XML-Based Revocation and Delegation in a Distributed Environment

Konstantina Stoupa[1], Athena Vakali[1], Fang Li[2], and Ioannis Tsoukalas[1]

[1] Department of Informatics, Aristotle University, Thessaloniki, Greece
{kstoupa,avakali,tsoukala}@csd.auth.gr
[2] Department of Computer Science, Shanghai Jiao Tong University, China
Fli@mail.sjtu.edu.cn

Abstract. The rapid increase on the circulation of data over the web has highlighted the need for distributed storage of Internet-accessible information due to the rapid increase on the circulation of data over the web. Thus, access control mechanisms should also be distributed in order to protect them effectively. A recent idea in the access control theory is the delegation and revocation of rights, i.e. the passing over of one clients rights to the other and vice versa. Here, we propose an XML-based distributed delegation module which can be integrated into a distributed role-based access control mechanism protecting networks. The idea of X.509v3 certificates is used for the transfer of authorization information referring to a client. The modules are XML-based and all of the associated data structures are expressed through Document Type Definitions (DTDs).

1 Introduction

Role-based access control [11] seems to be the ideal model for large-scale heterogeneous networks since they organize clients into categories according to their duties (in an organization or enterprise). Moreover, large-scale environments require a level of self-administration [1, 2, 7, 9]. If a client is off duty, (s)he should pass over her/his authorizations to another client without the interference of a central administrator. Such an opportunity is given through delegation of roles and authorizations. A system that supports delegation of rights, should also support their revocation. Each delegation demands auditing in order to have the ability to return to the initial condition.

Distributed systems require a medium for transferring access control information concerning a user. This can be achieved through the use of certificates which are electronic documents containing basically identity (and other information) about their owners [3]. The most well-known proposals covering this functionality are the X.509v3 certificates and the Attribute Certificates (an overview is presented in [5, 6, 10]. The public-key certificates X.509v3 is an ISO/IETF standard which certifies both the identity and the attributes of a client and they are digitally signed by a certification authority. An X.509v3 certificate except for the core fields (issuer, licensee, public keys, etc.), it also contains some fields for extension. Here we adopt the X.509v3 certificates and we use the extension fields to store all of the needed access control information (including the delegation and revocation information). Those certificates are signed by internal or

W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 299–308, 2004.

external authorities and they are updated each time an access control procedure modifies the characteristics of clients.

In the context of delegation earlier research efforts have focused on centralized delegation. RDM2000 (Role-based Delegation Model 2000) is a centralized role-based delegation model supporting hierarchical and multi-level delegation [12]. The main idea of this system is that it allows users acting in a specific role to delegate roles to other users. The series of PBDM (Permission-based Delegation Model) models extends this idea. PBDM0 also allows the user-to-user delegation of permissions while PBDM1 and PBDM2 supports role-to-role delegation [13]. In order to satisfy such a need, PBDM uses a central security administrator controlling the permission flow by defining separately delegatable roles. SQL also supports a kind of delegation since it uses the GRANT-REVOKE commands. It is about a user-to-user or user-to-role approach.

All of the above models are centrally administered and therefore, not adequate for large-scale distributed environments because the delegation mechanism would become a bottleneck. The major characteristics of our approach are:

1. the distributed orientation, i.e. there are several local delegation and revocation modules supporting local requests and a global module for servicing clients requests.
2. the support of both user-to-user and role-to-role delegation by employing the idea of administrative roles which can modify the features of regular roles. Therefore, each user is assigned both regular and administrative roles. In case the delegatee (whom the authorizations or roles are delegated to) is a role the delegator should be or act in an administrative role.
3. the support of delegated object which may be either role or authorization.
4. the ability to be integrated into a distributed access control mechanism having a central authorization certificate issuing authority.

Since there is great research interest in building XML-expressed access control policies (such XACML, XrML, ODRL), XML was our choice in implementing the module. Moreover, XML is appropriate for expressing semi-structured data and metadata (e.g. the format of the protected resources, the policies, etc). Such a module can be integrated into existing access control mechanisms or modern frameworks (such as XACML, XrML) able to protect the resources of large heterogeneous organizations in order to extend their functionality (so as to support delegation and revocation).

