Access Control for Collaborative Systems: A Web Services Based Approach

Anas Abou El Kalam, Yves Deswarte, Amine Baïna, Mohamed Kaâniche

Access Control for Collaborative Systems: A Web Services Based Approach

Anas Abou El Kalam
ENSI de Bourges-LIFO
anas.abouelkalam@ensi-bourges.fr

Yves Deswarte Amine Baïna Mohamed Kaâniche
LAAS-CNRS, Université de Toulouse
firstname.familyname@laas.fr

Abstract

Nowadays, systems are more and more open, distributed and collaborative. In this context, access control is an important issue that should be studied, specified and well enforced. This work proposes a new access control model for collaborative systems: "PolyOrBAC". On the one hand, we extend OrBAC (Organization-Based Access Control Model) to specify local as well as collaboration access control rules; on the other hand, we enforce these security policies by applying web services mechanisms (XML, SOAP, UDDI and WSDL). We thus present a representative scenario of secure collaborative applications. Furthermore, we propose a XACML-based implementation of PolyOrBAC and we discuss the most important approaches that emphasize access control in collaborative environments.

1. Introduction

Collaboration can be defined as a structure of interactions designed to facilitate the accomplishment of a specific goal (e.g., product) through people working together (and responsible for their actions). While many applications (industry, government) require efficient processing, the exchange and sharing of large amounts of data and services, as well as the openness of collaborative systems generate one of the most important problems in computer science: how to improve security without compromising the other functionalities of the system? Economic and strategic stakes − in particular those related to security − have become more and more important.

In this context, a well-founded security study should first begin by identifying who has access to what, when and in which conditions? This is what we commonly call an "access control policy". The latter is globally defined in the common criteria as the set of laws, rules and practices that regulate how an organization manages, protects, and distributes sensitive information [1].

Nevertheless, the access control policy can be badly designed and intentionally or accidentally violated. An access control model is used to rigorously specify and reason on the access control policy. It is thus intended to abstract the policy and to handle its complexity; to verify its consistency; and to detect and resolve possible conflicts, in particular when new users or sub-systems join the collaborative system. However, an access control model does not specify how the security policy is implemented. To achieve this aim, we use technical security mechanisms such as credentials, cryptographic transformations (e.g., signature, encryption), access control lists (ACL), firewall rules.

Our major aim in this paper is to define a global framework (model and mechanisms) for secure collaborative systems while respecting their functioning flexibilities. More precisely, we suggest using OrBAC as well as Web Services (WS) mechanisms to specify and enforce collaboration between organizations.

Concerning the model, as OrBAC (Organization-Based Access Control) expresses security rules only through abstract entities (organizations, roles, views, activities and contexts), it can specify a large range of security policies while respecting the local functioning of each organization. Moreover, the OrBAC formal system (based on first-order logic) is useful for specifying and reasoning about permissions, prohibitions and obligations [2].

However, OrBAC is not intended to manage "secure" interactions between the collaborating subsystems. To achieve this aim, web services technology provides several mechanisms and standards that could be interesting in our work.

The remainder of this paper is organized as follows: Section 2 presents the necessary background to understand our work. Then, in Section 3 a new access control framework −called PolyOrBAC− for specifying and enforcing security in collaborative systems is presented. Afterwards, Section 4 discusses some important existing strategies used to secure collaborative systems. Finally, in Section 5, we draw out conclusions and perspectives.

2. Background

2.1. Traditional access control models

Classical access control policies and models (discretionary "DAC" and mandatory access control "MAC" [3]) are not really adapted to collaborative systems. For instance, HRU represents the relationships between the subjects, the objects and the actions by a matrix M [4]. *M(s, o)* represents the "action" that a subject s is allowed to carry out on an object o. It is thus necessary to enumerate all the triples (s, o, a) that correspond to permissions defined by the security policy. Moreover, when new subjects, objects or actions are added to or removed from the system, it is necessary to update the policy. Role Based-Access Control (RBAC) is more flexible. Roles are assigned to users, permissions are assigned to roles and users acquire permissions by playing roles [5, 6]. Hierarchical RBAC [7] adds a requirement for supporting the role hierarchies, while constrained RBAC [8] enforces the separation of duties. RBAC is unquestionably suitable for a large range of organizations. Indeed, if users are added to the system, only the instances of the relationship between the users and the roles are updated.

