Cross-Domain Access Control via PKI *

Grit Denker and Jon Millen
Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA
{denker,millen}@csl.sri.com

Yutaka Miyake
KDD R&D Laboratories Inc., Kamifukuoka-shi, 356-8502, JAPAN
miyake@kddlabs.co.jp

Abstract

In this note we consider how role-based access control can be managed on a large scale over the Internet and across organizational boundaries. We take a PKI approach, in which users are identified using public key certificates, as are the servers. The main features of our approach are: access control by (client, role) pair; implied revocation based on the role hierarchy; automatic generation of certificate validity tickets; and certificate chains to prove a client role hierarchy to a server.

1 Introduction

The Internet, with Web browsers and Web servers, makes it convenient for anyone to access publicly available information on servers. Some servers provide private or sensitive information and require users to log in with passwords. This form of access control is suitable if users must be distinguished from one another, because the information is personal, or because usage accounts are set up individually.

In other cases a service is restricted, but permission is given on the basis of the role of the user, and not the individual user identity. This is particularly useful if the user and service belong to different organizations. An example might be that all members of a certain university department have access to the online services of a particular library at a different university. Another is that all employees in a company's retirement program may have access to the external investment company's performance summary.

In this note we consider how role-based access control can be managed on a large scale over the Internet and across organizational boundaries. For the purposes of this note, we assume the basic features and terminology of RBAC (Role-Based Access Control, [5]) as an underlying policy model, so that we can focus on the cross-domain issues. We take a PKI approach, in which users and their affiliations are identified using public key certificates, as are the servers. The use of "smart certificates" to identify users and their roles was described in [4]. We use similar certificates, but the cross-domain issues and other objectives lead us to a different way of using them: we have two kinds of certificates, and they are content-chained to reflect the role hierarchy. Timely revocation checking is also handled in a special way.

One of the challenges of large-scale inter-organizational PKI is that different organizations may be in the domain of different certification authorities (CAs). An organizational CA might have a certificate signed by a commercial CA (VeriSign, Thawte, etc.), but we will still regard it as the certificate issuing authority for its own organizational domain. Organizations are mutually suspicious in general, but may establish relationships with chosen partner organizations to support client-server activity.

Different organizations recognize each other by means of *cross-certificates*, in which one organization's identity, trust status, and CA public key information is signed by the other organization's CA. We will make use of cross certificates, but there are several issues that are not solved simply by the existence of such certificates.

1.1 Objectives

A server organization may deal with many different client organizations, and *vice versa*. The total number of users of a server may be so large, and changes in it so frequent, that an individual user identification database is not practical. This has several implications:

- The server uses the role, rather than the individual user identity, to control access.

* Supported by KDD Laboratories, Inc.

- User membership and role in a client organization is under the control of the client domain, and the server must trust the client organization to assign certificates accordingly.

A *client* domain issues certificates to users (clients) giving them a public key and one or more roles. A *server* domain contains one or more Web servers that may be accessible to users in other domains. The same domain may be both a client and server domain. Note that a standard X.509 certificate has an *issuer* field identifying the signer of a user certificate, in our case the client domain.

Server and client agree on certain role names (in the context of that client) as part of a prior administrative agreement. However, a server cannot be expected to keep track of detailed organizational changes and role hierarchies in client organizations. The client therefore has to provide proof to the server that:

- the user is still a member of the client organization in some designated role,
- the role needed for access is still recognized in the client organization, and
- the role hierarchy linking the user's certified role with the role needed for access permission is still intact.

Certificates can be revoked before their expiration date. For each certificate used to establish a user-role connection, there must be a way for the client to generate a short-term "ticket" either taking the place of the original certificate, or asserting that the original certificate has not recently been revoked. Real-time computation of digital signatures is burdensome, and should be avoided in high-traffic applications. However, hash-chain proofs can be used to obviate real-time signatures.

Our approach is presented conceptually in Section 2. This approach has been partially implemented in a prototype system called XDA (cross-domain access), which is described in Section 3. Briefly, the XDA system is an extension of Web page access using an ordinary SSL-capable browser and Web server. A client-side proxy monitors Web page requests and contacts the client CA to obtain tickets to send to the server. The CA module that performs the status response function is called the Privilege Granting Server (PGS). The prototype does not yet handle certificate chains, but can be extended easily to do so.

2 Certificates, Revocation, and Access

In the RBAC model, permissions are given to roles, where a permission approves access in a designated mode to one or more objects. A Web server associates permissions to roles with some form of access control list (ACL). The ACL is in a directory containing the pages or services it controls, and each ACL entry gives some mode of access to a specified user or role. Thus, an ACL entry can be regarded as a *(role, permission)* pair.

In our system, to support cross-domain access, the role is given a structure: *(client-organization, local-role)*, where the local role name is meaningful for the client organization, and different roles may be used in ACL entries for different organizations, for access to the same Web page.

