Individual Submission R. Harrison, Editor
Internet Draft Novell, Inc.
Document: draft-ietf-ldapbis-authmeth-02.txt November 14, 2001
Intended Category: Draft Standard
Obsoletes: RFC 2829, RFC 2830

Authentication Methods and Connection Level Security Mechanisms for LDAPv3

Status of this Memo

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

This document is intended to be, after appropriate review and revision, submitted to the RFC Editor as a Standard Track document. Distribution of this memo is unlimited. Technical discussion of this document will take place on the IETF LDAP Extension Working Group mailing list <ietf-ldapbis@OpenLDAP.org>. Please send editorial comments directly to the author <roger_harrison@novell.com>.

Abstract

This document describes LDAPv3 authentication methods and connection level security mechanisms that are required of all conforming LDAPv3 server implementations and makes recommendations for combinations of these mechanisms to be used in various deployment circumstances.

Among the mechanisms described are

- the LDAPv3 Bind operation used for authenticating LDAP clients to LDAP servers.

- the Start TLS operation used to initiate Transport Layer Security on an established connection between an LDAP client and server.

- various forms of authentication including anonymous authentication, password-based authentication, and certificate based authentication.

1. Conventions Used in this Document

1.1. Glossary of Terms

The following terms are used in this document. To aid the reader, these terms are defined here.

- "user" represents any application which is an LDAP client using the directory to retrieve or store information.

- "LDAP association" is used to distinguish the LDAP-level connection from any underlying TLS-level connection that may or may not exist.

1.2. Security Terms and Concepts

In general, security terms in this document are used consistently with the definitions provided in [RFC2828]. In addition, several terms and concepts relating to security, authentication, and authorization are presented in Appendix B of this document. While the formal definition of these terms and concepts is outside the scope of this document, an understanding of them is prerequisite to understanding much of the material in this document. Readers who are unfamiliar with security-related concepts are encouraged to review Appendix B before reading the remainder of this document.

1.3. Keywords

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 [ReqsKeywords].

2. Introduction

LDAPv3 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.

It is vital that these security functions be interoperable among all LDAP clients and servers on the Internet; 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 directory data via data-fetching operations,

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

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

(4) Unauthorized modification of directory data,

(5) Unauthorized modification of configuration information,

(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 hostile agents posing as a server.

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

(1) Client authentication by means of the SASL [SASL] mechanism set, possibly backed by the TLS [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 SASL mechanisms that provide data integrity services,

(4) Data confidentiality protection against snooping by means of the TLS protocol or SASL mechanisms that provide data confidentiality services,

(5) Server resource usage limitation by means of administrative service limits configured on the server, 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.

3. 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 an LDAP distinguished name [LDAPDN] 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 7, can be used.

(2) Implementations providing password-based authenticated access MUST support authentication using the DIGEST-MD5 SASL mechanism [4], as described in section 8.2. This provides client authentication with protection against passive eavesdropping attacks, but does not provide protection against active intermediary attacks.

(3) For a directory needing data security (both data integrity and data confidentiality) and authentication, the Start TLS operation described in section 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 8.3, and SHOULD support authentication with a certificate as described in section 9.1. 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 initiation of 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 the initiation of the SASL negotiation. If the client is configured to support multiple SASL mechanisms, it SHOULD fetch the supportedSASLmechanisms list both before and after the SASL security layer is negotiated. This allows the client to detect active attacks that 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. (This requirement is a SHOULD to allow for environments where the supportedSASLMechanisms list is provided to the client through a different trusted source, e.g. as part of a digitally signed object.)

Appendix A contains example deployment scenarios that list the mechanisms that might be used to achieve a reasonable level of security in various circumstances.

4. Bind Operation

The Bind operation allows authentication information to be exchanged between the client and server.

4.1. Bind Request

The Bind Request is defined in section 4.2 of [LDAPv3] as follows:

```
     BindRequest ::= [APPLICATION 0] SEQUENCE {
             version                 INTEGER (1 .. 127),
             name                    LDAPDN,
             authentication          AuthenticationChoice }

     AuthenticationChoice ::= CHOICE {
             simple                  [0] OCTET STRING,
                                     -- 1 and 2 reserved
             sasl                    SaslCredentials }

     SaslCredentials ::= SEQUENCE {
             mechanism               LDAPString,
             credentials             OCTET STRING OPTIONAL }
```

Parameters of the Bind Request are:

- version: A version number indicating the version of the protocol to be used in this protocol session. This document describes version 3 of the LDAP protocol. Note that there is no version negotiation, and the client just sets this parameter to the version it desires. If the client requests protocol version 2, a server that supports the version 2 protocol as described in [RFC1777] will not return any v3-specific protocol fields. (Note that not all LDAP servers will support protocol version 2, since they may be unable to generate the attribute syntaxes associated with version 2.)