We believe that our contribution is significant since there is little been done in distributed delegation of authorizations or roles. Moreover, we have decided to use XML to express the major entities of our models since there are already standardized XML-based access control languages, a feature that will help us in integrating our module into existing access control frameworks. Our work advances the current state of the art since we introduce the idea of delegation into Internet-accessed distributed protected networks. Moreover, we have tried to design a both user-to-user and role-to-role delegation of authorizations or roles in order to complete the functionality of such a module.

The remainder of the paper is as follows: Section 2 describes the appropriate environment and access control mechanism where our delegation/revocation modules can

best be integrated. Section 3 describes the general function of the delegation and revocation modules. Section 4 elaborates on delegation and revocation requests and their XML format is given. Moreover, we relate the delegation/revocation procedure with the Authorization Certificates by focusing on the modifications made on them in case a delegation or a revocation takes place.

2 Resource Accessing Over a Distributed Topology

The distributed environment consists of several servers (forming a local network or a Virtual Private Network (VPN)) protecting repositories and supporting a number of *internal clients*, i.e. clients accessing the system form inside the local network (Fig. 1). One of the servers is characterized as *master server* since it is also the one connecting the local environment with the Internet. The rest of servers are characterized as *slaves*. Every internal client sends his/her access control request to the server it is connected to (slave or master).

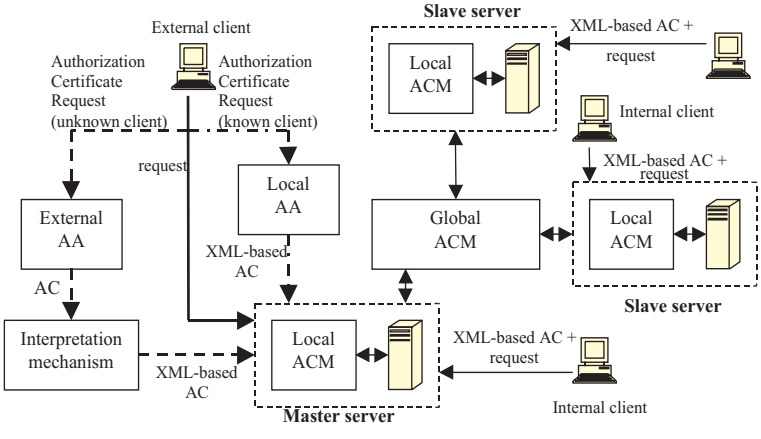

Fig. 1. Function of the distributed access control environment

In case of *external clients*, (i.e. clients trying to enter the local environment through Internet), every access control request is sent to the master server which decides which slave server should forward the request to. Of course, since master server is also supporting a repository, it may fulfill the request itself if the request refers to it.

This Internet-accessible distributed environment can be employed in a large-scale organization or enterprise (e.g a geographically distributed multinational enterprise) whose network can also be accessed by external partners through Internet. Consider the scenario of a national bank having many branches along the country. Every branch forms a sub-network supported by a local server. Of course, all of these sub-networks are connected to form the whole network of the bank. In this case a client may try to enter the network from outside (e.g. from his/her home) in which case acts like an external user,

or (s)he may try to gain access from inside the main network (e.g. through a machine connected to the sub-network of Athens branch). Internal users can send requests to the server (either master or slave) supporting their location. External users' requests pass through the master server, which processes them or redirect them to the appropriate slave server.

Each server contains a local access control mechanism whose rules have power only in the local network it supports. Moreover, there is a global access control mechanism which is placed in the "center" of the environment and contains some general rules or policies governing the function of all the servers. Since, the proposed environment is role-based, both subjects and objects are organized into roles [8] both *local* and *global* ones. Local roles have effect only in a sub-network. For example there may be a role named "employee" in the Athens branch which characterizes only the employees of this branch. A global role could be the general manager of the bank whose authorizations have effect in the whole network. XML is used for expressing roles, policies, authorizations and certificates which are associated with each user and contain his/her features.

