AMBAR Protocol: Access Management Based on Authorization Reduction *

Oscar Cánovas[1], Antonio F. Gómez[2]

[1] Department of Computer Engineering
[2] Department of Information and Communications Engineering
University of Murcia, Spain
ocanovas@ditec.um.es, skarmeta@dif.um.es

Abstract In the last years, SPKI, X.509 attribute certificates, or KeyNote has been proposed as mechanisms to create and specify authorization certificates, access control lists, or security policies in distributed environments. In this work we propose a new protocol able to negotiate and use some of these specifications. AMBAR is a multi-layered protocol based on a request/response model. In general, it provides functionality to transmit resource access requests, the authorization information related to those requests (credentials, ACLs), and results obtained from a certificate chain discovery method or compliance checker. It adds security by acting as a separate security layer inserted between the higher protocols and TCP (or another different transport protocol).

1 Introduction

Public key cryptography is widely recognized as being a fundamental technology on which several essential security services can be built. The Internet community is agreeing on the use of systems based on the X.509 standard [10] and the SSL protocol [2] in order to provide basic security services to e-commerce. In recent years, public key cryptography has been also proposed as a tool for solving the problems related to authorization and access control. SPKI/SDSI [8] and KeyNote [4] propose mechanisms for capturing security-relevant information and binding authorization data to public keys. Recently, the PKIX Working Group published a specification [9] defining the X.509 Attribute Certificates (AC) profile. However, most of the current security protocols do not provide any mechanism to negotiate, transmit, or process data related to authorization certificates or security policies.

In this paper, we propose a new access control protocol able to negotiate and to use authorizations based on public key cryptography. AMBAR (Access Control Based on Authorization Reduction) does not depend on a particular type of authorization or identity-based certificate, and it contains a negotiation phase designed to adapt the protocol to access control scenarios with different requirements (anonymity, confidentiality, credential recovery, etc.). In general,

* Partially supported by TEL-IFD97-1426 EU FEDER project (PISCIS)

it provides functionality to transmit resource access requests, the authorization information related to those requests (credentials, ACLs), and results obtained from a certificate chain discovery method or compliance checker.

2 Protocol requirements

We consider that the access control protocol must accomplish three main goals. First, it must be independent of applications or higher protocols, i.e., it must support any application-specific authorization, policy or request. Second, it must be able to operate with different identity-based infrastructures and authorization systems. Finally, access requests must be managed efficiently with the purpose of obtaining a good response time.

We can find in the literature some access control systems using authorizations [11]. In general, these systems process requests individually, i.e., there is not an implicit concept of protocol session, and therefore every request is transmitted together with the related credentials, ACLs, authorization decisions, etc. This situation is specially problematic when the communication is performed between the same client and server, since most of the exchanged information has been previously transmitted, and some calculations have already been computed. We consider that these protocols should be session-oriented, and they should keep a local cache of the information exchanged in a particular session in order to avoid unnecessary calculations and communications.

Next, we state all the requirements for the protocol. We also include some additional requirements not commented above.

1. The protocol must be able to negotiate which type of identity and authorization certificates will be used.
2. It should offer confidentiality services to protect the transmitted data.
3. The protocol must allow anonymous access to preserve user identity. Additionally, an identified access mode must be implemented too.
4. It must support several credentials distribution methods. In some scenarios, it will be suitable for a client to "push" authorizations to a server, which improves server performance. In other cases, it will be more suitable for a server to request or "pull" the credentials from an issuer or repository.
5. The protocol must provide a method for establishing authorized data streams between clients and servers. Higher level protocols should layer on top this protocol transparently.
6. The design must be modular in order to easily add further functionality.

3 AMBAR Overview

As we will see in this section, we have chosen to create an entirely new protocol layer for authorization. The design has been performed regarding some prudent engineering practices exposed in [1,3].

The AMBAR protocol consists of different components organized, as Figure 1 illustrates, in two layers.

Figure 1. AMBAR Architecture

- **Session Management module (SM).** This module transmits the client and server security preferences, and generates the cryptographic data used by the TC layer to protect the subsequent communications (if confidentiality was negotiated). Clients and servers negotiate the following parameters:
  - *Symmetric cipher.* Parties select the symmetric cipher and its key length.
  - *Operation mode.* AMBAR supports two operation modes: anonymous client mode and fully identified.
  - *Identity-based certificates.* It is possible to select X.509, OpenPGP [5], or SDSI certificates.
  - *Authorization-based certificates.* AMBAR supports SPKI certificates, PKIX attribute certificates and KeyNote asserts.
  - *Credentials distribution.* Parties can select whether the credentials will be provided by the client (push), or will be obtained by the server from either a repository or an issuer (pull).
- **Request Management module (RM).** The RM module transmits two types of messages: messages related to authorization requests and credentials; and messages related to decisions and ACLs. Contents and the sequence of these messages are determined by the negotiated operation mode and the method for distribution of credentials. As we mentioned previously, a session-oriented protocol lets the ability to perform some optimizations. Therefore, the RM module could be responsible for optimizing access control computations.
- **Authorization Results Management module (ARM).** The ARM module generates notifications and transmits the demanded resources. Negative notifications are transmitted by the server when the access is denied. If the access were granted, there would be two possible response messages: an affirmative notification if the client requested the execution of remote actions; or the controlled resource. It also enables (disables) the DSM module when an authorization request demanding the establishment (conclusion) of a data stream is granted.
- **Error Management module (EM).** Systems use the EM module to signal an error or caution condition to the other party in their communication. The EM module transmits a severity level and an error description.
- **Data Stream Management module (DSM).** The described request/response model is not suitable if we plan to use AMBAR as a transparent layer providing confidentiality, authentication and access control services. The

DSM module, initially disabled, controls the transmission of arbitrary data streams, which are enabled once a request demanding the activation of this module is granted.

