what action on which object. Since it is rather intuitive to consider that in an organization users play some roles (developer, project leader, etc.) role-based models specify privileges associated with each role, making management and administration of a security policy much easier. In Or-BAC this concept is defined through a ternary relation written in a first order language: $Empower(org, s, r)$, meaning that the organization org empowers user s in role r.

While traditional access control models focus on modeling users and role only, Or-BAC also provides an abstract view of actions and objects using the *Activity* and *View* entities. The following two predicates are defined by analogy with the predicate *Empower*: $Consider(org, \alpha, a)$, meaning that org considers that action α falls within the activity a, and $Use(org, o, v)$, meaning that org uses object o in view v. The Or-BAC core model is complete with the additional two predicates: $Permission(org, r, a, v)$, meaning that org grants role r the authorization to perform activity a on view v, and $Is\_permitted(s, \alpha, o)$, meaning that a subject s is permitted to perform an action α on object o. *Permission* corresponds to the organizational policy and is specified by the security designer, while *Is_permitted* corresponds to the concrete policy implemented and enforced on a service. As a consequence, Or-BAC models a two-level security policy where the concrete policy is derived from the organizational policy using the following logical rule:

$\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall r \in \mathcal{R}, \forall a \in \mathcal{A}, \forall v \in \mathcal{V},$

$$\begin{aligned} &Permission(org, r, v, a, c) \wedge \\ &Emplower(org, s, r) \wedge \\ &Consider(org, \alpha, a) \wedge \\ &Use(org, o, v) \wedge \\ &\rightarrow Is\_permitted(s, \alpha, o) \end{aligned}$$

The Or-BAC model also provides means to specify hierarchies of organizations, as described in [6]. This has two objectives. The first one is to model an organization structure through an organization hierarchy. Most companies and institutions are composed of departments, units, services, etc. All those entities are considered as sub-organizations in the Or-BAC model. The Or-BAC model combines inheritance mechanisms with the organization hierarchy. This enables us to define the general policy of an organization. Then, all sub-organizations inherit this policy while each of them has the possibility to add new rules and thus specify a sharper and more accurate policy. The second objective is to ease the collaboration inter-organizations. This will be useful in the context of Internet-based service composition. Actually, in the specific case of a collaboration, the group composed of those organizations can be seen as a meta organization, in such a way that it turns out to be the same issue as the modelling of a single organization.

The Or-BAC model is build on Datalog and therefore any Or-BAC policy has a unique solution and is tractable in polynomial time. The flexibility of such a model is the result of combination of the specification of security policy (set of *Permission*) and the description of the structure of the organization (*Empower*, *Use* and *Consider*). In the following section we examine how to use the resulting separation of generic security policy on the one hand and a concrete security policy one the other hand within our framework.

5.2 Uniform Security

In the following we discuss how the Or-BAC model can be applied in the context of Section 3 to provide uniform security across different services. With reference to Figure 3, the entity providing the service (Web Server or JXTA RV peer) needs to know who can do what to which resource. For example, the Web server needs to be aware of who is allowed to add, delete, download or upload a multimedia file. In the case of service based on JXTA, the RV peer needs to be aware that certain users will be creating and consuming specific content advertisements. Additionally, the users themselves need to know which users are allowed to obtain and provide state information and file chunks [19]. In other words, the concrete security policy is different from one service to another even though it remains the same from clients' point of view. Ideally, clients should have uniform security guarantees, irrespective of the underlying service.

