An XML-based Language for Access Control Specifications in an RBAC Environment*

Konstantina E. Stoupa
kstoupa@acn.gr

Athena I. Vakali
avakali@csd.auth.gr

Department of Informatics,
Aristotle University of Thessaloniki, Greece

Abstract - *Lately, Web-accessed resources have superceded the resources accessed by local or wide-area networks. Therefore, new mechanisms should be implemented for protecting resources from unknown clients. Attribute Certificates is a quite new technology offering such functionality. Those certificates are issued by Attribute Authorities validating the attributes of the owner of the certificate. Based on this technology an XML-based access control mechanism is introduced for protecting any kind of resources (from both known and unknown clients). The proposed model is ultimately role-based since both clients and protected resources are organized into roles. Moreover, an XML-based language is introduced to express roles, authorizations, delegation rules, hierarchies and certificates.*

Keywords: Access control, Attribute Certificates, Role-Based Access Control, XML-based language.

1 Introduction

Currently, a wide variety of services is available through the Internet and they need protection from unauthorized access. Access control and authentication are highly related, since granting access to an authorized user assumes the earlier authentication of his/her identity. For years, public key infrastructures have been employed for the authentication of users. The idea is quite simple: a few trusted authorities (certification authorities-CAs) authenticate the identity of a user signing with their private key an identity certificate containing such information.

Lately, this idea is also extended to the authorization mechanism. New types of certificates containing authorization information (known as attributes) have already been proposed. *Attributes* are characteristics of the requester related to the access control mechanism (e.g. roles). Attribute certificates (ACs) are signed by trusted Attribute Authorities (AAs) and they are bind to an identity certificate. Thus, every access control system should trust several AAs and accept the certificates they issue. Those environments not willing to trust anybody (except themselves), also have the responsibility of issuing their own ACs (after a negotiation with the subject). Unfortunately, there is not yet a standard governing the format of such certificates or the responsibilities of such authorities. Therefore, the whole idea is under consideration and research and only some abstracts have been made.

In this paper we present an XML-based access control environment for protecting any type of resource (files, system resources, services, etc.) from known and unknown clients. Each client is assigned with some roles certified by an attribute certificate. According to those roles, the access control mechanism decides whether it will grant or deny access. The proposed mechanism is ultimately role-based since both client and protected resources are organized into roles. Furthermore, we define an XML-based language for expressing roles authorizations, delegation rules, hierarchies and certificate.

1.1 Authorization Certificates

The most well-known proposals covering this functionality are the X.509v3 certificates and the Attribute Certificates (an overview is presented in [9]).

The public-key certificate 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 certificate authority. An X.509v3 certificate except for the standard fields, it also contains some extensions. The extensions field can be used for the incorporation of any number of additional fields and therefore attributes into the certificate. Therefore, there is no need for independent certificates leading to high protocol complexity and certificate administration. On the other hand, the integration of the two functions in one certificate is not problem-free since the two types of certificates may have different life durations, i.e. the clients authorized to perform an action may vary every week or even day, while

* **0-7803-7952-7/03/$17.00 © 2003 IEEE.**

identity certificates are designed to be valid for much longer period of time. In case that attributes need modification the whole certificate should be reissued. Moreover, the authority verifying the identity of a person may not be appropriate for certifying the corresponding authorization information ([6]).

The drawbacks of X.509v3 certificates try to overcome the Attributes Certificates which were developed by U.S. financial industry through the ANSI X9 committee. Those certificates are totally separated from identity certificates and bind attribute information to the certificate's subject. They are digitally signed by an attribute authority. Since attribute certificates does not contain a public key, they should be used in conjunction with an identity certificate.

In [12] a conjunction of the two certificates is proposed and it is called Smart Certificate. Those certificates are based to the extension of X.509v3 certificates and they can be both short- and long-lived eliminating the additional revocation mechanism. Furthermore, they can contain attributes issued by several authorities with various durations. Such a feature leads to low protocol and certificate administration complexity.

1.2 Attribute-Based Access Control

Attribute-based Access Control is a quite new idea which tends to extend the role-based access control. In [8] RT framework is introduced which is a family of role-based Trust-management languages for expressing policies and attributes in distributed access control mechanisms. RT covers almost all of the access control issues by using BNF representation.

In [5], Trust Policy Language (TPL) is introduced. It is about an XML-based language for mapping strangers to predefined business roles based on certificates. An SQL-based language for expressing mobile policies is defined in [3]. This idea further extends attributes certificate functionality, since they can not only transfer attributes of the owner but also some access control policies governing them.

