Network Working Group
Request for Comments: 2829
Category: Standards Track

M. Wahl
Sun Microsystems, Inc.
H. Alvestrand
EDB Maxware
J. Hodges
Oblix, Inc.
R. Morgan
University of Washington

May 2000

Authentication Methods for LDAP

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This document specifies particular combinations of security mechanisms which are required and recommended in LDAP [1] implementations.

1. Introduction

LDAP version 3 is a powerful access protocol for directories.

It offers means of searching, fetching and manipulating directory content, and ways to access a rich set of security functions.

In order to function for the best of the Internet, it is vital that these security functions be interoperable; therefore there has to be a minimum subset of security functions that is common to all implementations that claim LDAPv3 conformance.

Basic threats to an LDAP directory service include:

(1) Unauthorized access to data via data-fetching operations,

(2) Unauthorized access to reusable client authentication information by monitoring others' access,

(3) Unauthorized access to data by monitoring others' access,

(4) Unauthorized modification of data,

(5) Unauthorized modification of configuration,

(6) Unauthorized or excessive use of resources (denial of service), and

(7) Spoofing of directory: Tricking a client into believing that information came from the directory when in fact it did not, either by modifying data in transit or misdirecting the client's connection.

Threats (1), (4), (5) and (6) are due to hostile clients. Threats (2), (3) and (7) are due to hostile agents on the path between client and server, or posing as a server.

The LDAP protocol suite can be protected with the following security mechanisms:

(1) Client authentication by means of the SASL [2] mechanism set, possibly backed by the TLS credentials exchange mechanism,

(2) Client authorization by means of access control based on the requestor's authenticated identity,

(3) Data integrity protection by means of the TLS protocol or data-integrity SASL mechanisms,

(4) Protection against snooping by means of the TLS protocol or data-encrypting SASL mechanisms,

(5) Resource limitation by means of administrative limits on service controls, and

(6) Server authentication by means of the TLS protocol or SASL mechanism.

At the moment, imposition of access controls is done by means outside the scope of the LDAP protocol.

In this document, the term "user" represents any application which is an LDAP client using the directory to retrieve or store information.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3].

2. Example deployment scenarios

The following scenarios are typical for LDAP directories on the Internet, and have different security requirements. (In the following, "sensitive" means data that will cause real damage to the owner if revealed; there may be data that is protected but not sensitive). This is not intended to be a comprehensive list, other scenarios are possible, especially on physically protected networks.

(1) A read-only directory, containing no sensitive data, accessible to "anyone", and TCP connection hijacking or IP spoofing is not a problem. This directory requires no security functions except administrative service limits.

(2) A read-only directory containing no sensitive data; read access is granted based on identity. TCP connection hijacking is not currently a problem. This scenario requires a secure authentication function.

(3) A read-only directory containing no sensitive data; and the client needs to ensure that the directory data is authenticated by the server and not modified while being returned from the server.

(4) A read-write directory, containing no sensitive data; read access is available to "anyone", update access to properly authorized persons. TCP connection hijacking is not currently a problem. This scenario requires a secure authentication function.

(5) A directory containing sensitive data. This scenario requires session confidentiality protection AND secure authentication.

3. Authentication and Authorization: Definitions and Concepts

This section defines basic terms, concepts, and interrelationships regarding authentication, authorization, credentials, and identity. These concepts are used in describing how various security approaches are utilized in client authentication and authorization.

3.1. Access Control Policy

An access control policy is a set of rules defining the protection of resources, generally in terms of the capabilities of persons or other entities accessing those resources. A common expression of an access control policy is an access control list. Security objects and mechanisms, such as those described here, enable the expression of access control policies and their enforcement. Access control policies are typically expressed in terms of access control attributes as described below.