The OrBAC (Organization-based Access Control) model is an extension of RBAC that details permissions while remaining implementation independent. The main idea is to express the security policy with abstract entities only, and thus to completely separate the representation of the security policy from its implementation. Indeed, OrBAC is based on roles, views, activities (introduced in RBAC, VBAC [21], TBAC [9, 10]) to structure subjects, objects and actions.

In the next section, we first summarize OrBAC (Organization-based access control) and we discuss the limits of this model. Then, we present the most relevant Web Services mechanisms for our study. Finally, in Section 3, to overcome the OrBAC limits and to cover the particularities of collaborative systems, we couple an extension of OrBAC with web services mechanisms. The result is called PolyOrBAC.

2.2. OrBAC (Organization-based access control)

In OrBAC, an organization is a structured group of active entities, in which subjects play specific roles. An activity is a group of one or more actions, a view is a group of one or more objects, and a context is a specific situation that conditions the validity of a rule.

Actually, the Role entity is used to structure the link between the subjects and the organizations (Figure 1). The relationship Empower (org, r, s) means that org employs subject s in role r. In the same way, the objects that satisfy a common property are specified through views (Figure 2), and activities are used to abstract actions (Figure 3).

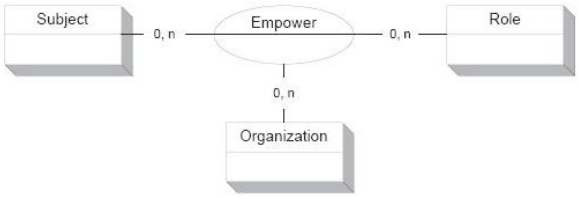

Figure 1: Abstracting subjects.

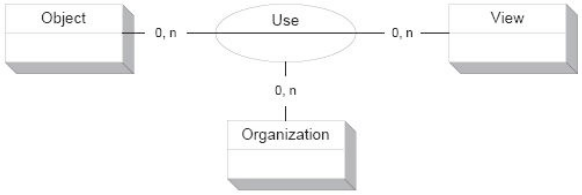

Figure 2: Abstracting objects.

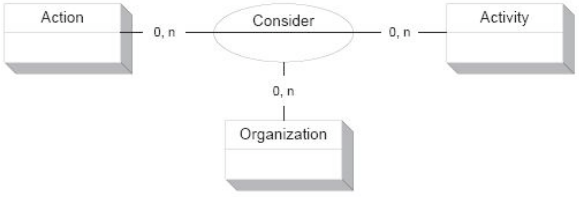

Figure 3: Abstracting actions.

Security rules have the following form: Permission (org; r; v; a; c), Obligation (org; r; v; a; c), and Prohibition (org; r; v; a; c). In the context "c", organization "org" grants role "r" the permission (or the obligation or the prohibition) to perform activity "a" on view "v".

As rules are expressed only through abstract entities, OrBAC is able to specify the security policies of several collaborating and heterogeneous organizations.

In fact, the same role e.g., "operator" can be played by several users belonging to different organizations; the same view e.g., "TechnicalFile" can designate TF-Table or TF1.xml (according to the organization); and the same activity "read" could correspond in a

particular organization to a "SELECT" action (if the organization has a database system) while in another organization it may specify an OpenXMLfile() action.

Two security levels can be distinguished in OrBAC :

-Abstract level: the security administrator defines security rules through abstract entities (roles, activities, views) without worrying about how each organization implements these entities.

-Concrete level: when a user requests an access, concrete authorizations are granted (or not) to him according to the concerned rules, the organization, the played role, the instantiated view / activity, and the current parameters.