2.1 Role Hierarchy and Certificate Chaining

In a role hierarchy, if role r' is a subordinate of role r, written $r \geq r'$, then any permissions associated with role r' are inherited by role r. Thus, if a user has a user-role certificate showing membership in role r, and a Web page requires role r', the user should be able to get permission.

We use two kinds of certificates to validate user role permissions. *User-role* certificates are used to prove that a user is a member of the client organization, give the user a public key, and specify a most-specific role for that user. *Role hierarchy* certificates encode direct subordination relationships among roles.

Normally, the user would begin with a user-role certificate for her most-privileged role, say r. Rather than have the client organization store or generate additional certificates for this user for all subordinate roles $r \geq r'$, we establish the relation $r \geq r'$ with a chain of role hierarchy certificates indicating $r \geq r_1, r_1 \geq r_2, ..., r_n \geq r'$. This set of certificates is sent by the client to the server, and the server verifies that this is a continuous chain linking the user to the required role, and (as discussed below) that all certificates in the chain are still valid.

In our cross-domain environment, having the server determine the relationship $r \geq r'$ from its own stored account information, as in [4], would not be consistent with our assumption that the server should not have to keep track of all client hierarchies.

The two kinds of certificates are shown in Figure 1 (not showing certificate fields such as validation period and other standard fields of public-key certificates). Some role hierarchy certificates have no sub-role. This is indicated by some constant value in the sub-role field. This kind of certificate is an *anchor* certificate and its role is an anchor role. Ordinary public key certificates and cross certificates do not have a role field.

The proposed certificate formats fit well with the X.509 PKI public-key certificates and PMI attribute certificates [1]. The "role" field of a role certificate and the "sub-role" field of a hierarchy certificate can be defined as extension fields within the X.509 standard.

Note that role hierarchy certificates are *content chained* rather than *signature chained*. Public key certificates are nor-

User Role Certificate

User	Role	PK	Issuer	sig

Role Hierarchy Certificate

Role	Sub Role	noPK	Issuer	sig

Figure 1. Role and hierarchy certificates

mally signature chained when the issuer CA is not known directly and its public key must be obtained from a another certificate. Thus, linking is on the subject-issuer relation, with a different key for each signature. In our case, linking is on the user-role and role-sub-role relations, with the same client CA key for each signature, except for the final cross-certificate.

2.2 Revocation and Access

Content chaining of certificates has advantages when it comes to revocation and access control. Both user-role and role-hierarchy relationships can be revoked. We assume an environment with *implied revocation*. In such an environment, there is a constraint saying that if a role r is eliminated, then *all more-privileged roles* must also be eliminated, unless their existence is preserved by an alternate path through the hierarchy in the direction of lower privilege to an anchor role.

For example, if a company division is eliminated, all of its departments are thereby eliminated. If a department is eliminated, the position of division librarian, who serves all the departments, is (normally) not eliminated, since there is an alternate path from that position through another department.

Thus, if $r_1 \geq r_2 \geq r_3$, and r_2 is eliminated, then permissions requiring r_3 still make sense but permissions requiring r_2 do not. Permissions requiring r_1 are still possible if r_1 is a super-role of some other less-privileged role with a path to an anchor role.

Eventually, the client domain administrator should tell each affected server domain administrator that ACL entries naming r_2 should be removed. However, there ought to be a way to prevent these accesses in the short term, after the revocation in the client domain but before the administrator has contacted the affected server domains, and this should happen as an immediate consequence of the revocation of role hierarchy certificates with role r_2.

Two alternative revocation policies are suggested to achieve this, with a different tradeoff of client work and server work: *full* revocation and *partial* revocation, explained below. We assume that client and server establish, at the beginning of their cooperation, which policy they follow, or that the policy specification is part of the exchange protocol or certificate format.

In both policies, the server must check the validity of all certificates in a submitted chain. The full-revocation policy requires that all affected certificates be revoked. In that case, checking a certificate chain up to the role required for access is sufficient. The partial-revocation policy saves some client revocations but requires a longer chain, extending beyond the required role to an anchor role. In this case, the role required for access can be anywhere in the chain.

In the partial revocation policy, to revoke a role, it suffices to explicitly revoke the role hierarchy certificates naming its subroles. This way, no valid certificate chain, starting from a user-role certificate of a user down to an anchor role can be established. If a role is eliminated by implication because some descendant role has been eliminated directly, some certificate in an anchored chain must have been revoked.

One might ask the following: if certificates have been revoked, and the client domain is trusted, why can't we simply assume that the client will refuse to send any revoked certificates? Then the server will not have to check their validity. The security threat here is that an attacker or malicious individual user (whose certificate has been revoked) might copy, save, and replay revoked certificates, and continue to use them until the nominal expiration date.