- name: The name of the directory object that the client wishes to bind as. This field may take on a null value (a zero length string) for the purposes of anonymous binds, when authentication has been performed at a lower layer, or when using SASL credentials with a mechanism that includes the name in the credentials. Server behavior is undefined when the name is a null value, simple authentication is used, and a password is specified. Note that the server SHOULD NOT perform any alias dereferencing in determining the object to bind as.

- authentication: information used to authenticate the name, if any, provided in the Bind Request.

Upon receipt of a Bind Request, a protocol server will authenticate the requesting client, if necessary. The server will then return a Bind Response to the client indicating the status of the authentication.

Authorization is the use of this authentication information when performing operations. Authorization MAY be affected by factors outside of the LDAP Bind request, such as lower layer security services.

4.2. Bind Response

The Bind Response is defined in section 4.2 of [LDAPv3] as follows.

```
BindResponse ::= [APPLICATION 1] SEQUENCE {
     COMPONENTS OF LDAPResult,
     serverSaslCreds    [ABNF] OCTET STRING OPTIONAL }
```

BindResponse consists simply of an indication from the server of the status of the client's request for authentication.

If the bind was successful, the resultCode will be success. Otherwise it will be one of:

- operationsError: server encountered an internal error.

- protocolError: unrecognized version number or incorrect PDU structure.

- authMethodNotSupported: unrecognized SASL mechanism name.

- strongAuthRequired: the server requires authentication be performed with a SASL mechanism.

- referral: this server cannot accept this bind and the client should try another.

- saslBindInProgress: the server requires the client to send a new bind request, with the same sasl mechanism, to continue the authentication process.

- inappropriateAuthentication: the server requires the client which had attempted to bind anonymously or without supplying credentials to provide some form of credentials.

- invalidCredentials: the wrong password was supplied or the SASL credentials could not be processed.

- unavailable: the server is shutting down.

If the server does not support the client's requested protocol version it MUST set the resultCode to protocolError.

If the client receives a BindResponse response where the resultCode was protocolError it MUST close the connection as the server will be unwilling to accept further operations. (This is for compatibility with earlier versions of LDAP, in which the bind was always the first operation and there was no negotiation.)

The serverSaslCreds are used as part of a SASL-defined bind mechanism to allow the client to authenticate the server to which it is communicating, or to perform "challenge-response" authentication. If the client bound with the password choice, or the SASL mechanism does not require the server to return information to the client, then this field is not to be included in the result.

4.3. Sequencing of the Bind Operation

4.3.1. Effect of Multiple Bind Requests

Subsequent to sending a bind request, A client MAY send a bind request to change its identity. Such a bind request has the effect of abandoning all operations outstanding on the connection. (This simplifies server implementation.) Authentication from earlier binds are subsequently ignored, and so if the bind fails, the connection will be treated as anonymous (see section 4.3.3). If a SASL transfer encryption or integrity mechanism has been negotiated, and that mechanism does not support the changing of credentials from one identity to another, then the client MUST instead establish a new connection.

For some SASL authentication mechanisms, it may be necessary for the client to invoke the BindRequest multiple times. If at any stage the client wishes to abort the bind process it MAY unbind and then drop the underlying connection. Clients MUST NOT invoke operations between two Bind requests made as part of a multi-stage bind.

4.3.2. Aborting SASL Bind Negotiation

A client may abort a SASL bind negotiation by sending a BindRequest with a different value in the mechanism field of SaslCredentials, or an AuthenticationChoice other than sasl.

If the client sends a BindRequest with the sasl mechanism field as an empty string, the server MUST return a BindResponse with authMethodNotSupported as the resultCode. This will allow clients to abort a negotiation if it wishes to try again with the same SASL mechanism.

4.3.3. Unbound Connection Treated as Anonymous

Unlike LDAP v2, the client need not send a Bind Request in the first PDU of the connection. The client may request any operations and the server MUST treat these as anonymous. If the server requires that the client bind before browsing or modifying the directory, the server MAY reject a request other than binding, unbinding or an extended request with the "operationsError" result.

