Network Working Group
INTERNET-DRAFT
Expires in six months
Intended Category: Standard

S.E. Kille
MessagingDirect Inc.
February 2000

X.509 Authentication SASL Mechanism
<draft-ietf-ldapext-x509-sasl-03.txt>

1. Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. 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."

> The list of current Internet-Drafts can be accessed at
> http://www.ietf.org/ietf/1id-abstracts.txt
>
> The list of Internet-Draft Shadow Directories can be accessed at
> http://www.ietf.org/shadow.html.

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

2. Abstract

This document defines a SASL [1] authentication mechanism based on X.509 strong authentication [3], providing two way authentication. This mechanism is only for authentication, and has no effect on the protocol encodings and is not designed to provide integrity or confidentiality services.

3. Model

The mechanism provides two way strong authentication as defined in X.509. **The encoding is based on that used by X.500 in the DAP, DSP, and** DISP protocols.

The mechanism is based on use of an asymmetric (public key) signing mechanism. This SASL mechanism contains two authentication mechanisms:

- Client authentication is where the client provides information to the server, so that the server can authenticate the client.

- Server authentication is where the server provides information to

the client, so that the client can authenticate the server.

This mechanism is given three SASL keys for different variants:

- "X509-C-<algorithm>" for client authentication only.

- "X509-S-<algorithm>" for server authentication only.

- "X509-B-<algorithm>" for client and server authentication. In this case client authentication is done prior to server authentication.

Each SASL key may be used with a list of algorithms. A list of supported algorithms is given in Section 7.

For Client Authentication ("X509-C-"):

1. **The client generates the credentials (SASLStrongCredentials) from information on both parties and a random number, and signs the** enclosed token with its own private key.

2. **The client sends credentials to the server.**

3. **The server verifies these credentials using the client's public key, and the authentication is complete.**

For Server Authentication ("X509-S-"):

1. **The server generates the credentials (SASLStrongCredentials) from information on both parties and a random number, and signs the** enclosed token with its own private key.

2. **The server sends credentials to the client.**

3. **The client verifies these credentials using the server's public key, and the authentication is complete.**

For most SASL based protocols, server only authentication will not be useful. However, this is included here, as the definition is required for "client and server", and it may be useful for future protocols.

For Client and Server Authentication ("X509-B-"), the procedure for "X509-C-" is performed and then followed by the procedure for "X509-S-". The Client needs to go first, as for some protocols the server will need to get information about the client from the client authentication in order to be able to perform the server authentication.

3.1. Encoding

The SASLStrongCredentials, which is the definition of the data format exchanged, is encoded using ASN.1 Distinguished Encoding Rules (DER).

4. Why this SASL Mechanism is Needed

This section discusses the requirements for this SASL mechanism.

4.1. Benefits of a Public Key Mechanism

The key benefit of asymmetric (public key) security, is that the secret (private key) only needs to be placed with the entity that is being authenticated. Thus a private key can be issued to a client, which can then be authenticated by ANY server based on a token generated by the client and the generally available public key. Symmetric authentication mechanisms (password mechanisms such as CRAM-MD5) require a shared secret, and the need to maintain it at both endpoints. This means that a secret key for the client needs to be maintained at every server which may need to authenticate the client.

This is particularly an issue for protocols such as LDAP, where a client may connect to and be authenticated by a large number of servers. In this situation, the requirement to maintain secret keys on all possible servers is not practical, which makes authentication mechanisms such as CRAM-MD5 unsuitable for LDAP in many situations.

4.2. Why Authentication Only?

This service provides authentication only. The primary reason for this is that it makes the mechanism very simple. It would be possible to define a more complex mechanism which exchanged session keys and also provided confidentiality and/or integrity.

There are a number of places where an authentication only service is useful:

- Where confidentiality and integrity are provided by lower layers (e.g., TLS or IPSec).

- Where confidentiality or integrity services are provided by the application (e.g., X.500 signed operations).

- Where physical and other security aspects of the environment do no require confidentiality and integrity services.

- For legacy applications where changes to the data exchange would be difficult to integrate.

4.3. Relationship to TLS

The functionality defined here can be provided by TLS, and it is important to consider why it is useful to have it in both places. There are a number of reasons for this:

- SASL. SASL also duplicates TLS functionality, and the rationale for this is clearly given in RFC 2222 [1]. These arguments apply here.

- Simplicity. This mechanism is simpler than TLS. If there is only a requirement for this functionality (as distinct from all of TLS), this simplicity will facilitate deployment.

- Layering. The SASL mechanism to establish authentication works cleanly with most protocols. This mechanism can fit more cleanly than TLS for some protocols.

- Proxy support. Proxys can be cleanly supported with this mechanism. This works because the proxy can authenticate the client, and then simply pass the credentials on the server, using the previous token member.

- This mechanism provides a simpler solution where no Data Confidentiality and integrity required.