2.3 Certificate Status Proofs

The client domain, specifically the PGS, finds a valid certificate chain to accompany an access request. The PGS can then create and sign a ticket guaranteeing that the required role is valid for the user. However, it was observed earlier that it is a burden for a client domain to perform signatures in real time. Instead, hash chain proofs can be used as they have been for Certificate Revocation Trees (CRTs) [2]. The idea is that the CA maintains a certificate revocation list (CRL), which is authenticated each time it is modified by the CA. The CA signature is attached to a tree of hash values with the property that a relatively short chain of hash values, together with the single previously computed signature, can validate the absence of a certificate (by serial number) from the CRL. A proof for each certificate in the chain is required.

3 Experimental Implementation

An important objective in the design of a prototype for the cross-domain access control (XDA) system was to demonstrate how the essential features of the XDA architecture can be implemented in a practical way using current technology.

The implementation was designed for an environment in which the server is a Web server using the HTTP and SSL protocols and the client uses a commonly available Web browser. The certificate authority was in an environment that we could augment with a trusted PGS module.

XDA provides additional specialized software modules: the proxy, Privilege Granting Server (PGS), and authorization server. These three new software components together with the main steps of the status request and its relationship to the SSL protocol are outlined in Figure 2. The data flow for the ticket occurs in parallel with the SSL protocol.

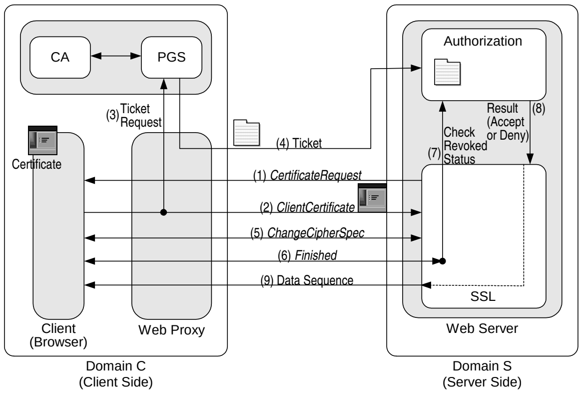

Figure 2. SSL Monitoring

The proxy monitors connection requests from the browser to Web servers. When the proxy sees that a secure connection is being set up with a server, using a user role certificate, the proxy automatically requests the PGS for a status validation.

A response from the PGS is, in general, either a freshly signed ticket or a certificate chain together with hash chain proofs for non-revocation. Our prototype PGS generates tickets consistent with OCSP (Online Certificate Status Protocol), which has been proposed as a standard for conveying the revocation status of certificates [3]. The OCSP response is forwarded by the proxy to the authorization server. Note that the ticket does not include the user public key or role; that information is conveyed separately in the user role certificate.

In the XDA prototype implementation we made use of the publicly available Tomcat Web server software, which supports Java servlets for special functions performed as part of Web page access. XDA servlets are associated with XDA-protected Web pages. In the prototype XDA system, the access control list is a file in the directory of the controlled Web page, listing the privilege values for which access is permitted. This mechanism is very much like the ".htaccess" file format used in UNIX-based Web servers, such as Apache.

The authorization server is a servlet that accepts tickets from the proxy, and it also handles requests from an XDA servlet associated with the Web page to confirm that a valid ticket has arrived for a given certificate.

4 Conclusion

Our cross-domain access (XDA) approach makes the greatest possible use of modern PKI concepts and standards to support scalable access control. A server ACL needs to have only one or a few entries for each remote client domain, since access control is by a role whose actual user membership is managed by the client domain.

We use cross-certificates to support client-domain relationships, user-role certificates to identify users, and role hierarchy certificates to define the hierarchy in a way that can be transmitted to server domains that are otherwise unaware of the hierarchy.

The server can check revocation of role assignments by checking the validity of certificates on a chain linking the user to the required role. The partial revocation strategy requires longer chains but permits implied revocation of roles. Using hash chain proofs, the client can avoid real-time computation of digital signatures.

This approach requires only a few customized software functions to be added in a modular way to client and server support.

References

[1] ITU-T. Draft revised ITU-T Recommendation X.509. ISO/IEC 9594-8: Information Technology – Open Systems Interconnection – The Directory: Public-Key and Attribute Certificate Frameworks, 2000. `ftp://ftp.bull.com/pub/OSIdirectory/4thEditionTexts/X.509_4thEditionDra%ftV2.pdf`.

[2] P. Kocher. On Certificate Revocation and Validation. In *Financial Cryptography*, volume LNCS 1465, pages 172–177, 1998.

[3] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure: Onlice Certificate Status Protocol - OCSP. Technical Report RFC2560, IETF, June 1999.

[4] J. Park and R. Sandhu. RBAC on the Web by smart certificates. In *ACM Workshop on Role-Based Access Control*, 1999.

[5] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 9(2):38–47, 1996.