Transport Layer Security Working Group
INTERNET-DRAFT
Expires Feb. 20, 1999.

S. Farrell
SSE

August 20, 1998

TLS extensions for AttributeCertificate based authorization

<draft-ietf-tls-attr-cert-01.txt>

Status of this memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This document describes extensions to [TLS] providing authorization support based on the use of X.509 AttributeCertificates.

1. Introduction

TLS provides authentication, data integrity and confidentiality services for Internet protocols. In many applications these services are not sufficient to provide the type of authorization services required.

For example, it may be that access to a resource should be controlled on the basis of the client's clearance or role.

An additional requirement is support for delegation, where a intermediate TLS server acts as a client but

where the final target TLS server must make its access decision on the basis of the initial client's rights.

While TLS doesn't provide support for these features, AttributeCertificates (ACs) in conjunction with TLS do provide a solution. [AC509] defines a profile of the X.509 AttributeCertificate which can be used in this context.

In the remainder of this document we describe the operational models, then the new messages and data structures and finally describe a method for embedding AC exchange into the TLS handshake.

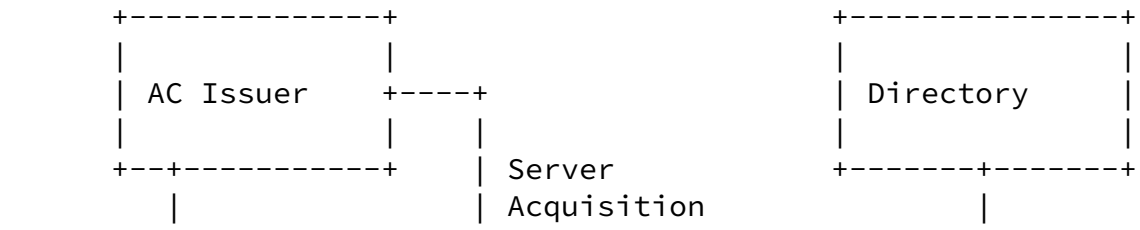
2. Operational Models

In some environments it is suitable for the client to "push" an AC to a server. This means that no new connections between the client and server domains are required. It also means that no search burden is imposed on servers, which improves performance.

In other cases it is more suitable for the client simply to authenticate to the server and for the server to request ("pull") the client's AC from the AC issuer or a repository. A major benefit of the "pull" model is that it can be implemented without changes to the client. It is also more suitable for some extranet cases, where the client's rights should be assigned within the server's domain.

There are a number of possible exchanges which can occur and three entities involved (client, server and AC issuer). In addition the use of a directory service as a repository for AC retrieval may be supported.

The diagram below shows the exchanges defined which are described in later sections.



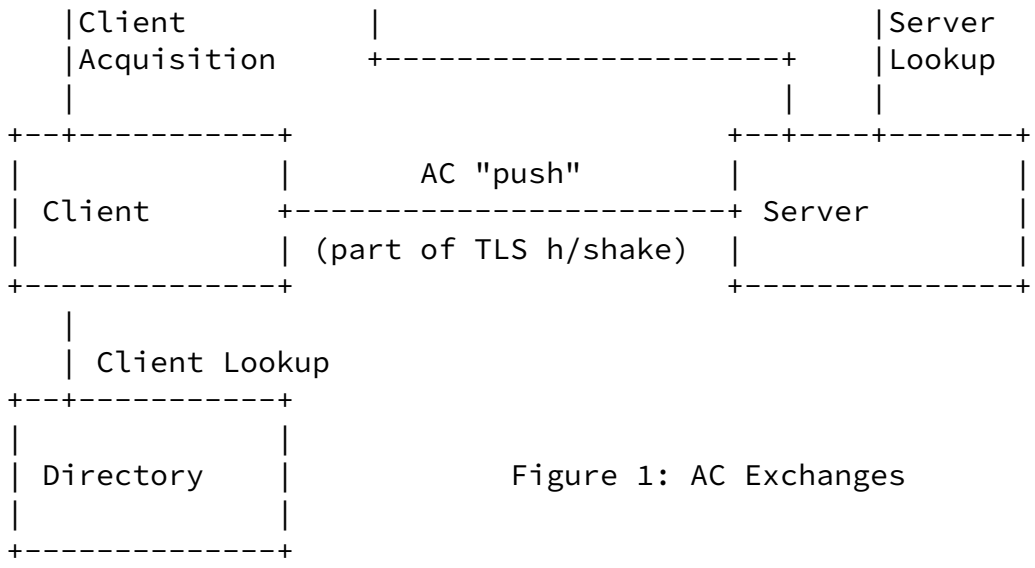


Figure 1: AC Exchanges

Of the above exchanges, the most important to embed in the TLS protocol is that between client and server. AC acquisition (from an issuer or directory) is handled as a higher layer protocol (a payload protocol from the TLS perspective).