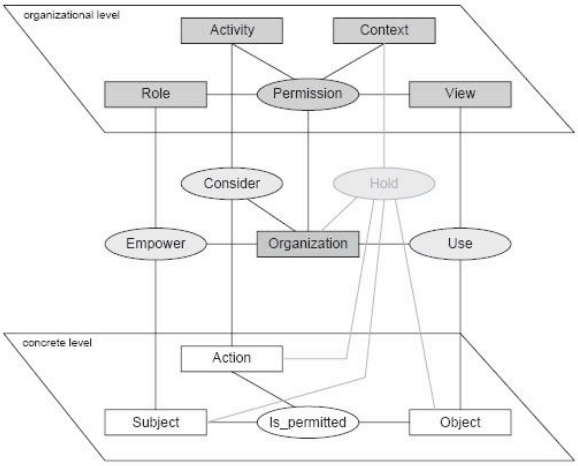

Figure 4: The OrBAC model.

The derivation of permissions (i.e., instantiation of security rules) can be formally expressed as follows:

∀org ∈ Org, ∀s ∈ S, ∀α ∈ A, ∀o ∈ O, ∀r∈ R, ∀a∈ A, ∀v∈ V, ∀c∈ C,
Permission (org, r, v, a, c) ∧
Empower (org, s, r) ∧
Consider (org, α, a) ∧
Use (org, o, v) ∧
Hold (org, s, α, o, c)
→ Is permitted(s, α, o)

This rule means:

If a security rule specifies that "in org, role r can carry out the activity a on the view v when the context c is True",
if "in org, r is assigned to subject s",
if "in org, action α is a part of activity a",
if "in org object o is part of view v" and,
if "the context c is True for the triple (org, s, α, o)".
Then subject s is allowed to carry out the action α on object o.

In our context, OrBAC present several benefits:

- *Rules expressiveness:* OrBAC defines positive authorizations (permissions), negative authorizations (interdictions), obligations, and constraints.
- *Abstraction of the security policy:* OrBAC has a structured and abstracted expression of the policy (Subjects are abstracted in Roles, Objects in Views, and Actions in Activities); it also separates the specification from the implementation of the security policy. Complexity can thus be well managed.
- *Scalability:* OrBAC has no limitation in size or capacity. It can define an extensible security policy. It is then easily applicable to large-scale environment.
- *Loose coupling:* each sub-system can manage its own security policy.
- Physical components management: network segment/physical equipment can be assimilated to an organization (network, firewall, gateways, routers, IDS, OS, DBMS, etc).
- *Evolvable:* a security policy in OrBAC is flexible and evolvable. It easily handles changes in organizations.
- *User-friendly:* the specification, management and update OrBAC security policy is a little bit intuitive.
- *Standardized:* OrBAC has a growing scientific community. Many research tracks are being conducted.

However, OrBAC is not really adapted to collaborative systems. First, OrBAC is not able to manage collaboration-related aspects. In fact, as OrBAC security rules have the Permission(org, r, v, a, c) form, it is not possible to represent rules that involve several independent organizations, or even, autonomous sub-organizations of a particular collaborative system. Moreover, it is impossible (for the same reason) to associate permissions to users belonging to other partner-organizations (or to sub-organizations). As a result, if we can assume that OrBAC provides a framework for expressing the security policies of several organizations, it is unfortunately only adapted to centralized structures and does not cover the distribution, collaboration and interoperability needs. Secondly, the translation of the security policy to access control mechanisms is not treated in OrBAC. It is thus necessary to describe suitable architecture, scenario and implementation (e.g., credential's generation) of the collaborative system's security.

To cover these limitations, we suggest calling on some mechanisms of the WS technology. The next sub-section presents the most relevant ones for our study.