Every access control mechanism (ACM) needs (a) a request (either access request or delegation request) and (b) authorization information about a client (which is sent through an XML-based authorization certificate (AC)).

External users can be both known and unknown. According to the case the request is fulfilled through two distinct routes.

1. In case the client is unknown to the system, a trusted Authorization Authority (AA) is asked to issue an Authorization Certificate (AC). Due to the miss of a general standard for such certificates, their format may vary according to which authority has issued them. Therefore, the certificate should be interpreted in an XML-expressed AC recognizable by the access control mechanism (dashed line route).
2. In case the client is known, a local Authorization Authority issues directly the XML-based certificate which is passed to the access control mechanism (dash dot line route).

3 Delegation and Revocation Modules

Delegation and revocation processes are part of the access control mechanism. Therefore, these processes have to be included in a distributed access control system (as the one depicted in Fig. 1).

3.1 The Delegation Module

Fig. 2 depicts the format of the proposed delegation module. The client sends a delegation request escorted by its XML-based authorization certificate (AC) to the local delegation mechanism (which is part of the local Access Control Mechanism–ACM). This mechanism can fulfill delegation concerning local entities (roles and authorizations). In case, the delegation request concerns local entities and it is valid, the local delegation mechanism satisfies the request and updates the authorizations certificate which is sent back to its owner. On the other hand, when the delegation request refers to global entities, it is passed over to the global delegation mechanism (which is part of the global ACM),

which is in charge of sending the updated authorization certificate to the client. Moreover, there is a database storing copies of the authorization certificates required for the revocation procedure.

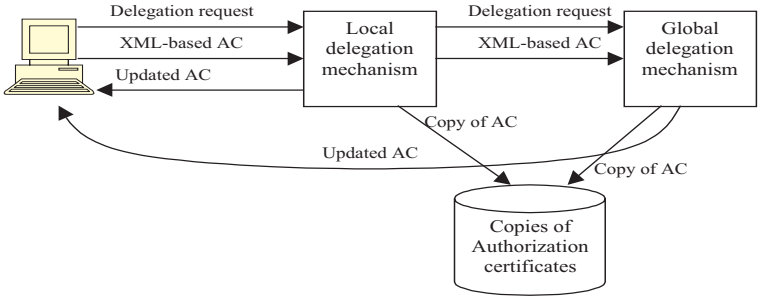

Fig. 2. Function of delegation module

3.2 The Revocation Module

The revocation procedure is always conducted by both global and local modules in unison. The reason is that a delegation of roles or authorizations which initially concerned local entities may also be propagated to global entities through cascading delegation. For example, consider the general manager delegates his role to the manager of Athens branch, who in his turn delegates it to the financial manager of his branch. In case, the general manager wants to revoke his role, the revocation mechanism should scan the base with the copies of ACs and communicate with all of the local mechanisms servicing roles that have been granted the revoked role or authorization, in our case the Athens branch. A revocation request is always sent from the interested client to the connected local module and typically revocation is practiced as:

1. **One-Level**: in case the delegation was one-level. This type of revocation may concern:
   (a) *Delegation of a local role*: the revocation request is fulfilled by the local revocation module. (dash lines)
   (b) *Delegation of a global role*: the revoked object is a global role or authorization and therefore the request is passed to the global module which fulfills it (dash lines+dash dot lines).
2. **Cascading**: in case of cascading delegation. This type of revocation may concern:
   (a) *Cascading into local roles*: the revocation request is fulfilled by the local module by revoking the object from every role where it is delegated. (dash lines)
   (b) *Cascading into both local roles of the same subnetwork and global roles*: the local mechanism revokes the object from the local roles and afterwards it sendsa revocation request where the revocator is a local role and the revocatee a global one (dash lines+dash dot lines).
   (c) *Cascading into global and local roles of another subnetwork*: the local mechanism upon receiving the request tries to fulfill it to the point the revoked object

has not be delegated to global roles. In this case it passes over the updated request (with the new revocator and revocatee) to the global mechanism which in its turn revokes the object from the global roles. In case the revoked object has been delegated to local roles empowered into other sub-networks, the global module sends an appropriate request to the appropriate local revocation module (dashed lines+dash dot lines+round dot lines).