Nereus is a trust-management framework where attributes are assigned to clients through distinct certificates ([10]). A format for both credential and delegation certificates is proposed. Those certificates are directed to access control mechanism which, according to predefined policies, decides whether it will grant or deny access privileges to the requester.

Finally, in [2] a Certificate-Based Authorization Simulation System is introduced. It emulates some basic functions of an operating system, such as machine, user and file management using certificates instead of access control lists.

2 Access Control Basics

The core issues governing an access control mechanism function are: (a) *the subject* who is the client willing to access the protected resources, (b) *the object* which is the protected resource and (c) *the access mode*. These issues are engaged in the definition of an authorization rule which defines which subject may conduct which type of operation over which object.

2.1 Ultimate Role-Based Access Control

For reasons of unification, our model will be an ultimately role-based access control tool. Sandhu et. al. in [13] introduced the idea of roles in subjects. It is the most appropriate identification method for wide heterogeneous web-accessed repositories. *Role* is a named collection of privileges needed to perform specific activities in the system. The important benefit of role-based models is that they organize the various requesters into categories according to their characteristics. This feature allows the access control administrators to define security policies affecting a group of clients whose identities are of not interest. Clients are assigned one or more roles and the same holds between roles and permissions.

Role-based models have a number of characteristics that make them flexible and effective. Authorizations are specified for a set of subjects, e.g. employees, managers, etc. Thus, the number of rules to be defined is really minimized. Each role defines a certain obligation and duty in an organization. Thus, their use seems to fulfill exactly the needs of modern distributed organizations. Finally, roles can be naturally organized into hierarchies

Subjects are the clients willing to access the protected resources. It has already been mentioned that both known and unknown clients are allowed to enter a distributed web-accessed environment. Therefore, the framework is obliged in both cases to assign roles to subjects (through attribute certificates). It is expected that in the latter case, roles will be less privileged than those assigned to known clients.

Based on the advantages offered by roles and by the work in [11], we have extended the use of roles into objects, too. Since protected *objects* form a huge and heterogeneous set, their categorization into groups seems mandatory. In case, there should be a different policy for each protected object the access control mechanism would become a bottleneck for the modern distributed systems responsible for the protection of various resources (files, system resources, services, etc.). Therefore, we may have the following object roles related to subjects: video files, audio files, data files, etc.

Another feature of role-based access control models is the idea of role hierarchies. The *role hierarchies* represent which roles have at least the authorizations of their children. Therefore, a child node in a role-tree has more authorizations than its parent.

2.2 Delegation

Since modern access control models are destined to protect wide, distributed environments, they should be self-administered in order to avoid becoming a bottleneck. Such a functionality is achieved through *delegation of roles* i.e. the ability of a role to pass (over its authorizations) to another role. Roles are organized into a *delegation hierarchy* where a role is allowed to delegate its authorizations to a direct (or indirect) child role or to a role in the same level.

Delegation is related to the following features (which we incorporate in our framework) ([1], [4]):

- *Scope*: It is not scarce that a role may be involved in more than one delegation hierarchies The scope of its power is given by the identity of a hierarchy.
- *Permanence*: in case a delegation is permanent, the delegator permanently passes on his(her) authorizations to the delegatee.
- *Monotonicity*: this feature refers to the power that the delegator possesses after the delegation. In a monotonic delegation, the delegator maintains his(her) authorizations
- *Totality*: this feature refers to how completely the authorizations assigned to a role are delegated to another. In case of a total delegation the delegator passes over all of his(her) authorizations.
- *Levels of delegation*: it defines whether a role can be further delegated and for how many times.
- *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.

3 The proposed access control framework

We have implemented a language in our attempt to support the topology shown in Figure 1. In the one end is a repository of resources needing protection and at the other end there is a client requesting to grant access to a protected resource. For the request to be fulfilled, it should first pass through the access control mechanism. This mechanism needs two type of information: *(a)* a request from the client containing under which operation mode the client wants to access which object (arriving through route A), and *(b)* an attribute certificate containing the authorization information related to the client. Those attributes may be received through two distinct routes: *(a)* in case the client is unknown to the system, it asks a trusted Attribute Authority (AA) to issue an Attribute Certificate (AC). Due to the miss of a standard for those certificates, their format may vary according to which authority has issued them. Therefore, the certificate should be interpreted in an XML-based attribute certificate recognizable by the access control mechanism (red route). *(b)* In case the client is known, a local attribute authority issue directly the XML-based certificate which is passed to the access control mechanism (blue route).