2.3. Web Services-based mechanisms

Web Services (WS) is increasingly considered as a set of technologies that provide platform-independent protocols and standards used for exchanging heterogeneous interoperable data services. Software applications written in various programming languages and running on various platforms can use WS to exchange data over computer networks in a manner similar to inter-process communication on a single computer. WS also provide common infrastructure and services (e.g., middleware) for data access, integration, provisioning, cataloging and security. These functionalities are made possible through the use of open standards, such as:

- XML (Extensible Mark-up Language) creates "common" information formats and shares both the format and the data on the Internet/intranets [10].
- SOAP (Simple Object Access Protocol) acts as a data transport mechanism to send data between applications in one or several operating systems. SOAP specifies how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer and exchange information [11].
- WSDL (Web Services Description Language) is an XML-based language used to describe the services that a business offers and to provide a way for individuals and other businesses to access those services [12].
- UDDI (Universal Description, Discovery, and Integration) is an XML-based registry/directory for businesses worldwide, which enables businesses to list themselves and their services on the Internet and discover each other [13].

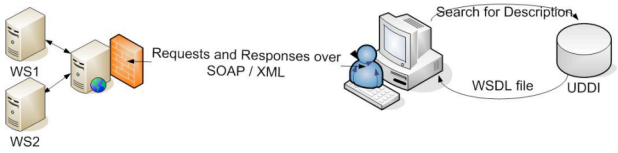

Figure 5: Functioning of web services.

Basically, when a user wants to use a specific WS (Figure 5), he contacts the UDDI to look for the WSDL file of the WS, then sends a request to the site that hosts this service, and finally receives the WSDL file containing the description of the service as well as the URL of the hosting site of the WS.

At low levels, data services are represented by the data storage element (DSE) and data themselves. The DSE is responsible for saving and retrieving files to/from local storage (e.g., disk, database). DSE is accessible via the WS interface, which exposes all the DSE functionalities (i.e., services virtualization).

Web services (WS) have several benefits that could be interesting in our context:

- Interoperability: WS support interoperability between software components from different platforms.
- Resources sharing: WS are well adapted to web applications where organizations share their resources.
- Standardized mechanisms: WS use open protocols and standards (e.g., HTTP, XML). They can be used easily with today's Web Interfaces.
- Easiness: a small amount of code is necessary to develop a WS. Moreover, the execution of a WS does not necessarily require huge resources.
- Compatibility with OrBAC: it is easy to couple web services with OrBAC.

3. PolyOrBAC

In this section, we suggest adapting OrBAC as well as WS mechanisms to specify and enforce secure collaboration between organizations. The global framework is called PolyOrBAC. The main idea is:

- Extending OrBAC to be able to express local access control policies (for each organization) as well as (collaboration) rules implying several organizations. In this way, the same rule, for example Permission(organization, role, activity, view, context), could concern several internal as well as external accesses.
- Using existing WS standards to enforce the collaboration at service and resource levels.

3.1. Scenario of execution

Let us develop a simplified (but representative) scenario adapted to collaborative systems. We distinguish two global phases.

3.1.1. First phase publication and negotiation of collaboration rules as well as the corresponding access control rules.

First, each organization determines which resources it will offer to external partners. Web services are then developed on application servers, and referenced on the Web Interface (in UDDI) to be accessible to external users. At this stage, we find in B security rules such as:
Permission(B, Accountant, Account, Consulting, Urgency) and instances (of relations) such as:

- Empower(Bob, Accountant, Accountant),
- Consider(B, OpenXMLFile(), Consulting), and
- Use(B, WS1, Account).

Second, when an organization publishes its WS at the UDDI registry, the other organizations can contact it to express their profit sharing. In the example below, organization B offers WS1, and organization A is interested in using WS1.

Third, organizations A and B negotiate and come to an agreement concerning the use of WS1.

Fourth, A and B establish a contract and jointly define security rules concerning the access to WS1, These security rules are registered – according to an OrBAC format – in a database containing the Security policy1.

The steps of this phase are given in Figure 6.

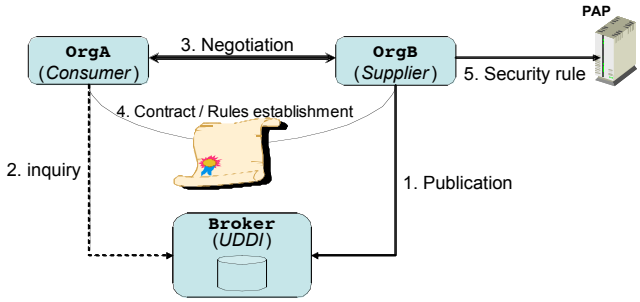

Figure 6: Mutual negotiation of access rules for distant services.