Every time a revocation request is serviced, both the client and the copies base is informed.

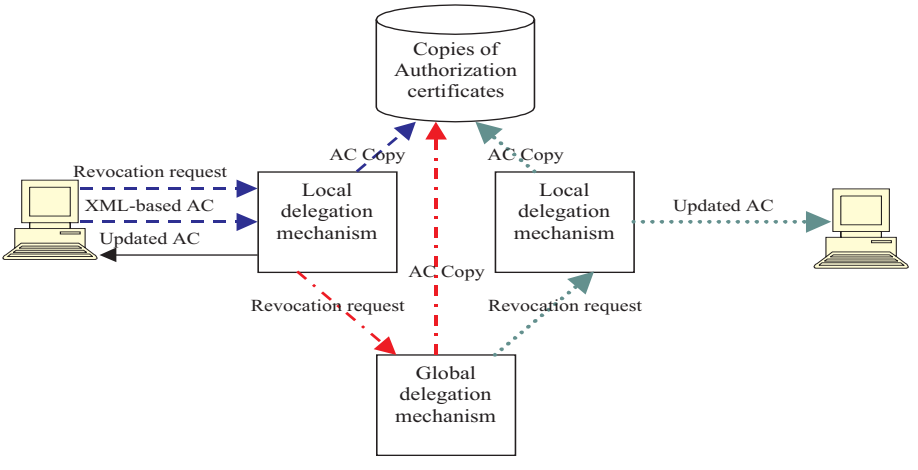

Fig. 3. Function of revocation module

4 Delegation and Revocation Requests

Delegation is triggered through delegation requests which define which *delegator* (users acting in a certain role, or a role) wants to delegate which *delegation object* (certain authorizations or whole roles) to which *delegatee* (user or role).

Each delegation request is represented by the following ELEMENT (in DTD[1]).

```
<!ELEMENT delegation_request (delegation_structure,
                              delegation_constraints)>
```

where (a) the delegation structure part defines the delegator and the delegate, as well as the roles (or authorizations) that are to be delegated (i.e the delegation object) and (b) the delegation constraints part describes the features of the delegation.

[1] The reason we use DTD instead of XML Schema is brevity in presentation. In the implmentation XML Schema will be used.

4.1 Delegation / Revocation Structure

Delegation structure is expressed with the following element:

```
<!ELEMENT delegation_structure (delegator,delegatee+,
           delegated_object,scope)>
```

The *delegator* may be a user acting in a regular role or an administrative role (either local or global), the *delegatee* may be a user or a regular role and the *delegated_object* may be a regular role or a set of authorizations. When users act in regular roles, they cannot delegate roles or authorizations to other roles since they are not allowed to modify a role. Such a task is allowed only to administrative roles (forming the administrative hierarchy). Thus, the following cases arise in a delegation request:

1. For delegators acting in regular roles
   (a) Delegate role to user. When a user u_1 acting in regular role r_1 delegates his regular role r_2 to user u_2 (of course $r_2 \leq r_1$ in the role hierarchy, which means that since u_1 possesses r_1 (s)he also possesses r_2)
   (b) Delegate authorizations to user. When user u_1 acting in regular role r_1 delegates his/her authorization a to user u_2.
2. For delegators acting in administrative roles
   (a) Delegate role to role. When user u_1 acting in administrative role r_1 delegates his/her regular role r_3 to regular role r_2.
   (b) Delegate authorizations to role. For example, administrative role r_1 delegates authorization a to regular role r_2.

```
<!ELEMENT revocation_request (revocator, revocatee,
          revocated_object,scope)>
<!ELEMENT revocator (user?, revocator_role)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT revocator_role (#PCDATA)>
<!ATTLIST revocator_role type (regular|administrative)>
<!ATTLIST revocator_role id ID #REQUIRED>
<!ELEMENT revocatee (#PCDATA)>
<!ATTLIST revocatee type (user|role)>
<!ELEMENT revocated_object (#PCDATA)>
<!ELEMENT scope (rrh1_identity, (arh_identity, rrh2_identity)?)>
<!ELEMENT rrh1_idenity (#PCDATA)>
<!ELEMENT arh_identity (#PCDATA)>
<!ELEMENT rrh2_idenity (#PCDATA)>
```