5. Token Definition

The SASLStrongCredentials defined here are based on the StrongCredentials defined in X.511, making use of the SIGNED Macro and Certification Path definitions of X.509. Two optional fields have been added, the second of which makes use of GeneralName defined in X.509 [6]. The credentials definition is given here for clarity. The formal definitions of CertificationPath, AlgorithmIdentifier, and DistinguishedName are by reference to X.511. The formal definition of GeneralName is given in X.509.

There are a number of names referenced in these definitions. There are two entities involved in the interaction:

1. **Signer.** This is the entity that is creating the SASLStrongCredentials, and singing it with its private key. The signer is also referred to as the "subject", in line with PKIX terminology, as the signer is essentially proving its own identity.

2. **Target.** This is the entity which is expected to verify the signature. The target's name is included in order to prevent replay attacks, and so it is only the target that can securely verify the

SASLStrongCredentials (other entities can verify the signature, but they would not be able to detect replay attack). This enables the token to be used to build trust chains.

```
SASLStrongCredentials ::= SET {
    certification-path [0] CertificationPath OPTIONAL,
    bind-token         [1] SASLToken,
    name               [2] DistinguishedName OPTIONAL} -- signer

SASLToken ::= SIGNED { SEQUENCE {
    algorithm          [0] AlgorithmIdentifier,
    name               [1] DistinguishedName,  -- target
    time               [2] UTCTime,
    random             [3] BITSTRING,
    response           [4] BITSTRING OPTIONAL,
    target-name        [20] GeneralName OPTIONAL,
    signer-name        [21] GeneralName OPTIONAL,
    previous-token     [22] SASLStrongCredentials OPTIONAL}}

PreviousToken ::= SEQUENCE {
    token       [0] SASLToken,
    protocol           [1] Protocol OPTIONAL }

Protocol ::= INTEGER

GeneralName ::= CHOICE {
    otherName          [0] OtherName,
    rfc822Name         [1] IA5String,
    dNSName            [2] IA5String,
    x400Address        [3] ORAddress,
    directoryName      [4] Name,
    edipartyname       [5] EDIPartyName,
    uniformresouceidentifier [6] IA5String,
    iPAddress          [7] OCTET STRING,
    registeredID       [8] OBJECT IDENTIFIER }
```

The elements of SASLStrongCredentials are as follows:

certification-path:

This provides a mechanism for exchange of certificates, which may help the recipient to verify the credentials. If this is included, it must be consistent with the name in SASLStrongCredentials.name. This information is provided by the entity generating the token to facilitate verification. The entity verifying the token is not required to use this information.

bind-token:

This is the signed token, which is the core of the credentials.

name:This is the distinguished name of the signer of the token. For client authentication, this will need to be included unless the information is carried in another protocol element of the exchange (which will typically not be the case). For server authentication, this will not normally be needed, as the client will have a priori knowledge of the server it is connecting to.

The entity verifying the token shall ensure that this name is consistent with the certificate, as a part of the verification process.

The signed token contains the following elements.

algorithm:
This is the signature algorithm used to sign the token. This is included for compatibility with X.509, and generally implies both an asymmetric algorithm and a hash algorithm. The value SHALL be consistent with the algorithm defined by the SASL mechanism (e.g., DSA-SHA1).

name:This is the distinguished name of the target (which will verify the token). For client authentication, this will be the name of the server.

This element is mandatory for compatibility with X.511. If a name form other than Distinguished Name is used, this element should contain a null distinguished name, and a name included in the signer-name parameter.

time:This is the time that the token expires.

random:
This is a random number, which must be unique for the target over the valid life of the token. This is included to prevent replay attack. It is recommended that this number is at least as long as the block size of the hash algorithm used.

response:
This is used to carry a number derived from random if challenge response of authentication is required. This shall be used in the client phase of X509-B- and in no other circumstances. In this case, the value used in this field in the client authentication shall be used in the SASLToken.random field of the server authentication.

items 5-19:
There is a gap in the sequence numbering. Items 6-9 are used in X.500 DAP, but are not appropriate here.

target-name:
This is a very general definition of a name, taken from X.509(v3). This definition is being used by ongoing work on PKI. This enables authentication identifiers other than distinguished names to be used. This will be important when the target does not have a distinguished name.

This field is needed when access control is to be applied on the basis of a name different to the one doing the signing. The name of the entity doing the signing is in SASLStrongCredentials.name, and this is verified by the signing process. This field, verified by the signature, is an alternate name to be used for access control purposes in the authentication and for ongoing purposes with the association established. In SASL terminology, this is the "authorization entity".

Note that this description is for tutorial purposes only, and the formal definition is taken from X.511.

signer-name
This has the same function as name, but allows for encodings other than Distinguished Name.

previous-token
This gives a mechanism to include a previous token, which includes a SASLToken, and optionally a protocol if this is different from the current protocol. The mechanism might be used in a firewall, which does protocol relay. The initial token is generated by the client, which is then encapsulated in another token generated by the firewall. This enables a signed trust chain to be built. The "change of protocol" enables a server to use a different protocol on behalf of its client (e.g., an ACAP server performing a directory lookup on behalf of the ACAP client).