For example, if the agreement between A and B is "users from A have the permission to consult B's accounts, B should add the Empower(B, PartnerA, Accountant) association to its base2. In this notation, PartnerA means any user from A.

We assume that the security policy database already contains the rule Permission(B, Accountant, Account, Consulting, Urgency).

The derivation of the permission (i.e., instantiation of security rules) mentioned above can be formally expressed as follows (Figure 7):

Permission(B, Accountant, Account, Consulting, Urgency) ∧
Empower(B, Bob, Accountant). ∧
Consider(B, OpenXMLFile(), Consulting) ∧
Use(B, WS1, Account) ∧
Hold (B, PartnerA, OpenXMLFile(),WS1, Urgency)
→ Is permitted(PartnerA, OpenXMLFile(), WS1)

Figure 7: Derivation of permissions in PolyOrBAC.

3.1.2. Second phase access to remote/collaboration services.

At runtime, if a user wants to carry out an activity, the security-related services check requestor/request authentication, verify its credentials, make an authorization decision based on the security policy, and finally, deny or authorize the access (in some cases, this access is accompanied with some obligations or recommendations). In this vision, it is important to separate authentication from authorization, and access decision from permissions enforcement.

In our study, we use an AAA (Authentication, Authorization and Accounting) architecture: the authorization decision is requested by a requestor (user) or a resource service, if the security policy allows this access, an authorization ticket is delivered to the requestor; the latter presents the ticket with the authorization context to the resource or service. More precisely, if a user from "A" (let us note it Alice) wants to carry out an activity, A is first authenticated. Then, protection mechanisms of organization A check if the OrBAC security policy of A allows this activity. We suppose that this activity contains local as well as external accesses. Local accesses should be controlled according to A's security policy, while remote accesses should respect the agreements established between organization "A" and the other organizations (containing the requested services).

If, for example, Alice's Activity invokes (among others) B's web service WS1, the access to WS1 should be controlled by B's Policy Enforcement Point "PEP", according to: (1) The OrBAC security policy of B, and (2) the agreement established between A and B about WS1.

It is important to note that the same (abstract) rule, e.g., Permission(B, Accountant, Account, Consulting), can correspond to local as well as collaboration accesses. In fact, the decision corresponding to local access can be done according to:

Permission(B, Accountant, Account, Consulting, Urgency) ∧
Empower(B, Bob, Accountant) ∧
Consider(B, SELECT, Consulting) ∧

[1] In the OASIS/XACML (*eXtensible Access Control Markup Language*), this base is called a PAP, for *Policy Access Point*.

[2] We assume that this base already contains the rules mentioned before.

Use(B, Table1, Account) ∧
Hold (B, Bob, SELECT, Table1, Urgency)
→ Is permitted(Bob, SELECT, Table1)

While the decision corresponding to remote access can be done according to:

Permission(B, Accountant, Account, Consulting, Urgency) ∧
Empower(B, PartnerA, Accountant) ∧
Consider(B, OpenXMLFile(), Consulting) ∧
Use(B, WS1, Account) ∧
Hold (B, PartnerA, OpenXMLFile(),WS1, Urgency)
→ Is permitted(PartnerA, OpenXMLFile(), WS1)

Let us also remind that the decision of "Which user from A is associated to PartnerA, and so, authorized to access to WS1" is done according to the A's security policy. In other words, A defines internally instances such as (Alice, PartnerA), (Jean, PartnerA).
In this way, when Alice is authenticated and authorized (by A's policy) to play PartnerA, an XML-based authorization ticket "T1" is generated (based on the positive decision) and granted to Alice.

T1 contains the following elements:

- the virtual user played by Alice: "PartnerA",
- Alice's organization: "A",
- the agreement's (between A and B) ID,
- the requested service: "WS1",
- the invocated method, e.g., "OpenXMLFile()", and
- a timestamp to prevent reply attacks.