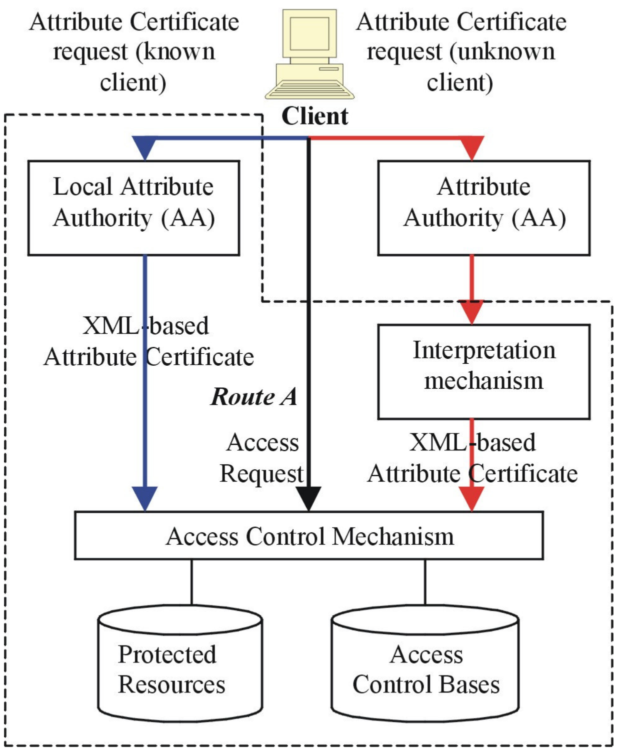
Attribute Certificate request (known client)
Client
Attribute Certificate request (unknown client)
Local Attribute Authority (AA)
Attribute Authority (AA)
XML-based Attribute Certificate
Route A
Access Request
Interpretation mechanism
XML-based Attribute Certificate
Access Control Mechanism
Protected Resources
Access Control Bases

Figure 1 : The architecture of the proposed environment

3.1 Function of Access Control Mechanism

The access control mechanism requires the following information to work (not always concurrently) which is stored into encrypted separate base as shown in Figure 2: *(a)* The certified roles of the subject acquired by the attribute certificate, *(b)* the authorizations of the client defining which resources (s)he is allowed to access and under which mode, *(c)* the description of roles, *(d)* the role hierarchies (in Role-Based Access Control models, roles are organized into hierarchies. A role may participate into many hierarchies defining different protection domains), *(e)* the delegation certificates which are send to the access control mechanism in order to verify the validity of delegation, *(f)* the delegation rules in order to decide if the required delegation will be accepted and *(g)* the delegation hierarchy that defines the roles that a parent role can delegate its rights to.

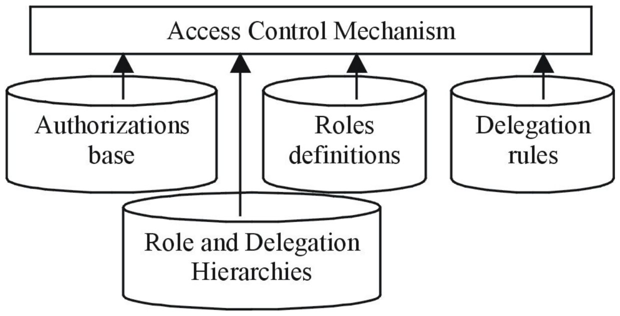

Figure 2: Co-operation between access control bases and mechanism

4 An XML-based Language

An XML-based language has been designed for supporting the proposed web-accessed environment. XML has all of the characteristics servicing our goals, like flexibility, extensibility and structure. *Document Type Definition* (DTD) has been adopted for expressing the format of all the access control issues. The reason for using DTD instead of XML Schema is its brevity and conciseness.

4.1 Specification of roles

The proposed model is an ultimately role-based access control one, and thus both objects and subjects are organized into roles. *Subject roles* are defined by the following DTD:

```
<!ELEMENT subject_role(name, scope+,
   (activation_cond, deactivation_cond, qualifications)?) >
<!ATTLIST subject_role id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT scope (#PCDATA)>
<!ELEMENT activation_cond (ANY)>
<!ATTLIST activation_cond type
                    (temproral | event_driven)>
<!ELEMENT deactivation_cond (ANY)>
<!ATTLIST deactivation_cond type
                    (temproral | event_driven)>
```

where every role is identified by a unique *id* and *name*. Moreover, each role has effect in a certain *scope* which is defined by the identity of a role hierarchy. A role may participate in various hierarchies. Furthermore, according to [8] a role may be activated and deactivated according to the satisfaction of certain conditions. Those conditions may be *temporal*, e.g. a role is activated in July 4th, or *event-driven* e.g. a role is deactivated in case another role is activated. Finally, *qualifications* define the characteristics that a subject should possess in order to be assigned the role.