Protocols are represented as Integers, identified by the TCP Port Number.

6. Distinguished Names

The X.509 strong authentication mechanism makes use of distinguished names to identify the target. For some protocols, such as LDAP [2], this is natural. For protocols which make use of internet domain names to identify objects, the representation defined in RFC 2247 [5] MAY be used as an alternative to subject-name in the token. For an Internet Mailbox the local part must be encoded as a domain component. For example "J.Bloggs@widget.com" is represented as the distinguished name "DC=J.Bloggs,DC=widget,DC=com".

7. Supported Algorithms

The following signature algorithms are recognized for use with this mechanism, and identified by a key. Each key would be combined to make three possible SASL mechanisms. For example for the DSA-SHA1 algorithm would give X509-C-DSA-SHA1, X509-S-DSA-SHA1, and X509-B-DSA-SHA1. All algorithm names are constrained to 13 characters, to keep within the total SASL limit of 20 characters.

The following table gives a list of algorithm keys, noting the object identifier and the body which assigned the identifier.

Key	Object Id	Body
RSA-MD4	1.3.14.2.2.2	OIW
RSA-MD5	1.3.14.2.2.3	OIW
RSA-MD4-ENCR	1.3.14.2.2.4	OIW
DSA-SHA	1.3.14.2.2.13	OIW
DSA-SHA-COMM	1.3.14.2.2.20	OIW
RSA-MD2	1.3.14.7.3.1	OIW
ELGAMAL-MD2	1.3.14.7.3.2	OIW
RSA-MD2-ENCR	1.2.840.113549.1.1.2	RSA
RSA-MD5-ENCR	1.2.840.113549.1.1.4	RSA
RSA-SHA1-ENC	1.2.840.113549.1.1.5	RSA
MSP-SDNS	2.16.840.1.101.2.1.1.1	DMS
MSP-MOSAIC	2.16.840.1.101.2.1.1.2	DMS
DSA-SHA1	1.2.840.10040.4.3	ANSI

Two special algorithm keys are defined:

- IMPLICIT. This is used when the signer has a priori knowledge of the algorithm to use. The algorithms is then identified solely by the AlgrithmIdentifier Object Identifier in the token.

- X-*. Any algorithm starting with "X-" is reserved for private extension.

Support of the DSS-SHA1 algorithm is recommended for use with this mechanism.

8. Example

The following example shows use with IMAP4. The example is designed to illustrate the protocol interaction and does not provide valid encoding examples.

S: * OK IMAP4 server ready
C: AOO1 CAPABILITY

```
S: * CAPABILITY IMAP4 AUTH=CRAM-MD5 AUTH=X509-C-DSA-SHA1 AUTH=X509-C-RSA-MD5
S: A001 OK done
C: AOO2 AUTHENTICATE X509-C-DSS-SHA1
S: +
C: AE31FF05.......==
S: + 13c3FF44.......==
S: AOO3 OK Welcome, authenticated user: CN=Joe Bloggs,O=Widget,C=GB
```

Editor's Note:
The ASN.1 values here are fake. A real example should be used

(perhaps include ASN.1 value notation), when one can be generated from a prototype implementation.

9. Security Considerations

These algorithms are designed to be used for authentication where the underlying transport service cannot guarantee confidentiality. These mechanisms do not prevent an authenticated association from being hijacked.

10. Acknowledgments

Design ideas included in this document are based on those from ITU and ISO, and the IETF ASID Working Groups. Useful ideas were taken from a note "X.500 Strong Authentication Mechanisms for LDAPv3" by Mark Wahl. The contributions of individuals in these working groups, including Harald Alvestrand (Maxware), Alexis Bor (Directory Works), David Boyce (Isode), William Curtin (DISA), Bruce Greenblatt (RSA), Steve Hole (Esys), Tim Howes (Netscape), John Myers (Netscape), Chris Newman (Inno-soft), Frank Siebeblist (DASCOM), Erik Skovgaard (Geotrain), and Sean Turner (IECA) are gratefully acknowledged.

11. Bibliography

[1] J. Meyers, "Simple Authentication and Security Layer", RFC 2222, October 1997.

[2] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2252, December 1997.

[3] ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8:1997, Information Technology - Open Systems Interconnection - The Directory: Authentica-tion Framework.

[4] ITU-T Recommendation X.511 (1997) | ISO/IEC 9594-8:1997, Information Technology - Open Systems Interconnection - The Directory: Abstract Service Definition.

[5] S. Kille, M.Wahl, A. Grimstad, R. Huber, S. Sataluri, "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, January 1998.

12. Author's Address

Steve Kille
MessagingDirect Inc
The Dome, The Square
Richmond, Surrey,
TW9 1DT, UK

Phone: +44-20-8332-9091
Email: Steve.Kille@messagingdirect.com