If the client did not bind before sending a request and receives an operationsError, it may then send a Bind Request. If this also fails or the client chooses not to bind on the existing connection, it will close the connection, reopen it and begin again by first sending a PDU with a Bind Request. This will aid in interoperating with servers implementing other versions of LDAP.

4.4. Using SASL for Other Security Services

The simple authentication option provides minimal authentication facilities, with the contents of the authentication field consisting only of a cleartext password. Note that the use of cleartext passwords is not recommended over open networks when the underlying transport service cannot guarantee confidentiality; see the "Security Considerations" section.

The sasl choice allows for any mechanism defined for use with SASL [RFC2222] not specifically prohibited by this document (see section 4.4.1). The mechanism field contains the name of the mechanism. The credentials field contains the arbitrary data used for authentication, inside an OCTET STRING wrapper. Note that unlike some Internet application protocols where SASL is used, LDAP is not text-based, thus no Base64 transformations are performed on the credentials.

If any SASL-based integrity or confidentiality services are enabled, they take effect following the transmission by the server and reception by the client of the final BindResponse with resultCode success.

The client can request that the server use authentication information from a lower layer protocol by using the SASL EXTERNAL mechanism (see section 5.5.1.2).

4.4.1. Use of ANONYMOUS and PLAIN SASL Mechanisms

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.

4.4.2. Use of EXTERNAL SASL Mechanism

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 [RFC2401]), 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.

4.4.3. SASL Mechanisms not Considered in this Document

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

4.5. SASL Authorization Identity

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

When the "EXTERNAL" SASL 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.

4.5.1. Authorization Identity Syntax

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

```
; 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 [LDAPDN] section 3.


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

4.5.1.1. DN-based Authorization Identity

All servers that support the storage of authentication credentials, such as passwords or certificates, in the directory MUST support the dnAuthzId choice. The format for distinguishedName is defined in section 3 of [LDAPDN].

4.5.1.2. Unspecified Authorization Identity

The uAuthzId choice allows for compatibility with client applications that wish to authenticate to a local directory but do not know their own distinguished name or that do not have a directory entry. The format of utf8string 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.

4.6. SASL Service Name for LDAP

For use with SASL [SASL], 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.

4.7. SASL Integrity and Privacy Protections

Any negotiated SASL integrity and privacy protections SHALL start on the first octet of the first LDAP PDU following successful completion of the SASL bind operation. If lower level security layer is negotiated, such as TLS, any SASL security services SHALL be layered on top of such security layers regardless of the order of their negotiation.

5. Start TLS Operation

The Start Transport Layer Security (StartTLS) operation provides the ability to establish Transport Layer Security [TLS] on an LDAP association.

5.1. Start TLS Request

A client requests TLS establishment by transmitting a Start TLS request PDU to the server. The Start TLS request is defined in terms of the [LDAPv3] ExtendedRequest as follows:

```
    ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName             [0] LDAPOID,
        requestValue            [LDAPv3] OCTET STRING OPTIONAL }
```

The requestName portion of the Start TLS request MUST be the OID "1.3.6.1.4.1.1466.20037".

The requestValue field is absent.

The client MUST NOT send any PDUs on this connection following this request until it receives a Start TLS extended response.

5.2. Start TLS Response

When a Start TLS request is made, the server MUST return a Start TLS response PDU to the requestor. The Start TLS response id defined in terms of the [LDAPv3] ExtendedResponse as follows:

```
     ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
             COMPONENTS OF LDAPResult,
             responseName     [10] LDAPOID OPTIONAL,
             response         [11] OCTET STRING OPTIONAL }
```

The responseName portion of the Start TLS response MUST be the OID "1.3.6.1.4.1.1466.20037". (Note that this OID is the same OID value used in the requestName of the Start TLS request.)

The response field is absent.

The server MUST set the resultCode field to either success or one of the other values outlined in section 5.2.2.

5.2.1. "Success" Response

If the ExtendedResponse contains a resultCode of success, this indicates that the server is willing and able to negotiate TLS. Refer to section 5.3, below, for details.

5.2.2. Response other than "success"

If the ExtendedResponse contains a resultCode other than success, this indicates that the server is unwilling or unable to negotiate TLS.

If the Start TLS extended request was not successful, the resultCode will be one of:

operationsError	(operations sequencing incorrect; e.g. TLS already established)
protocolError	(TLS not supported or incorrect PDU structure)
referral	(this server doesn't do TLS, try this one)
unavailable	(e.g. some major problem with TLS, or server is shutting down)

The server MUST return operationsError if the client violates any of the Start TLS extended operation sequencing requirements described in section 5.3, below.

If the server does not support TLS (whether by design or by current configuration), it MUST set the resultCode to protocolError (see section 4.1.1 of [LDAPv3]), or to referral. The server MUST include an actual referral value in the LDAP Result if it returns a resultCode of referral. The client's current session is unaffected if the server does not support TLS. The client MAY proceed with any LDAP operation, or it MAY close the connection.

The server MUST return unavailable if it supports TLS but cannot establish a TLS connection for some reason, e.g. the certificate server not responding, it cannot contact its TLS implementation, or if the server is in process of shutting down. The client MAY retry the StartTLS operation, or it MAY proceed with any other LDAP operation, or it MAY close the connection.

5.3. Sequencing of the Start TLS Operation

This section describes the overall procedures clients and servers MUST follow for TLS establishment. These procedures take into consideration various aspects of the overall security of the LDAP association including discovery of resultant security level and assertion of the client's authorization identity.

Note that the precise effects, on a client's authorization identity, of establishing TLS on an LDAP association are described in detail in section 5.5.

5.3.1. Requesting to Start TLS on an LDAP Association

The client MAY send the Start TLS extended request at any time after establishing an LDAP association, except that in the following cases the client MUST NOT send a Start TLS extended request:

- if TLS is currently established on the connection, or
- during a multi-stage SASL negotiation, or

- if there are any LDAP operations outstanding on the connection.

The result of violating any of these requirements is a resultCode of operationsError, as described above in section 5.2.2.

The client MAY have already performed a Bind operation when it sends a Start TLS request, or the client might have not yet bound.

If the client did not establish a TLS connection before sending any other requests, and the server requires the client to establish a TLS connection before performing a particular request, the server MUST reject that request with a confidentialityRequired or strongAuthRequired result. The client MAY send a Start TLS extended request, or it MAY choose to close the connection.

5.3.2. Starting TLS

The server will return an extended response with the resultCode of success if it is willing and able to negotiate TLS. It will return other resultCodes, documented above, if it is unable.

In the successful case, the client, which has ceased to transfer LDAP requests on the connection, MUST either begin a TLS negotiation or close the connection. The client will send PDUs in the TLS Record Protocol directly over the underlying transport connection to the server to initiate TLS negotiation [TLS].

5.3.3. TLS Version Negotiation

Negotiating the version of TLS or SSL to be used is a part of the TLS Handshake Protocol, as documented in [TLS]. Please refer to that document for details.

5.3.4. Discovery of Resultant Security Level

After a TLS connection is established on an LDAP association, both parties MUST individually decide whether or not to continue based on the privacy level achieved. Ascertaining the TLS connection's privacy level is implementation dependent, and accomplished by communicating with one's respective local TLS implementation.

If the client or server decides that the level of authentication or privacy is not high enough for it to continue, it SHOULD gracefully close the TLS connection immediately after the TLS negotiation has completed (see sections 5.4.1 and 5.5.2 below). If the client decides to continue, it MAY attempt to Start TLS again, it MAY send an unbind request, or it MAY send any other LDAP request.

5.3.5. Assertion of Client's Authorization Identity

The client MAY, upon receipt of a Start TLS response indicating success, assert that a specific authorization identity be utilized in determining the client's authorization status. The client accomplishes this via an LDAP Bind request specifying a SASL mechanism of "EXTERNAL" [SASL] (see section 5.5.1.2 below).

5.3.6. Server Identity Check

The client MUST check its understanding of the server's hostname against the server's identity as presented in the server's Certificate message, in order to prevent man-in-the-middle attacks.

Matching is performed according to these rules:

- The client MUST use the server hostname it used to open the LDAP connection as the value to compare against the server name as expressed in the server's certificate. The client MUST NOT use the server's canonical DNS name or any other derived form of name.

- If a subjectAltName extension of type dNSName is present in the certificate, it SHOULD be used as the source of the server's identity.

- Matching is case-insensitive.

- The "*" wildcard character is allowed. If present, it applies only to the left-most name component.

For example, *.bar.com would match a.bar.com and b.bar.com, but it would not match a.x.bar.com nor would it match bar.com. If more than one identity of a given type is present in the certificate (e.g. more than one dNSName name), a match in any one of the set is considered acceptable.