Fig. 4. Function of revocation module

Since an organization may contain many roles and administrative hierarchies, we should define the *scope* of the delegation, i.e. to which hierarchies it refers. Thus, the request structure should be enriched with two more fields defining the identity of the regular role and the administrative role hierarchy.

Similarly, the revocation request (since it is "opposite" to the delegation request) occurs when a client acting in a role revokes all or part of the rights (s)he has delegated in the past. Therefore, a revocation request should include the information shown in Fig. 4, where (a) $revocatee$ is the role or person the delegated object has been given to, (b) *revocated object* is the target of revocation (which may be an authorization or a whole role), (c) *rrh1_identity* is the scope of the revocator role, (d) *arh_identity* is the scope of the revocator role in case it is an administrative one and, (e) *rrh2_identity* is the scope of the revocatee.

4.2 Delegation Constraints

Typically, the main delegation constraints are:

1. *Permanence*: in case a delegation is permanent, the delegator permanently passes on his(her) authorizations to the delegatee.
2. *Monotonicity*: this feature refers to the "power" that the delegator possesses after the delegation. In a monotonic delegation, the delegator maintains his(her) authorizations.
3. *Totality*: this feature refers to the extent with which authorizations assigned to a role are delegated to another. In case of a total delegation the delegator passes over all of his(her) authorizations.
4. *Levels of delegation*: it defines whether a role can be further delegated and for how many times.
5. *Activation/de-activation condition* : every delegation should take place when a condition is fulfilled and it should be cancelled according to a de-activation condition. Those conditions can be anything, e.g. temporal ones (June 6th 2004).

A delegation request may or may not contain constraints, or it may contain a part of them. Therefore, the final format of the delegation request tuple is shown in Fig. 5.

After the completion of a delegation (or revocation) the AC of the delegator and the delegatees (or revocator and revocatees) should be informed. Such authorization certificates contain the characteristics of their owners. When a certificate is initially issued (both by the local and external authorization authorities) should include the following information: (a) the *licensee id*, (b) the *issuer id*, (c) the regular roles of the licensee, (d) the administrative roles, (d) the valid period, which depicts the life duration of the certificate and (e) some extension fields which include the following delegation/revocation information:

1. *Denied authorizations*: this list is expanded every time the licensee acting in a role delegates monotonically an authorization.
2. *Delegated objects*: this is a list consisting of the roles and authorizations that have been delegated to the licensee.
3. *Revocable objects*: a list containing the roles and authorizations that the licensee has delegated but (s)he has the right to revoke them at some time. Of course, this list contains only those subjects that have been temporary delegated.

```
<!ELEMENT delegation_request (delegation_structure,
          delegation_constraints)>
<!ELEMENT delegation_structure (delegator,delegatee+,
          delegated_object,scope)>
<!ELEMENT delegator (user?, role)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT role (#PCDATA)>
<!ATTLIST role id ID #REQUIRED>
<!ATTLIST role type (regular|administrative)>
<!ELEMENT delegatee (#PCDATA)>
<!ATTLIST delegatee type (role|user)>
<!ELEMENT delegated_object (#PCDATA)>
<!ELEMENT scope ((rrh1_identity, arh_identity,rrh2_identity)?)>
<!ELEMENT rrh1_idenity (#PCDATA)>
<!ELEMENT arh_identity (#PCDATA)>
<!ELEMENT rrh2_idenity (#PCDATA)>
<!ELEMENT delegation_constraints (permanence,
          monotonicity, delegation_levels,
          activation_condition, deactivation_condition)>
<!ELEMENT permanence empty>
<!ATTLIST permanence type (permanent|temporary)>
<!ELEMENT monotonicity empty>
<!ATTLIST monotonicity type (monotonic|non_monotonic)>
<!ELEMENT delegation_levels (#PCDATA)>
<!ELEMENT activation_condition (ANY)>
<!ATTLIST activation_condition type (temporal|event_driven)>
<!ELEMENT deactivation_condition (ANY)>
<!ATTLIST deactivation_condition type (temporal|event_driven)>
```