Note that T1 is delivered to any user (from A) allowed to access to WS1 (e.g., Jean). When Alice presents its request as well as T1 (as a proof) to B, B extracts the T1's parameters, and processes the request. By consulting its security rules, B associates the role Accountant to the virtual user "PartnerA" (representing Alice in B) according to Empower(B, PartnerA, Accountant). The access decision is then done according to the rule presented in Figure 7.

3.2. WS mechanisms in PolyOrBAC

In our implementation, as we use a WS-based architecture, messages exchanged (e.g., services) between A and B are XML files that obey SOAP protocols. Moreover, PolyOrBAC could be integrated perfectly into XACML architecture (Figure 8) [15, 16].

In this architecture, an access request arrives at the Policy Enforcement Point (PEP), the PEP creates an XACML request and sends it to the Policy Decision Point (PDP), which evaluates the request and sends back a response. The response can be either access permitted or denied, with the appropriate obligations. The PDP comes to a decision after evaluating the relevant policies. To get the policies, the PDP uses the PAP to extract the security rules (e.g., Permission(Organization, Role, View, Activity, Context)). The PDP may also invoke the Policy Information Point (PIP) service to retrieve the attribute values related to the organization, the subject, the web service (resource), or the environment (the context). This consists in evaluating the associations Empower (org, s, r), Consider (org, α, a), Use (org, o, v) and Hold (org, s, α, o, c). The authorization decision arrived at by the PDP is sent to the PEP. The PEP:

- fulfils the obligations and/or informs the subject about the recommendations, and,
- based on the authorization decision sent by PDP, either permits or denies access.

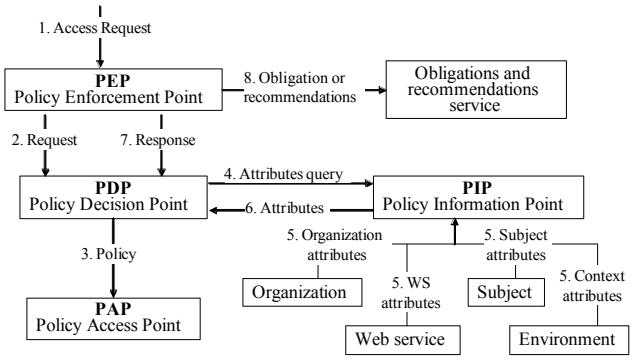

Figure 8: The XACML Architecture.

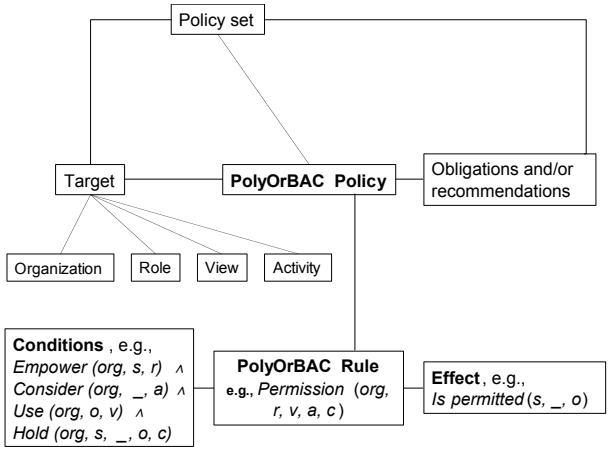

Figure 9: PolyOrBAC-architecture based on XACML.

Figure 9 describes the components of a PolyOrBAC implementation based on XACML (the target contains instances of (Organization, Role, View, Activity).

3.3. Discussion

PolyOrBAC offers several benefits:

- *Peer to peer approach*: we use a decentralized architecture where organizations mutually negotiate their common rules; each organization is responsible for its user's authentication and is liable for their use of other organizations' services; it also controls the access to its own resources and services.
- *Independence*: even if all PolyOrBAC rules are specified according to OrBAC, organizations are loosely coupled, e.g., each organization keeps its specific security policy, security objectives, services, applications, operating system, etc.
- *Information non-disclosure*: the WS technology allows communications between organizations without intimate knowledge of each other's IT systems; moreover, even if remote accesses are possible, it is not necessary to know the hierarchical composition of the other organizations.
- *Extensible structure*: the OrBAC extensibility and the WS standards facilitate the management and the integration of new organizations (with their users, data, services, policy, etc.).