In our proposed security framework clients express their policy, based on some rules and organization structure, and service producers derive the corresponding concrete policy as seen in Figure 4. For a group of students at the University of Ottawa for example, a security policy written

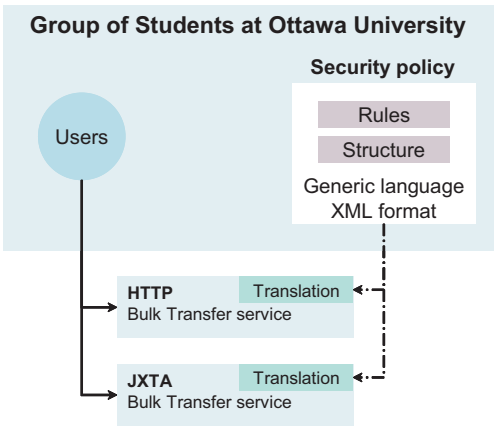

Figure 4. Uniform Security Framework

using the Or-BAC model can take the form:

• Rules:

$Permission(OttawaU, Student, Share, Videofile)$
$Permission(OttawaU, Professor, Admin, Videofile)$

• Structure description:

$Empower(OttawaU, John, Student)$
$Empower(OttawaU, Marie, Professor)$
$Consider(OttawaU, Send, Share)$
$Consider(OttawaU, Get, Share)$
$Consider(OttawaU, Delete, Admin)$
$Consider(OttawaU, AddNew, Admin)$
$Use(OttawaU, *.avi, Videofile)$

The producer will derive the concrete policy using the rule defined in section 5.1 and provide its own implementation. In the case of the Web server, this is expressed in Or-BAC as:

$Is\_permitted(John, Put, video1.avi)$
$Is\_permitted(John, Get, video2.avi)$
$Is\_permitted(Marie, Delete, video3.avi)$

The outlined method enables clients to specify generic security policies without customizing it to the underlying service. Furthermore, it lends itself to formal constraint and completeness verification, and can be rapidly extended to other non-functional tasks such as context modeling, multi-organization management, conflict management and network adaptation.

6 Related Work

The motivation for addressing security issues in SOA-based applications is presented in [11]. A general view on security is presented in [21] which investigates security issues from both the network and application layer perspective in peer-to-peer networks, and [5] which discusses security requirements in service-oriented architectures as applied to ubiquitous computing.

Applications built using Web Services protocols would naturally gravitate toward WS-Security specification (SOAP Message Security) which provides mechanisms for securing exchanges of SOAP messages [16] and emerging specification such as WS-Trust in order to provide a framework for security tokens and to broker trust relationships between participants.

The Web Services Security Policy Language (WS-SecurityPolicy) [13] is based on WS-Policy and enables the specification of security constraints such as for example restricting Web Service access to only SOAP messages signed with X.509 certificates. In [4] authors rightly argue that the policies specified using WS-SecurityPolicy correspond to low level security enforcement choices, and that a higher level policy language should be defined. They suggest a new language based on a formal semantics and propose associated tools allowing them to prove the correctness of the policies. The type of security policies addressed focus on authorization data, certificates and password management, etc., which are necessary but which are not derived from higher level access control policies; and lower level security solutions should preferably be inferred from the authorizations given to the users.

Various publications address SOA access control issues. Design goals and a distributed access architecture for a general Web Services access control are presented in [14]. This proposal differs from ours since it calls for a parallel deployment of specialized Web Services responsible for filtering and processing access requests. Dynamic reconfiguration of a Web Services depending on chosen policies is investigated in [2], which addresses the preservation of interoperability for non-functional tasks such as security. The proposed solution involves a specialized programming technique based on Aspects Oriented Programming, allowing clean separation of common issues.

Numerous formal access control models have been proposed to improve and adapt the traditional mandatory (MAC) [3] and discretionary (DAC) [8] model to the IT development. Among them are RBAC (role-based) [12], TBAC (task-based) [20] and Or-BAC (organization-based) [1]. These models bring new abstractions and concepts that enable to specify high-level policy in complex and large scale IT systems. Extensive adoption of Web Services will certainly lead to the need for such formal models. An At-

tribute Based Access Control (ABAC) model is proposed in [23], which is based on subject, object, and environment attributes and supports both mandatory and discretionary access control needs. The proposed framework relies on SOAP and XACML to exchange access control data, and focuses on modeling access control to individual Web Services. In contrast, our research focuses on translating and applying one access control policy to any provider of a required service.