Fig. 5. Function of revocation module

5 Conclusions

In this paper we introduced a distributed delegation/revocation mechanism supporting Internet-accessible distributed environments consisting of several local networks. The most appropriate access control model seemed to be the role-based one. Roles were mainly introduced to organize the subjects (clients requesting access). We have also tried to use them for the categorization of protected resources. XML has been adopted to express requests and other access control issues (roles, policies, etc.). The medium proposed for transferring access control (and delegation) information about a user is the Authorization Certificate. Since there is not yet a standard format of such certificates we have introduced an interpretation mechanism which accepts every external certificate and translates it into an XML format recognizable by the access control mechanism. Here we have mainly focused on the delegation and revocation procedures. The future goal is to implement the proposed structures and algorithms in a prototype environment in order to evaluate their usage mainly over the Internet-accessed resources.

References

1. E. Barka and R. Sandhu. Framework for Role-Based Delegation Models. In *Proc. 16th Annual Computer Security Applications Conference*, pages 168–176, 2000.
2. E. Barkaand R. Sandhu. A role-based Delegation Model and Some Extensions. In *Proc. 23rd National Information Systems Security Conference*, 2000.
3. J. Dai and J. Alves-Foss. Certificate Based Authorization Simulation System. In *Proc. 25th Annual Int. Computer Software and Applications Conference*, pages 190–195, 2001.
4. C. Goh and A. Baldwin. Towards a more Complete Model of Role. In *Proc. 3rd ACM Workshop on Role-Based Access*, pages 55–61, 1998.
5. A. Herzberg, Y. Mass, L. Mihaeli, D. Naor, and Y. Ravid. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *Proc. Symposium on Security and Privacy*, pages 2–14, 2000.
6. J. Linn and M. Nystrom. Attribute Certification. An Enabling Technology for Delegation and Role-Based Control in Distributed Environments. In *Proc. 4th ACM Workshop on Role-based access control*, pages 121–130, 1999.
7. P. Michiardiand R. Molva. Inter-domain authorization and delegation for business-to-business e-commerce. In *Proc. 1st E-business and E-work Conference*, 2002.
8. M.J. Moyer and M. Ahamad. Generalized Role-Based Access Control. In *Proc. IEEE 21st Int. Conference on Distributed Computing Systems*, pages 391–398, 2001.
9. S. Na and S. Cheon. Role Delegation in Role-Based Access Control. In *Proc. 5th ACM Workshop on Role-Based Access Control*, pages 39–44, 2000.
10. J.S. Park and R. Sandhu. Binding identities and attributes using digitally signed certificates. In *Proc. 16th Annual Computer Security Applications Conference*, pages 120–127, 2000.
11. R.S. Sandhu, E.J. Coyne, and H.L. Feinstein. Role-Based Access Control Models. *IEEE Computer*, 29(2):38-47, 1996.
12. L. Zhang, G.-J. Ahn, and B.-T. Chu. A Rule-based Framework for Role-Based Delegation and Revocation. *ACM Trans. on Information and System Security*, 6(3):404–441, 2003.
13. X. Zhang, S. Oh, and R. Sandhu. PBDM: A Flexible Delegation Model in RBAC. In *Proc. 8th Symposium on Access Control Models and Technologies*, pages 149–157, 2003.