If the hostname does not match the dNSName-based identity in the certificate per the above check, user-oriented clients SHOULD either notify the user (clients MAY give the user the opportunity to continue with the connection in any case) or terminate the connection and indicate that the server's identity is suspect. Automated clients SHOULD close the connection, returning and/or logging an error indicating that the server's identity is suspect.

Beyond the server identity checks described in this section, clients SHOULD be prepared to do further checking to ensure that the server is authorized to provide the service it is observed to provide. The client MAY need to make use of local policy information.

5.3.7. Refresh of Server Capabilities Information

The client MUST refresh any cached server capabilities information (e.g. from the server's root DSE; see section 3.4 of [LDAPv3]) upon TLS session establishment. This is necessary to protect against active-intermediary attacks that may have altered any server capabilities information retrieved prior to TLS establishment. The server MAY advertise different capabilities after TLS establishment.

5.4. Closing a TLS Connection

Two forms of TLS connection closure--graceful and abrupt--are supported.

5.4.1. Graceful Closure

Either the client or server MAY terminate the TLS connection on an LDAP association by sending a TLS closure alert. This will leave the LDAP association intact.

Before closing a TLS connection, the client MUST [RGH10]either wait for any outstanding LDAP operations to complete, or explicitly abandon them [LDAPv3].

After the initiator of a close has sent a TLS closure alert, it MUST discard any TLS messages until it has received a TLS closure alert from the other party. It will cease to send TLS Record Protocol PDUs, and following the receipt of the alert, MAY send and receive LDAP PDUs.

The other party, if it receives a TLS closure alert, MUST immediately transmit a TLS closure alert. It will subsequently cease to send TLS Record Protocol PDUs, and MAY send and receive LDAP PDUs.

5.4.2. Abrupt Closure

Either the client or server MAY abruptly close the entire LDAP association and any TLS connection established on it by dropping the underlying TCP connection. In this circumstance, a server MAY send the client a Notice of Disconnection [LDAPv3] before dropping the TCP connection.

5.5. Effects of TLS on a Client's Authorization Identity

This section describes the effects on a client's authorization identity brought about by establishing TLS on an LDAP association. The default effects are described first, and next the facilities for client assertion of authorization identity are discussed including error conditions. Lastly, the effects of closing the TLS connection are described.

Authorization identities and related concepts are described in Appendix B.

5.5.1. TLS Connection Establishment Effects

5.5.1.1. Default Effects

Upon establishment of the TLS connection onto the LDAP association, any previously established authentication and authorization identities MUST remain in force, including anonymous state. This holds even in the case where the server requests client authentication via TLS -- e.g. requests the client to supply its certificate during TLS negotiation (see [TLS]).

5.5.1.2. Client Assertion of Authorization Identity

A client MAY either implicitly request that its LDAP authorization identity be derived from its authenticated TLS credentials or it MAY explicitly provide an authorization identity and assert that it be used in combination with its authenticated TLS credentials. The former is known as an implicit assertion, and the latter as an explicit assertion.

5.5.1.2.1. Implicit Assertion

An implicit authorization identity assertion is accomplished after TLS establishment by invoking a Bind request of the SASL form using the "EXTERNAL" mechanism name [SASL, LDAPv3] that SHALL NOT include the optional credentials octet string (found within the SaslCredentials sequence in the Bind Request). The server will derive the client's authorization identity from the authentication identity supplied in the client's TLS credentials (typically a public key certificate) according to local policy. The underlying mechanics of how this is accomplished are implementation specific.

5.5.1.2.2. Explicit Assertion

An explicit authorization identity assertion is accomplished after TLS establishment by invoking a Bind request of the SASL form using the "EXTERNAL" mechanism name [SASL, LDAPv3] that SHALL include the credentials octet string. This string MUST be constructed as documented in section 4.5.

5.5.1.2.3. Error Conditions

For either form of assertion, the server MUST verify that the client's authentication identity as supplied in its TLS credentials is permitted to be mapped to the asserted authorization identity. The server MUST reject the Bind operation with an invalidCredentials resultCode in the Bind response if the client is not so authorized.

Additionally, with either form of assertion, 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 [IPSEC]), 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.

After the above Bind operation failures, any client authentication and authorization state of the LDAP association is lost, so the LDAP association is in an anonymous state after the failure. TLS connection state is unaffected, though a server MAY end the TLS connection, via a TLS close_notify message, based on the Bind failure (as it MAY at any time).

5.5.2. TLS Connection Closure Effects

Closure of the TLS connection MUST cause the LDAP association to move to an anonymous authentication and authorization state regardless of the state established over TLS and regardless of the authentication and authorization state prior to TLS connection establishment.

6. LDAP Association State Transition Tables

To comprehensively diagram the various authentication and TLS states through which an LDAP association may pass, this section provides a state transition table to represent a state diagram for the various states through which an LDAP association may pass during the course of its existence and the actions that cause these changes in state.

6.1. LDAP Association States

The following table lists the valid LDAP association states and provides a description of each state. The ID for each state is used in the state transition table in section 6.4.

ID	State Description
S1	no Auth ID no AuthZ ID TLS: no Creds OFF
S2	no Auth ID no AuthZ ID TLS: no Creds ON
S3	no Auth ID no AuthZ ID TLS: Creds Auth ID "I", ON.
S4	Auth ID = Xn AuthZ ID= Yn [TLS: no Creds, Off]
S5	Auth ID = Xn AuthZ ID= Yn [TLS: no Creds, On]
S6	Auth ID = Xn AuthZ ID= Yn