3.2. Access Control Factors

A request, when it is being processed by a server, may be associated with a wide variety of security-related factors (section 4.2 of [1]). The server uses these factors to determine whether and how to process the request. These are called access control factors (ACFs). They might include source IP address, encryption strength, the type of operation being requested, time of day, etc. Some factors may be specific to the request itself, others may be associated with the connection via which the request is transmitted, others (e.g. time of day) may be "environmental".

Access control policies are expressed in terms of access control factors. E.g., a request having ACFs i,j,k can perform operation Y on resource Z. The set of ACFs that a server makes available for such expressions is implementation-specific.

3.3. Authentication, Credentials, Identity

Authentication credentials are the evidence supplied by one party to another, asserting the identity of the supplying party (e.g. a user) who is attempting to establish an association with the other party (typically a server). Authentication is the process of generating, transmitting, and verifying these credentials and thus the identity they assert. An authentication identity is the name presented in a credential.

There are many forms of authentication credentials -- the form used depends upon the particular authentication mechanism negotiated by the parties. For example: X.509 certificates, Kerberos tickets, simple identity and password pairs. Note that an authentication mechanism may constrain the form of authentication identities used with it.

3.4. Authorization Identity

An authorization identity is one kind of access control factor. It is the name of the user or other entity that requests that operations be performed. Access control policies are often expressed in terms of authorization identities; e.g., entity X can perform operation Y on resource Z.

The authorization identity bound to an association is often exactly the same as the authentication identity presented by the client, but it may be different. SASL allows clients to specify an authorization identity distinct from the authentication identity asserted by the client's credentials. This permits agents such as proxy servers to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying [2]. Also, the form of authentication identity supplied by a service like TLS may not correspond to the authorization identities used to express a server's access control policy, requiring a server-specific mapping to be done. The method by which a server composes and validates an authorization identity from the authentication credentials supplied by a client is implementation-specific.

4. Required security mechanisms

It is clear that allowing any implementation, faced with the above requirements, to pick and choose among the possible alternatives is not a strategy that is likely to lead to interoperability. In the absence of mandates, clients will be written that do not support any security function supported by the server, or worse, support only mechanisms like cleartext passwords that provide clearly inadequate security.

Active intermediary attacks are the most difficult for an attacker to perform, and for an implementation to protect against. Methods that protect only against hostile client and passive eavesdropping attacks are useful in situations where the cost of protection against active intermediary attacks is not justified based on the perceived risk of active intermediary attacks.

Given the presence of the Directory, there is a strong desire to see mechanisms where identities take the form of a Distinguished Name and authentication data can be stored in the directory; this means that either this data is useless for faking authentication (like the Unix "/etc/passwd" file format used to be), or its content is never passed across the wire unprotected - that is, it's either updated outside the protocol or it is only updated in sessions well protected against snooping. It is also desirable to allow authentication methods to

carry authorization identities based on existing forms of user identities for backwards compatibility with non-LDAP-based authentication services.

Therefore, the following implementation conformance requirements are in place:

(1) For a read-only, public directory, anonymous authentication, described in section 5, can be used.

(2) Implementations providing password-based authenticated access MUST support authentication using the DIGEST-MD5 SASL mechanism [4], as described in section 6.1. This provides client authentication with protection against passive eavesdropping attacks, but does not provide protection against active intermediary attacks.

(3) For a directory needing session protection and authentication, the Start TLS extended operation [5], and either the simple authentication choice or the SASL EXTERNAL mechanism, are to be used together. Implementations SHOULD support authentication with a password as described in section 6.2, and SHOULD support authentication with a certificate as described in section 7.1. Together, these can provide integrity and disclosure protection of transmitted data, and authentication of client and server, including protection against active intermediary attacks.

If TLS is negotiated, the client MUST discard all information about the server fetched prior to the TLS negotiation. In particular, the value of supportedSASLMechanisms MAY be different after TLS has been negotiated (specifically, the EXTERNAL mechanism or the proposed PLAIN mechanism are likely to only be listed after a TLS negotiation has been performed).