In the next section, we address (and compare with PolyOrBAC) the most important access control models and mechanisms used for secure collaborative systems.

4. Related work

4.1. RBAC based approaches

Intermezzo [17] is one of the first works that address security issues in collaborative systems. It proposes a Role-based language that can be applied to collaborative settings. In the same logic, RBTM (Role-Based Trust Management) [14] has modeled collaboration systems by using role delegations and role mapping across multiple collaborating organizations. Typically, each organization can delegate local roles to users belonging to other organizations.

However, neither the role mapping nor the delegation process is intuitive in heterogeneous and dynamic systems. Moreover, Intermezzo's work as well as RBTM, only abstract subject (by roles), while PolyOrBAC expresses the whole security policy with abstract entities only.

4.2. O2O: Virtual Private Organizations Approach

The main concept of O2O (for Organization to Organization) is Virtual Private Organizations (VPO) [18]. Actually, if an organization Alice.org wants to interoperate with Bob.org, each organization defines its VPO, respectively called A2B (for Alice2Bob) and B2A. The VPO A2B contains a security policy that manages how subjects from Alice.org may have access to Bob.org; and similarly, B2A control accesses of subjects from Bob.org to Alice.org.

The O2O approach has some limitations. First, O2O allows a given subject to keep the same role when accessing remote organizations (the RSSO principle), while in several real applications, privileges associated to the same role name can differ from an organization to another. Second, as a new VPO is created for every temporal collaboration between two organizations, the management of the VPO becomes heavy. Moreover, this VPO is destroyed after the collaboration, and it will be necessary to recreate it if the same collaboration is carried out later. Third, as the VPO A2B and its security policy is defined and administrated by B, A should know some internal information about B such as the structure of B, its roles, hierarchy, etc. Therefore, this approach does not preserve the organization's privacy.

4.3. Coalition Based Access Control

CBAC [22] contains three levels of abstraction: user-object, role and coalition levels. First, a user requests access to remote object (user-object level) and the user's local role is identified (role level). Then, credentials associated with this role are extracted (role level) and a request containing the credentials is sent to the entity handling the requested object (coalition level). Afterwards, credentials necessary to access this object are extracted from those associated to the roles. These credentials are then compared with required credentials (role level). Finally, access to remote objects is permitted or denied (user-object level).

The CBAC approach has some limitations. The first problem mentioned for the O2O approach is also valid for CBAC. Second, we need to assign identifiers to different organizations composing the coalition, and an identifier to the coalition. This implies that there is a third part organization, responsible for assigning these identifiers. While in our approach, we prefer a decentralized architecture. And finally, the separation

between the security policy's specification and its implementation is not clear.

Unfortunately, due to space limitation, it is impossible to discuss all the existing works. But globally, we can conclude that PolyOrBAC improves some important points, in particular, PolyOrBAC:

- Completely separates the representation of the security policy from its implementation;
- Provides a global and homogeneous view of the security policy;
- Improves the management of the security policy and reduces considerably its complexity;
- Provides a global framework (security policy and security mechanisms) while several existing works deal only with the policy;
- Takes advantage of the WS technology in implementing the concepts of PolyOrBAC (which provides interoperability and privacy).

5. Conclusions and perspectives

PolyOrBAC is an extension of OrBAC that enables a better access control for collaborative systems in distributed and heterogeneous contexts. In these systems, users belonging to an organization need to dynamically access resources controlled by other organizations. In PolyOrBAC, security rules are specified only through abstract entities. Moreover, the same OrBAC security policy can be used for local as well as remote (collaboration) accesses. In this way, PolyOrBAC improves the management of the security policy and reduces considerably its complexity. Besides, PolyOrBAC uses web services mechanisms (XML, SOAP, WSDL, UDDI) to implement the security policy. In addition, we have shown how PolyOrBAC can be incorporated in a XACML architecture. The use of WS standards in PolyOrBAC improves the interoperability and the (secure) resources sharing, which are crucial points in collaborative systems. Now, we are looking for extending the formal system associated to OrBAC, in particular for detecting and resolving conflicts that could arise between the security policies associated to several collaborating systems. We also consider defining an administration model associated to PolyOrBAC. Finally, we should study the negotiation process of collaboration policies as well as the exchange of credentials. Approaches like those used by TrustBuilder [19] or Trust-X [20] could be interesting in our context.