An AC carrier structure (acinfo) is defined which allows for requests for ACs, responses to same and for pushing ACs plus external controls in a standard manner. This structure is used in all messages.

3. Data Structures

acinfo is the data structure which is used to "push" or "pull" an AC and/or associated control information. acinfo may also be used when requesting an AC from an issuer.

A set of public key certificates may also be "pushed" with an acinfo in order to assist the recipient in certificate handling.

```
ACInfo ::= SEQUENCE {
     type      INTEGER {
                    clientAcquire       (0),
                    clientAcquireResp   (1),
                    serverAcquire       (2),
                    serverAcquireResp   (3),
                    acRequest           (4),
                    acResponse          (5)
               },
```

```
            peer       [0] GeneralName     OPTIONAL,
            template   [1] SEQUENCE OF Attribute OPTIONAL,
                 -- only one occurrence of each attr type
            ac         [2] AttributeCertificate OPTIONAL,
            pps        [3] PairingProofs   OPTIONAL,
            certs      [4] Certificates    OPTIONAL
       }
```

When an intermediate server (IS) delegates an AC to a target server (TS) then the target (TS) must be able to verify that the intermediate (IS) hasn't "stolen" the AC. This is achieved by having the IS produce the following signed ASN.1 structure:

```
       PairingProof ::= SIGNED SEQUENCE {
            init       GeneralName,
            target     GeneralName,
            issuer     GeneralName,
            serial     INTEGER,
            time       GeneralizedTime,
            nonce      BIT STRING,
            sigalg     AlgorithmIdentifier
       }
```

If traced delegation is required (ensuring that no entity in the chain stole the AC) then a sequence of pairing proofs must be sent (with the current one left most):

```
       PairingProofs ::= SEQUENCE OF PairingProof
```


So long as TS can verify this data then it can check (via whatever targeting is included in the AC) that IS hasn't stolen the AC.

<<note: a scheme which didn't require the client to sign would be better (assuming all servers have signature keys is reasonable). We would still need to provide a trace of delegation starting with the client somehow.>>

<<additional checks to be defined, e.g. initiator from current pair must be target from next pair. times shouldn't be out of whack by too much, etc.>>

4. AC Acquisition

4.1 Client Acquisition

This exchange occurs when a client requires a new AC from

an issuer. The AC issuer MUST ensure that the client is authenticated, most simply via TLS with client authentication. This exchange may occur at network login time or may happen automatically during (or before) the client handshake with a TLS server (e.g. upon receipt of the ACRequest message from the server - see section 8 below).

<<a full handshake between client and issuer may to too much overhead (e.g. if 10,000 clients arrive at 9am). A simple CMS over UDP wrapping of the acinfo would be more efficient and may be added later>>

The client sends the following message to the server (the syntax used is described in [TLS]):

```
struct {
     opaque acinfo<1..2^24-1>
} ClientACRequest
```

acinfo field contents are specified below.

Field	Description
type	clientAcquireResp
peer	server name or missing
template	attributes which the client wishes to be present in the AC
ac	missing
pps	missing
certs	missing

The server responds with:

```
struct {
     enum {
          success(0),
          success_with_changes(1),
          failure(2),
          denied(3),
          (255)
     } acstatus;
     opaque acinfo<1..2^24-1>
} ClientACResponse
```

acinfo field contents are specified below.

Field	Description

Field	Description
type	clientAcquireResp
peer	missing
template	missing
ac	the AC or missing
pps	missing
certs	missing

4.2 Server Acquisition

Server acquisition occurs where a client has established a TLS connection to the server, but hasn't provided (or can't provide) an AC. The case where the client provides a PairingProof is also supported.

Note that the server may keep a TLS session open (or resume a session) with the issuer for multiple exchanges. AC issuers MUST support this feature, servers MAY make use of this feature.

The server sends the following message to the issuer:

```
struct {
     opaque acinfo<1..s^24-1>
} ServerACRequest
```

Field	Description
type	serverAcquire
peer	missing if pps supplied mandatory containing client name if pps missing
template	missing
ac	missing
pps	missing if peer supplied mandatory if peer missing
certs	may be present if pps supplied

The issuer responds with:

```
     struct {
          enum {
               success(0),
               success_with_changes(1),
               failure(2),
               denied(3),
               (255)
          } acstatus;
```

```
            opaque acinfo<1..2^24-1>
        } ServerACResponse
```

The fields here are as described for the ClientACResponse except that the type is serverAcquireResp.

4.3 Client and Server Lookup

ACs are retrieved via LDAP. The LDAP connection MAY be secured with TLS according to local policy.

Given a GeneralName for the peer a client or server will perform the following lookup.

