The concept of a Distributed Repository for Validating X.509 Attribute Certificates in a Privilege Management Infrastructure

Adam Mihai Gergelya, Bogdan Crainicu

"Petru Maior" University of Tirgu-Mures, 1 N. Iorga st., Tirgu-Mures, 540088, Romania

Abstract

The present paper proposes a conceptual method for implementing a distributed repository for X.509 Attribute Certificates in a Privilege Management Infrastructure. It describes the idea behind this kind of repository, the security advantages of having a distributed repository, problems and access methods, and finally the techniques used to store and access the attribute certificates for validation by the requesting clients.

Keywords: X.509 Public Key Infrastructure; X.509 Attribute Certificate; Privilege Management Infrastructure; Distributed Repository.

1. Introduction

One of the most engaging aspects in managing distributed systems is the complexity of security management, mainly access control. The traditional Access Control List (ACL) cannot always be able to provide the desired level of security in large scale infrastructures. Many different access control techniques are available today, but Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC) prevail. In the case of RBAC, access privileges are allocated to roles rather than to individual users. Therefore, RBAC has the advantage that it can significantly simplify the management of access controls for large numbers of users.

ABAC, as a replacement for or adjunct to RBAC, performs decisions relying on attributes of requestors and resources, being more suitable for distributed systems. ABAC assigns attributes to users, which can be certified by

* Corresponding author. Tel.: +40-745-527032.
E-mail address: adam.gergely@it.upm.ro.
specific authorities and later verified against access control policies. ABAC uses labeled objects and user attributes instead of permissions to provide access control. We can say that a role in ABAC can be regarded as a group of mixed attributes rather than representing an unitary group of permissions as in RBAC.

The Privilege Management Infrastructure (PMI) is a concept that manages user authorizations by using the ITU-T X.509 Attribute Certificates (ACs). This was introduced in the 2001 version of X.509, and in 2005 a delegation service was added to improve the PMI. In 2009, an interdomain authorization was also added to enhance the current version of PMI.

Using an analogy, the concept of PMI with authorization mirrors the concept of Public Key Infrastructure (PKI) with authentication. While PKI uses Public-Key Certificates (PKCs) to prove the identity of a certificate subject, PMI uses ACs to check and validate authorization data to a certificate subject for a given resource.

However, PMI is rather a subcomponent of a PKI, because PMI still relies on PKI for validating signatures of the Attribute Authorities (AAs). Basically, PMI is used to provide the necessary framework to effectively make use of the attribute certificates for proper privilege authorizations. PKI and PMI infrastructures are related by information contained in the identity and attribute certificates.

Based on PMI ACs, the paper proposes a new conceptual method that deals with deployment of a distributed repository suitable for validating the ACs by a requesting entity in a large scale networked infrastructure, where a PMI context is involved.

2. PMI Model

Although ACs were first defined in X.509(97), it was not until the 4th edition of X.509 (ISO 9594-8:2001) [3] that a full infrastructure for the use of attribute certificates was defined. This infrastructure is termed a Privilege Management Infrastructure (PMI), and it enables privileges to be allocated, delegated, revoked and withdrawn in an electronic way. A PMI is to authorization what a Public Key Infrastructure (PKI) is to authentication [1].

PMI entails Sources of Authority (SoA) as trusted root and Attribute Authority (AA) as ACs’ issuer. They do not use Certification Authorities (CAs) like PKI does. However, there is a relationship between PKI and PMI, because the PMIs are based on PKIs, for example, when attribute certificates need to be digitally signed by the AA that issued them. At this point, when the signature of the AA needs to be verified, PKI is used.

A PMI consists of 5 types of components [3]:

- Attribute Authorities (AAs) (also called Attribute Certificate Issuer) - to issue and revoke ACs.
- Attribute Certificate Users - to parse or process an AC.
- Attribute Certificate Verifier - to check the validity of an AC and then make use of the result
- Client - to request an action for which authorization checks are to be made
- Repositories - to store and make available certificates and Certificate Revocation Lists (CRLs).

An AC (or Authorization Certificate) is a digital document which contains attributes issued by the AA to a holder (which is the subject of the attribute certificate). When the attributes are used to provide authorization to a holder, that attribute certificate essentially becomes an authorization certificate. An AC does not contain a public key because the issuer of the AC has the Attribute Certificate Verifier under its control. No authentication is required here for the attribute certificate. The authentication is, however, verified with the public-key certificate.

An AC references a public-key certificate which is the digital identity of the holder of both the public-key certificate and the attribute certificate. Further more, a role within an attribute certificate can be further delegated to other entities, effectively creating an AC chain, but this procedure is not recommended by RFC 3281 because of the complexity of chain processing and administration [2].

ACs may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements [2].

Fig. 1 describes the elements of PMI [4]. There are two types of attribute certificate distribution, push and pull [4]. In some environments it is suitable for a client to push an AC to a server. This means that no new connections
between the client and server are required. It also means that no search burden is imposed on servers, which improves performance [4].