The definition of *object roles* is quite simple since it is defined by the following DTD:

```
<!ELEMENT object_role (name, description)>
<!ATTLIST object_role id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

where *id* and *name* uniquely identifies the role and *description* may contain any information characterizing the role.

Both subject and object roles are organized into hierarchies where an authorization rule referring to a node may propagate to lower levels. This functionality limits the number of the needed authorization rules. Those hierarchies are stored into XML files which are based on the following DTD:

```
< !ELEMENT subject_hierarchy(node)>
< !ELEMENT node (node ?)>
< !ATTLIST node role_id IDREF>
```

A subject hierarchy is defined by a root node which may contain no or more nodes. That way a tree is defined. Delegation hierarchies are defined the same way and are stored into appropriate XML files.

4.2 Authorizations

The function of every access control system is based on the definition of authorizations. An authorization is a rule that generally defines which subject can access which object and under which action mode. Therefore, in our grammar authorizations are defined according to the following DTD:

```
<!ELEMENT authorization (subject_role, object,
            access_mode, provisional_action,
            environment_condition,)>
<!ATTLIST authorization id ID #REQUIRED>
<!ATTLIST authorization isdelegated (yes | no)>
<!ELEMENT subject_role (#PCDATA)>
<!ATTLIST subject_role role_id ID>
<!ELEMENT object (object_name|object_role)>
<!ELEMENT object_name (#PCDATA)>
<!ELEMENT object_role (#PCDATA)>
<!ELEMENT access_mode (#PCDATA)>
<!ELEMENT provisional_action (ANY)>
<!ELEMENT environment_condition (ANY)>
```

The first three elements (*subject_role, object* and *access_mode*) are the basic parts of an authorization rule. *Subject role* is identified by its name and its id which is taken by the XML files containing the definitions of roles. *Object* may be an independent resource or an object role. Thus, the tool can protect objects both independently and in groups. Since, metadata of protected resources are stored into XML files, in case of the independent protection, an XPath expression is used in order to identify the protected part of such a file. The *access mode* defines the operation the subject is allowed to perform over an object, e.g. execution, write, etc. The *provisional_action* element defines the action that should be performed before or after access is granted. The idea of provisional actions is described in detail in [7]. Examples of such actions may be log of session, sending an email to administrator, encrypting the protected target, etc. Finally, an authorization rule contains an *environmental condition* that should be satisfied for the rule to take effect. Those conditions may be temporal, events, etc. Our language is quite flexible since it can be extended and support the needs of various environments.

4.3 XML-based Attribute Certificates

The proposed framework is an XML-based access control tool, where the attributes certificates should be interpreted into XML-based ones in order to be comprehended by the access control mechanism. Those certificates contain the roles of a client. The DTD of those certificates will be the following :

```
<!ELEMENT attribute_certificate (issuer, licensee,
                              attribute+, valid_period)>
<!ELEMENT issuer (#PCDATA)>
<!ELEMENT licensee (#PCDATA)>
<!ELEMENT attribute (name, value)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT valid_period  (not_before, not_after)>
<!ELEMENT not_before  (date, time?)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT not_after  (date, time?)>
```

The components of an attribute certificate are: (a*) the issuer*, which contains the public key of the attribute authority, (b) *the licencee* who is the client possessing the certificate and (s)he is identified by his(her) public key, (c) *the attributes*, which may be multiple and they are characterized by their names and their values, and (e) *the valid period*, i.e. the element defining the life of the certificate. The valid period is limited by a *not_before* and *not_after* element which define the date and optionally the time that the life of the certificate begins and terminates.

4.4 Delegation Certificate and Delegation Rules

The environment, before accepting a delegation, it should be certain that the delegator truly delegated his(her) rights to the delegatee. Such an issue is satisfied through the use of delegation certificates. Therefore, a delegation certificate will have the following format (in a DTD):