7 Conclusion

A certain level of security in collaborative applications is paramount to ensure its adoption by users. Our collaborative application is based on a service-oriented architecture, and as such relies on security features present in the technologies used to implement the service itself. A more interesting challenge is is to specify the access control policies independently of the underlying service. An infrastructure that offers such a uniform policy will enable service consumers to change service providers without having to renegotiate access control policies. We propose to use the formal Or-BAC model to specify abstract policies and translate them, as needed, to adapt to the specific service provider. Future work will concentrate on defining exact access control policies and integration with our service aggregation framework. The following step will be, based on this idea, to propose a more general and formal framework that applies to generic service oriented applications and generic service composition using, among other things, the ability of the Or-BAC model to manage several organizations.

References

[1] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03)*, June 2003.

[2] F. Baligand and V. Monfort. A concrete solution for web services adaptability using policies and aspects. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 134–142, New York, NY, USA, 2004. ACM Press.

[3] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR-2997, Rev. 1, MITRE Corporation, Bedfort, MA, March 1976.

[4] K. Bhargavan, C. Fournet, and A. D. Gordon. Verifying policy-based security for web services. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 268–277, New York, NY, USA, 2004. ACM Press.

[5] D. Cotroneo, A. Graziano, and S. Russo. Security requirements in service oriented architectures for ubiquitous computing. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 172–177. ACM Press, 2004.

[6] F. Cuppens, N. Cuppens-Boulahia, and A. Miège. Inheritance hierarchies in the Or-BAC Model and application in a network environment. In *Proceedings of the Foundations of Computer Security (FCS'04)*, Turku, Finland, July 2004.

[7] F. Cuppens and A. Miège. Modelling contexts in the Or-BAC model. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, pages 416–427, Las Vegas, Nevada, USA, December 2003.

[8] DAC. A Guide to Understanding Discretionary Access Control in Trusted Systems, 1987.

[9] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.

[10] End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP). *http://www.ietf.org/rfc/rfc3923.txt*, 2004.

[11] J. Epstein, S. Matsumoto, and G. McGraw. Software security and soa: danger, will robinson! *IEEE Security and Privacy Magazine*, 4(1):80–83, Jan.-Feb. 2006.

[12] D. F. Ferraiolo and R. Kuhn. Role-Based Access Controls. In Z. Ruthberg and W. Polk, editors, *Proceedings of the 15th NIST-NSA National Computer Security Conference*, pages 554–563, Baltimore, MD, 13-16 October 1992.

[13] http://www.w3.org/TR/wsdl.

[14] R. Kraft. Designing a distributed access control processor for network services on the web. In *XMLSEC '02: Proceedings of the 2002 ACM workshop on XML security*, pages 36–52, New York, NY, USA, 2002. ACM Press.

[15] OASIS Reference Model for Service Oriented Architecture, Committee Draft 1.0. February 2006.

[16] OASIS Standards - Web Services Security v1.1. 2006.

[17] M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *International Conference on Web Information Systems Engineering (WISE '03)*, pages 3–12, 2003.

[18] A. Roczniak and A. El-Saddik. Jade: jabber-based authoring in distributed environments. In *ACM international conference on Multimedia (ACM-MM '05)*, pages 279–282. ACM Press, 2005.

[19] A. Roczniak, J. Melhem, P. Lévy, and A. El-Saddik. Design of distributed collaborative application through service aggregation. In *International Symposium on Distributed Simulation and Real Time Applications (DS-RT '06)*, Malaga, Spain, October 2006.

[20] R. Thomas and R. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. In *Proceedings of the 11th IFIP Working Conference on Database Security*, Lake Tahoe, California, USA, August 1997.

[21] D. S. Wallach. A survey of peer-to-peer security issues. In *Proceedings of the International Symposium on Software Security*, Tokyo, Japan, November 2002.

[22] www.jabber.org.

[23] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services (ICWS'05)*, pages 561–569, Washington, DC, USA, 2005. IEEE Computer Society.