– **Transport Convergence module (TC).** The TC module provides a common format to frame SM, RM, ARM, EM, and DSM messages. This module takes the messages to be transmitted, authenticates the contents, then applies the agreed symmetric cipher (always a block-cipher), and encapsulates the results. The cryptographic data used to protect the information is computed by the SM module during the negotiation phase.

4 Some details of the protocol

In order to show some details of the messages related to the request/response phase, we will analyze in this section the *push* distribution method using its typical message sequence. Negotiation phase has been omitted due to the lack of space (more information about AMBAR can be found in [6]). Therefore, we will assume that both client and server have already negotiated cryptographic preferences and operation modes. The employed notation is described through the explanation of the messages. We will consider a *transaction* as the different messages related to a specific authorization request, and a *session* as the sequence of different transactions.

In a session based on the *push* method, clients calculate the authorization proof after receiving the ACL controlling the resource from servers.

1	**Request**	$C \Rightarrow S\ \{T_{ID}, T_{Step}, SFlag, Request, [Asserts]^{0..N}\}^{k_{MAC}}_{k_{SYMM_s}}$
2	**ACL**	$S \Rightarrow C\ \{T_{ID}, T_{Step}, ACL\}^{k_{MAC}}_{k_{SYMM_c}}$
3	**Calculation**	$C \Rightarrow S\ \{T_{ID}, T_{Step}, Calculation\}^{k_{MAC}}_{k_{SYMM_s}}$
4	**Neg_Notification**	$S \Rightarrow C\ \{T_{ID}, T_{Step}, Notification\}^{k_{MAC}}_{k_{SYMM_c}}$
4	**Aff_Notification**	$S \Rightarrow C\ \{T_{ID}, Notification\}^{k_{MAC}}_{k_{SYMM_c}}$
4	**Resource**	$S \Rightarrow C\ \{T_{ID}, Resource\}^{k_{MAC}}_{k_{SYMM_c}}$

The *Request* message, generated by the RM module, represents the authorization request formulated by the client. It contains an identifier of transaction T_{ID}, a transaction step identifier T_{Step}, a flag indicating whether this is a request for a data stream ($SFlag$), a set of asserts or authorizations related to the request, and the authorization request. Data are ciphered using K_{SYMM_S}, and are authenticated with K_{MAC} (derived during the SM phase). All messages analyzed in this section will be protected in the same way.

The server response, generated by the RM module, is the ACL message. It contains the ACL protecting the resource, the same T_{ID} included in the request, and an incremented transaction step identifier T_{Step}.

Once the client receives the ACL, it creates a certificate chain from its public key to the ACL entry related to the resource. That chain may be composed by authorization and ID certificates, and it is the output of the certificate chain discovery method (or trust management engine). The client-side RM module sends that result to the server in the *Calculation* message.

The final step is the server response to the calculation. If the calculation were wrong, the server would send a *Neg_Notification* message. That message includes the error description ($Notification$), an incremented transaction step T_{Step}, and the T_{ID} identifier (included in all the ARM messages). On the other hand, when the server validates the request, it returns the *Resource* message (when the requested resource is a file, document, etc.) or an *Aff_Notification* message (if the request is for a remote action).

5 Conclusions

We have introduced AMBAR as a new protocol able to negotiate and to use some of the proposed specifications for distributed authorization architectures. It proposes a message format for transmitting authorization information, and it has been designed session-oriented in order to optimize the way the authorization decisions are made (saving unnecessary calculations and transmissions). AMBAR does not depend on a particular type of authorization or identity-based certificate, and it can be easily extended to support future proposals. Currently, AMBAR has been implemented in C++ and it is being tested using authorization certificates based on SPKI [7].

References

1. M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 1(22):6–15, January 1996.
2. A. O. Alan, P. Freier, and P. C. Kocher. *The SSL Protocol Version 3.0*, 1996. Internet Draft.
3. R. Anderson and R. Needham. Robustness principles for public key protocols. Number 963 in Lecture Notes in Computer Science. Springer, 1995.
4. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The KeyNote Trust Management System Version 2*, September 1999. Request For Comments (RFC) 2704.
5. J. Callas, L. Donnerhacke, H. Finney, and R.Thayer. *OpenPGP Message Format*, 1998. Request For Comments (RFC) 2440.
6. O. Canovas and A. F. Gomez. AMBAR Protocol: Access Management Based on Authorization Reduction. Technical report, University of Murcia, May 2001. UM-DITEC-2001-7.
7. Intel Corporation. *Common Data Security Architecture (CDSA)*. World Wide Web, http://developer.intel.com/ial/security, 2001.
8. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.
9. S. Farrel and R. Housley. *An Internet Attribute Certificate Profile for Authorization*. Internet Engineering Task Force, 2001. draft-ietf-pkix-ac509prof-06.
10. R. Housley, W. Ford, and D. Solo. *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, January 1999. Request for Comments (RFC) 2459.
11. A. Maywah. An implementation of a secure web client using SPKI/SDSI certificates. Master's thesis, M.I.T., May 2000.