If a SASL security layer is negotiated, the client MUST discard all information about the server fetched prior to SASL. In particular, if the client is configured to support multiple SASL mechanisms, it SHOULD fetch supportedSASLMechanisms both before and after the SASL security layer is negotiated and verify that the value has not changed after the SASL security layer was negotiated. This detects active attacks which remove supported SASL mechanisms from the supportedSASLMechanisms list, and allows the client to ensure that it is using the best mechanism supported by both client and server (additionally, this is a SHOULD to allow for environments where the supported SASL mechanisms list is provided to the client through a different trusted source, e.g. as part of a digitally signed object).

5. Anonymous authentication

Directory operations which modify entries or access protected attributes or entries generally require client authentication. Clients which do not intend to perform any of these operations typically use anonymous authentication.

LDAP implementations MUST support anonymous authentication, as defined in section 5.1.

LDAP implementations MAY support anonymous authentication with TLS, as defined in section 5.2.

While there MAY be access control restrictions to prevent access to directory entries, an LDAP server SHOULD allow an anonymously-bound client to retrieve the supportedSASLMechanisms attribute of the root DSE.

An LDAP server MAY use other information about the client provided by the lower layers or external means to grant or deny access even to anonymously authenticated clients.

5.1. Anonymous authentication procedure

An LDAP client which has not successfully completed a bind operation on a connection is anonymously authenticated.

An LDAP client MAY also specify anonymous authentication in a bind request by using a zero-length OCTET STRING with the simple authentication choice.

5.2. Anonymous authentication and TLS

An LDAP client MAY use the Start TLS operation [5] to negotiate the use of TLS security [6]. If the client has not bound beforehand, then until the client uses the EXTERNAL SASL mechanism to negotiate the recognition of the client's certificate, the client is anonymously authenticated.

Recommendations on TLS ciphersuites are given in section 10.

An LDAP server which requests that clients provide their certificate during TLS negotiation MAY use a local security policy to determine whether to successfully complete TLS negotiation if the client did not present a certificate which could be validated.

6. Password-based authentication

LDAP implementations MUST support authentication with a password using the DIGEST-MD5 SASL mechanism for password protection, as defined in section 6.1.

LDAP implementations SHOULD support authentication with the "simple" password choice when the connection is protected against eavesdropping using TLS, as defined in section 6.2.

6.1. Digest authentication

An LDAP client MAY determine whether the server supports this mechanism by performing a search request on the root DSE, requesting the supportedSASLMechanisms attribute, and checking whether the string "DIGEST-MD5" is present as a value of this attribute.

In the first stage of authentication, when the client is performing an "initial authentication" as defined in section 2.1 of [4], the client sends a bind request in which the version number is 3, the authentication choice is sasl, the sasl mechanism name is "DIGEST-MD5", and the credentials are absent. The client then waits for a response from the server to this request.

The server will respond with a bind response in which the resultCode is saslBindInProgress, and the serverSaslCreds field is present. The contents of this field is a string defined by "digest-challenge" in section 2.1.1 of [4]. The server SHOULD include a realm indication and MUST indicate support for UTF-8.

The client will send a bind request with a distinct message id, in which the version number is 3, the authentication choice is sasl, the sasl mechanism name is "DIGEST-MD5", and the credentials contain the string defined by "digest-response" in section 2.1.2 of [4]. The serv-type is "ldap".

The server will respond with a bind response in which the resultCode is either success, or an error indication. If the authentication is successful and the server does not support subsequent authentication, then the credentials field is absent. If the authentication is successful and the server supports subsequent authentication, then the credentials field contains the string defined by "response-auth" in section 2.1.3 of [4]. Support for subsequent authentication is OPTIONAL in clients and servers.

6.2. "simple" authentication choice under TLS encryption

A user who has a directory entry containing a userPassword attribute MAY authenticate to the directory by performing a simple password bind sequence following the negotiation of a TLS ciphersuite providing connection confidentiality [6].