<<derivation of an entry name from the peer GeneralName is tbs - and probably very hard!>>

Once the directory entry has been identified then the value(s) of the attributeCertificateAttribute should be retrieved. This is defined in [X.509] as:

```
attributeCertificateAttribute ATTRIBUTE  ::= {
     WITH SYNTAX               AttributeCertificate
     EQUALITY MATCHING-RULE    certificateExactMatch
     ID                        id-at-attributeCertificate
}

id-at-attributeCertificate ::= OBJECT IDENTIFIER
                                    { 2 5 4 58 }
```

The selection of which of the values of the attribute are to be read is out of scope of this specfication.

5. TLS Protocol Extensions

The basic requirement is to be able to include an acinfo structure into TLS handshakes.

5.1 Session Management

The TLS session state (section 7, p22) now requires a new item:

authorization information
: An acinfo structure containing the authorization information for the session. This element of the state may be null.

5.2 Use of TLS Alerts

No new TLS alerts are required.

5.3 Handshake Protocol Extensions

The basic approach is to handle the acinfo structure in the same way as X.509 public key certificates are handled. This means that we define an ACRequest which the server can send to the client and an ACInfo with which the client can respond. These are analogous to the existing CertificateRequest and CertificateResponse messages. The extension to the interworking diagram from TLS (section 7.3) is shown on the next page.


```
Client                                              Server
ClientHello              ----->
                                                ServerHello
                                               Certificate*
                                         ServerKeyExchange*
                                        CertificateRequest*
                                                 ACRequest*
                         <-----              ServerHelloDone
Certificate*
ACInfo*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                 ----->
                                       [ChangeCipherSpec]
                                                Finished
ApplicationData          <---->             ApplicationData
```

<<note: the new messages could be avoided if the Certificate and CertificateRequest messages were extended, however, the syntax and handling of the ACRequest and ACInfo seem sufficiently different to warrant new messages.>>

It is to be expected that a typical scenario would be where a client establishes a session with a server without authorization support. At some point the client requests access to a resource which can only be granted if a client AC is verified. At this point the server may cause a renegotiation using a HelloRequest message.

This is similar to cases where a client first establishes anonymously and is then forced into mutual

authentication.

The following sections define the new messages in more detail.

5.3.1 ACRequest message

When this message will be sent:
A server can optionally request an AC from the client. This message, if sent, will immediately follow the CertificateRequest (if it is sent).

Structure of this message:

```
struct {
     opaque acinfo<1..2^24-1>
} ACRequest
```


Field	Description
type	acRequest
peer	missing
template	a hint to show the client which attributes are required
ac	missing
pps	missing
certs	missing

The client MAY ignore the template field if present.

5.3.2 ACInfo message

When this message will be sent:
This message must be sent if the client has received an ACRequest message from the server. If sent, it will immediately follow the Certificate message sent by the client (if sent).

Structure of this message:

```
struct {
     opaque acinfo<1..2^24-1>
} ACInfo
```

Field	Description
type	acResponse
peer	missing
template	missing
ac	the AC ; may be missing if a pps is present

pps	optionally present
certs	optionally present

If the ac is missing but the pps is present then the server may attempt to acquire or lookup the AC on the basis of validation of the pps signature. Note however that the pps signature does not authenticate the client to the server as it is not necessarily tied to the TLS session.

Upon receipt of this message the server can now validate the AC. However the server may have to wait until after the Finished message in order to trust the AC (i.e. when the client is authenticated).

6. Conformance

Both clients and servers MUST support the AC "push" and the relevant acquisition exchanges. Client and servers MAY support the lookup exchanges.

A signing algorithm is required for the PairingProof structure, dsaWithSHA1 MUST be supported. Other signature algorithms specified in [CMS] MAY be supported.

7. Security Considerations

A server which accepts an AC from an unauthenticated client may be using a stolen AC. For this reason it is strongly recommended that servers only accept ACs from authenticated clients.

A server which is presented with an AC which is delegatable will be able to masquerade as the AC owner to any of the other servers to which the AC can be delegated. This means that the client is effectively trusting the server to the extent that the AC is delegatable by that server. For this reason it is recommended to limit the delegation scope of ACs to the minimum required.

8. References

[AC509] Farrell, S,"An Internet AttributeCertificate Profile for Authorization",

draft-ietf-tls-ac509prof-00.txt, August 1998.

[CMS] Housley, R., "Cryptographic Message Syntax", draft-ietf-smime-cms-05.txt, May 1998.

[TLS] Dierks, T., Allen C., "The TLS Protocol Version 1.0", draft-ietf-tls-protocol-05.txt, November 1997.

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

Author's Address

Stephen Farrell,
SSE Ltd.
Fitzwilliam Court,
Leeson Close,
Dublin 2,
IRELAND

tel: +353-1-676-9089
email: stephen.farrell@sse.ie