Acknowledgment

This work is partially supported by the French SATIN ACI and by the European project CRUTIAL.

6. References

[1] Common Criteria for Information Technology Security Evaluation, v3, Part 1: Introduction and general model, 79 p., ISO/IEC 15408-1, July 2005.

[2] A. Abou El Kalam, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, C. Saurel, "OrBAC", IEEE 4th Int. Workshop on Policies for Distributed Systems, IEEE Computer Society Press, Italy, 4-6 June 2003.

[3] D.E. Bell, L.J. LaPadula, Secure Computer Systems, MTR 2997, MITRE corp., USA, 1976.

[4] M.A. Harrison, W.L. Ruzzo and J.D. Ullman, "Protection in Operating Systems", Communication of the ACM, 19(8), august 1976, pp. 461-471.

[5] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. "Role-based access control models". IEEE Computer, 29(2), 1996, pp38–47.

[6] D. Ferraiolo, R. Sandhu, S. Gavrila, D.Kuhn, R. Chandramouli "A Proposed Standard for Role-Based Access Control", ACM Transactions on Information and System Security, v 4, n° 3, 2001.

[7] R.S. Sandhu, "Role Hierarchies and Constraints for Lattice-Bases Access Controls", in 4th ESORICS, Springer-Verlag, Rome, Italy, September 25-27, 1996.

[8] G. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification", ACM Transactions on Information and System Security, vol. 3, n° 4, 2000.

[9] J. Vitek, C. Jensen, "A View-Based Access Control Model for CORBA", Secure Internet Programming, LNCS 1603, Springer, 1999.

[10] W3C, "Extensible Markup Language (XML)", W3C Recommendation, February 2004.

[11] W3C, "SOAP, Version 1.2" W3C Recommendation, June 2003.

[12] W3C, "WSDL, Version 2.0", W3C Candidate Recommendation, March 2006.

[13] OASIS, "UDDI Specifications TC, Universal Description", v3.0.2, February 2005.

[14] N. Li, J.C. Mitchell, W.H. Winsborough, Design of A Role-based Trust-management Framework, IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 2002. pp. 114-130.

[15] A. Matheus. "How to Declare Access Control Policies for XML Structured Information Objects using OASIS" HICSS, v. 7, no. 7, pp. 168a, 2005.

[16] OASIS, XACML Specification V1.1, OASIS: www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf, 24 July 2003.

[17] W. K. Edwards, "Policies and Roles in Collaborative Applications". ACM Conference on Computer Supported Cooperative Work (CSCW), November 16-20, 1996, Boston, MA, USA 1996.

[18] F. Cuppens, N. Cuppens-Boulahia et C. Coma. "O2O: Virtual Private Organizations to Manage Security Policy Interoperability", 2nd Int. Conference on Information Systems Security, Calcuta, India, 2006.

[19] T. Yu, M. Winslett, K. Seamons, "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiations". ACM Transactions and Information System Security, 6(1), February 2003.

[20] E. Bertino, E. Ferrari, A. Squicciarini, "X-TNL: An XML Based Language for Trust Negotiations", 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003, pp. 81–84.

[21] T. Fink, M. Koch, C. Oancea. "Specification and Enforcement of Access Control in Heterogeneous Distributed Applications". International Conference on Web Services (ICWS-Europe), 2003, pp. 88-100.

[22] V. Atluri, J. Warner, "Automatic Enforcement of Access Control Policies Among Dynamic Coalitions", Distributed Computing and Internet Technology, 1st Int. Conference (ICDCIT), 2004, pp. 369-378.