The client will use the Start TLS operation [5] to negotiate the use of TLS security [6] on the connection to the LDAP server. The client need not have bound to the directory beforehand.

For this authentication procedure to be successful, the client and server MUST negotiate a ciphersuite which contains a bulk encryption algorithm of appropriate strength. Recommendations on cipher suites are given in section 10.

Following the successful completion of TLS negotiation, the client MUST send an LDAP bind request with the version number of 3, the name field containing the name of the user's entry, and the "simple" authentication choice, containing a password.

The server will, for each value of the userPassword attribute in the named user's entry, compare these for case-sensitive equality with the client's presented password. If there is a match, then the server will respond with resultCode success, otherwise the server will respond with resultCode invalidCredentials.

6.3. Other authentication choices with TLS

It is also possible, following the negotiation of TLS, to perform a SASL authentication which does not involve the exchange of plaintext reusable passwords. In this case the client and server need not negotiate a ciphersuite which provides confidentiality if the only service required is data integrity.

7. Certificate-based authentication

LDAP implementations SHOULD support authentication via a client certificate in TLS, as defined in section 7.1.

7.1. Certificate-based authentication with TLS

A user who has a public/private key pair in which the public key has been signed by a Certification Authority may use this key pair to authenticate to the directory server if the user's certificate is requested by the server. The user's certificate subject field SHOULD be the name of the user's directory entry, and the Certification Authority must be sufficiently trusted by the directory server to

have issued the certificate in order that the server can process the certificate. The means by which servers validate certificate paths is outside the scope of this document.

A server MAY support mappings for certificates in which the subject field name is different from the name of the user's directory entry. A server which supports mappings of names MUST be capable of being configured to support certificates for which no mapping is required.

The client will use the Start TLS operation [5] to negotiate the use of TLS security [6] on the connection to the LDAP server. The client need not have bound to the directory beforehand.

In the TLS negotiation, the server MUST request a certificate. The client will provide its certificate to the server, and MUST perform a private key-based encryption, proving it has the private key associated with the certificate.

As deployments will require protection of sensitive data in transit, the client and server MUST negotiate a ciphersuite which contains a bulk encryption algorithm of appropriate strength. Recommendations of cipher suites are given in section 10.

The server MUST verify that the client's certificate is valid. The server will normally check that the certificate is issued by a known CA, and that none of the certificates on the client's certificate chain are invalid or revoked. There are several procedures by which the server can perform these checks.

Following the successful completion of TLS negotiation, the client will send an LDAP bind request with the SASL "EXTERNAL" mechanism.

8. Other mechanisms

The LDAP "simple" authentication choice is not suitable for authentication on the Internet where there is no network or transport layer confidentiality.

As LDAP includes native anonymous and plaintext authentication methods, the "ANONYMOUS" and "PLAIN" SASL mechanisms are not used with LDAP. If an authorization identity of a form different from a DN is requested by the client, a mechanism that protects the password in transit SHOULD be used.

The following SASL-based mechanisms are not considered in this document: KERBEROS_V4, GSSAPI and SKEY.

The "EXTERNAL" SASL mechanism can be used to request the LDAP server make use of security credentials exchanged by a lower layer. If a TLS session has not been established between the client and server prior to making the SASL EXTERNAL Bind request and there is no other external source of authentication credentials (e.g. IP-level security [8]), or if, during the process of establishing the TLS session, the server did not request the client's authentication credentials, the SASL EXTERNAL bind MUST fail with a result code of inappropriateAuthentication. Any client authentication and authorization state of the LDAP association is lost, so the LDAP association is in an anonymous state after the failure.

9. Authorization Identity

The authorization identity is carried as part of the SASL credentials field in the LDAP Bind request and response.

When the "EXTERNAL" mechanism is being negotiated, if the credentials field is present, it contains an authorization identity of the authzId form described below.

Other mechanisms define the location of the authorization identity in the credentials field.