```
<!ELEMENT delegation_certificate (delegator,
          delegatee+, scope+, permanence, monotonicity,
   totality, delegation_levels, activation, deactivation)>
<!ELEMENT delegator (#PCDATA)>
<!ELEMENT delegatee (#PCDATA)>
<!ELEMENT scope (#PCDATA)>
<!ELEMENT permanence empty>
<!ATTLIST permanence value (yes|no)>
<!ELEMENT monotonicity empty>
<!ATTLIST monotonicity value
            (monotonic/non_monotonic)>
<!ELEMENT totality (delegated_authorization?)>
<!ELEMENT delegated_authorization empty>
<!ATTLIST delegated_authorization id ID>
<!ELEMENT delegation_level empty>
<!ELEMENT delegation_leve times CDATA>
<!ELEMENT activation (ANY)>
<!ELEMENT deactivation (ANY)>
```

The *delegator* and the *delegatee* are identified by their role ids. The delegation may be multiple, i.e. delegator's authorizations may be inherited by various delegatees. Since the delegator role may participate into many hierarchies, the element *scope* defines the delegation hierarchies that this delegation takes effect. Element *permanence* may take two values, yes or no. In case of a permanent delegation the delegator cannot recall his(her) authorizations. In case *monotonicity* element has a value of monotonic in its attribute, both delegator and delegatee possess the delegated authorizations. The *totality* element contains the delegated authorizations which are identified by their unique id. The *delegation level* element defines if the authorizations can be further delegated and for how many times. Thus, attribute *times* may have a value of 0, in case further delegation is not accepted, or more. Finally, *activation* and *deactivation* elements may contain any condition that should be fulfilled for the delegation to take place or to be terminated respectively.

The access control mechanism has access to delegation rules stored in a separate base which have the same format as the certificate. Therefore, for a delegation to take place, there should be a rule consenting to the delegation certificate in all its parts. For example, if the certificate says that a delegation will take place in June 26th and the delegation rule defines that such delegations should not take place before July 1st the certificate will be declared.

5 Conclusions

We have introduced an attribute-based access control environment able to protect resources from known and unknown users. Since an access control mechanism needs some information about the subject, attribute certificates have been employed. Attribute Certificates contain the roles of a client and moreover verify their validity. Such certificates can be issued both by local and external Attribute Authorities. In the later case the system should interpret them into a format understandable by the access control mechanism. We have not yet implemented the interpreter but the completion of this task is among our future goals. The use of a standard for attribute certificates will help a lot our work.

XML is used for expressing roles, authorizations, rules, etc. Therefore, the access control mechanism cooperates with XML-bases containing all of the needed information in order to grant or deny access. Our goal is to build a functional and friendly environment implementing the proposed mechanism able to protect distributed web-accessed resources belonging to one organization.

6 Acknowledgments

The studies of the graduate student Konstantina E. Stoupa are funded by the State Scholarship's Foundation of Greece.

References

[1] E. Barka, and R. Sandhu, "Framework for Role-Based Delegation Models", Proc. 16th Annual Computer Security Applications Conference, pp. 168-176, December 2000.

[2] J. Dai and J. Alves-Foss, "Certificate Based Authorization Simulation System", Proc. 25th Annual International Computer Software and Applications Conference, pp. 190-195, October 2001.

[3] V. Doshi, A. Fayad, S. Jajodia and R. MacLean, " Using Attribute Certificates with Mobile Policies in Electronic Commerce Applications", Proc. 16th IEEE Annual Computer Security Applications Conference, pp. 298-307 , December 2000.

[4] C. Goh and A. Baldwin, " Towards a more Complete Model of Role", Proc. 3rd ACM Workshop on Role-Based Access, pp. 55-61, October 1998.

[5] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", Proc. Symposium on Security and Privacy, pp. 2-14, May 2000.

[6] ISO/IEC 9594-8:2001: Information Technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks". Available at: http://www.iso.org

[7] M. Kudo and S. Hada, " XML Document Security based on Provisional Authorization", Proc. of the 7th ACM Conf. on Computer and Communications Security, pp. 87-96, 2000.

[8] N. Li, B. N. Grosof and J. Feigenbaum, "Delegation Logic: A Logic-based Approach to Distributed Authorization*", ACM Trans. On Information and System Security*, Vol. 6, Issue 1, pp. 128-171, 2003.

[9] J. Linn and M. Nystrom "Attribute Certification: An Enabling Technology for Delegation and Role-Based Control in Distributed Environments", Proc. 4th ACM Workshop on Role-based access control, pp. 121-130, October 1999.

[10] Z. Miklós, "A Decentralized Authorization Mechanism for E-Business Applications", Proc. IEEE International Workshop on Trust and Privacy in Digital Business, pp. N/A, September 2002.

[11] M. J. Moyer and M. Ahamad, "Generalized Role-Based Access Control", Proc. IEEE 21st Int. Conference on Distributed Computing Systems, pp. 391-398, April 2001.

[12] J. S. Park and R. Sandhu, "Smart Certificates: Extending X.509 for Secure Attribute Services on the Web", Proc. 22nd National Information Systems Security Conference, pp. N/A, October 1999.

[13] Sandhu R. S., Coyne E. J. and Feinstein H. L., "Role-Based Access Control Models", IEEE Computer, pp.38-47, Feb. 1996.