![Fig. 1. Elements of PMI [4]](image)

In other cases, it is more suitable for a client simply to authenticate to the server and for the server to request or pull the client’s AC from an AC issuer or a repository. A major benefit of the pull model is that it can be implemented without changes to the client or to the client–server protocol. It is also more suitable for some inter-domain cases where the client’s rights should be assigned within the server’s domain, rather than within the client’s domain [4].

An AC comprises a digitally signed SEQUENCE of [1]:

- the version number of this AC (v1 or v2).
- identification of the holder of this AC.
- identification of the AA issuing this AC.
- the identifier of the algorithm used to sign this AC
- the unique serial number of this AC.
- the validity period of this AC.
- the sequence of attributes being bound to the holder.
- any optional extensions.

Table 1. presents a comprehensive comparison of PKI and PMI [1].

<table>
<thead>
<tr>
<th>Concept</th>
<th>PKI entity</th>
<th>PMI entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate</td>
<td>Public Key Certificate</td>
<td>Attribute Certificate</td>
</tr>
<tr>
<td>Certificate issuer</td>
<td>Certification Authority</td>
<td>Attribute Authority</td>
</tr>
<tr>
<td>Certificate user</td>
<td>Subject</td>
<td>Holder</td>
</tr>
<tr>
<td>Certificate binding</td>
<td>Subject's Name to Public Key</td>
<td>Holder's Name to Privilege Attribute(s)</td>
</tr>
<tr>
<td>Revocation</td>
<td>Certificate Revocation List</td>
<td>Attribute Certificate</td>
</tr>
<tr>
<td>Revocation List</td>
<td></td>
<td>Revocation List</td>
</tr>
<tr>
<td>Root of trust</td>
<td>Root CA or Trust Anchor</td>
<td>Source of Authority</td>
</tr>
<tr>
<td>Subordinate authority</td>
<td>Subordinate CA</td>
<td>Attribute Authority</td>
</tr>
</tbody>
</table>
Extensions to X.509 ACs are concerned with specifying some aspects of policy that govern the use and applicability of the ACs. The extensions can be split up into five categories [1]:

- basic extensions.
- privilege revocation extensions.
- an extension to support roles.
- source of authority extensions.
- delegation of authority extensions.

Details about X.509 AC Extensions are provided in [1].

3. Implementing a new distributed repository schema for X.509 Attribute Certificate

A more efficient way to authenticate a client and check its privilege levels is to inspect its Authority Certificate (AC) via a repository, rather than accepting the AC pushed by the client. Security advantages behind this method can be summarized as follows:

- The requesting entity which requests validation can interrogate a trustworthy database and not rely on any real or altered information pushed by the client.
- Access to the repository may be secured, and the effectiveness increases if there is a proper standard way to interrogate these attribute certificates.
- It simplifies the process of authorization because the user does not need to explicitly tell the requesting entity what are its privileges. The client can have its privileges looked-up in a database and automatically extract its privileges from there.

If this method will become the dominant method for interrogating the privileges of a user, it means that the databases which hold these attribute certificates must be of considerable size and access to them must be of a top quality. In today's computer networking, only a distributed system can offer high volume transaction efficiency when serving thousands of requests per second. But the selection of specific technologies on distributed systems is covered by implementation details and it is not the subject of the paper.

Fig. 2 shows the relationships between the AA, Distributed Repository, AC Holder and AC Verifier.

Fig. 2 The new proposed Distributed Repository for x.509 Attribute Certificates
The proposed architecture components for integration of the ACs within a distributed repository are:

- One or multiple SQL / Oracle databases across several servers with permanent replication system.
- Separate authorization servers and credentials servers.
- Several pre-cached permissions of users on response servers only (like the DNS cache system).

The corresponding high-level workflows for interrogating an AC for a particular user would be:

- The Client connects to the Server (which is also the validating entity).
- The Server queries the Distributed Database for the AC of the connecting client, after verifying its Public-Key Certificate.
- The Distributed Database replies to the Server with the authorization credentials for that particular user.
- The Server acknowledges and registers the Client’s permissions.

4. Conclusions

The distributed repository represents a more efficient solution to store and access the attribute certificates for two main reasons:

- The internal management of the ACs is more robust and more secure, because these ACs and their backups are distributed evenly on more systems, which causes redundancy, scalability and fault tolerance.
- The external user, this “centralized” (which in fact, is a distributed) system of accessing the ACs directly from a trusted authority is more efficient than having the user present its AC.

Despite the fact that maintaining a distributed repository involves more cost, more equipment dedicated for this purpose, more maintenance and workload, in the long run it offers a more reliable and trustworthy service, especially in this critical area of real-life digital security domain. The paper advances just a conceptual, theoretical PMI-based method that can provide the basis for building real technological solutions for implementing a distributed repository for X.509 ACs.

References