The authorization identity is a string in the UTF-8 character set, corresponding to the following ABNF [7]:

```
; Specific predefined authorization (authz) id schemes are
; defined below -- new schemes may be defined in the future.

authzId     = dnAuthzId / uAuthzId

; distinguished-name-based authz id.
dnAuthzId  = "dn:" dn
dn         = utf8string    ; with syntax defined in RFC 2253

; unspecified userid, UTF-8 encoded.
uAuthzId   = "u:" userid
userid     = utf8string    ; syntax unspecified
```

A utf8string is defined to be the UTF-8 encoding of one or more ISO 10646 characters.

All servers which support the storage of authentication credentials, such as passwords or certificates, in the directory MUST support the dnAuthzId choice.

The uAuthzId choice allows for compatibility with client applications which wish to authenticate to a local directory but do not know their own Distinguished Name or have a directory entry. The format of the string is defined as only a sequence of UTF-8 encoded ISO 10646 characters, and further interpretation is subject to prior agreement between the client and server.

For example, the userid could identify a user of a specific directory service, or be a login name or the local-part of an RFC 822 email address. In general a uAuthzId MUST NOT be assumed to be globally unique.

Additional authorization identity schemes MAY be defined in future versions of this document.

10. TLS Ciphersuites

The following ciphersuites defined in [6] MUST NOT be used for confidentiality protection of passwords or data:

```
TLS_NULL_WITH_NULL_NULL
TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA
```

The following ciphersuites defined in [6] can be cracked easily (less than a week of CPU time on a standard CPU in 1997). The client and server SHOULD carefully consider the value of the password or data being protected before using these ciphersuites:

```
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
```

The following ciphersuites are vulnerable to man-in-the-middle attacks, and SHOULD NOT be used to protect passwords or sensitive data, unless the network configuration is such that the danger of a man-in-the-middle attack is tolerable:

```
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
TLS_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
TLS_DH_anon_WITH_DES_CBC_SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA
```

A client or server that supports TLS MUST support at least TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA.

11. SASL service name for LDAP

For use with SASL [2], a protocol must specify a service name to be used with various SASL mechanisms, such as GSSAPI. For LDAP, the service name is "ldap", which has been registered with the IANA as a GSSAPI service name.

12. Security Considerations

Security issues are discussed throughout this memo; the (unsurprising) conclusion is that mandatory security is important, and that session encryption is required when snooping is a problem.

Servers are encouraged to prevent modifications by anonymous users. Servers may also wish to minimize denial of service attacks by timing out idle connections, and returning the unwillingToPerform result code rather than performing computationally expensive operations requested by unauthorized clients.

A connection on which the client has not performed the Start TLS operation or negotiated a suitable SASL mechanism for connection integrity and encryption services is subject to man-in-the-middle attacks to view and modify information in transit.

Additional security considerations relating to the EXTERNAL mechanism to negotiate TLS can be found in [2], [5] and [6].

13. Acknowledgements

This document is a product of the LDAPEXT Working Group of the IETF. The contributions of its members is greatly appreciated.

14. Bibliography

[1] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[2] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

[5] Hodges, J., Morgan, R. and M. Wahl, "Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security", RFC 2830, May 2000.

[6] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[7] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[8] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

15. Authors' Addresses

Mark Wahl
Sun Microsystems, Inc.
8911 Capital of Texas Hwy #4140
Austin TX 78759
USA

EMail: M.Wahl@innosoft.com

Harald Tveit Alvestrand
EDB Maxware
Pirsenteret
N-7462 Trondheim, Norway

Phone: +47 73 54 57 97
EMail: Harald@Alvestrand.no

Jeff Hodges
Oblix, Inc.
18922 Forge Drive
Cupertino, CA 95014
USA

Phone: +1-408-861-6656
EMail: JHodges@oblix.com

RL "Bob" Morgan
Computing and Communications
University of Washington
Seattle, WA 98105
USA

Phone: +1-206-221-3307
EMail: rlmorgan@washington.edu

16. Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.