```
       [TLS: Creds Auth ID "I", On]
   S7 Auth ID = I
       AuthZ ID= Jn
```

	[TLS: Creds Auth ID "I", On]
S8	Auth ID = I AuthZ ID= is based on "I" [TLS: Creds Auth ID "I", On]

6.2. Actions that Affect LDAP Association State

The following table lists the actions that can affect the state of an LDAP association. The ID for each action is used in the state transition table in section 6.4.

ID	Action
A1	Client binds anonymously
A2	Error: Inappropriate authentication
A3	Client or Server: close TLS connection (section 5.5.2)
A4	Client StartTLS Server: client auth NOT required
A5	Client: StartTLS Server: client creds requested Client: {TLS creds: Auth ID "I"]
A6	Client: Bind w/simple password or SASL mechanism (e.g. DIGEST-MD5 password, Kerberos, etc. û except EXTERNAL [Auth ID "X" maps to AuthZ ID "Y"]
A7	Client Binds SASL EXTERNAL w/ credentials: AuthZ ID "J" [Explicit Assertion (section 5.5.1.2.2)]
A8	Client Bind SASL EXTERNAL w/saslcredentials: NULL [Implicit Assertion (section 5.5.1.2.1)]

6.3. Decisions Used in Making LDAP Association State Changes

Certain changes in the state of an LDAP association are only allowed if the server can affirmatively answer a question. These questions are applied as part of the criteria for allowing or disallowing a state change in the state transition table in section 6.4.

ID	Decision Question
D1	Can TLS Credentials Auth ID "I" be mapped to AuthZ ID "J"?
D2	Can a valid AuthZ ID "J" be derived from TLS Credentials Auth ID "I"?

6.4. LDAP Association State Transition Table

The LDAP Association table below lists the valid states for an LDAP association and the actions that could affect them. For any given row in the table, the Current State column gives the state of an LDAP association, the Action column gives an action that could affect the state of an LDAP assocation, and the Next State column gives the resulting state of an LDAP association after the action occurs.

The initial state for the state machine described in this table is S1.

Current		Next	
State	Action	State	Comment
S1	A1	S1	
S1	A2	S1	Error: Inappropriate authentication
S1	A4	S2	
S1	A5	S3	
S1	A6	S4	
S2	A1	S2	
S2	A2	S2	Error: Inappropriate authentication
S2	A3	S1	
S2	A6	S5	
S3	A1	S3	
S3	A3	S1	
S3	A6	S6	
S3	A7 and D1=NO	S3	Error: InvalidCredentials
S3	A7 and D1=YES	S7	
S3	A8 and D2=NO	S3	Error: InvalidCredentials
S3	A8 and D2=YES	S8	
S4	A6	S4	
S4	A2	S4	Error: Inappropriate Authentication
S4	A3	S1	
S4	A4	S5	
S4	A5	S6	
S5	A6	S5	
S5	A2	S5	Error: Inappropriate Authentication
S5	A3	S1	
S5	A1	S2	
S6	A6	S6	
S6	A3	S1	
S6	A1	S3	
S6	A7 and D1=NO	S6	Error: InvalidCredentials
S6	A7 and D1=YES	S10	
S6	A8 and D2=NO	S6	Error: InvalidCredentials
S6	A8 and D2=YES	S8	
S7	A1	S3	
S7	A3	S1	
S7	A6	S6	
S7	A7	S7	
S7	A8 and D2=NO	S3	Error: InvalidCredentials
S7	A8 and D2=YES	S8	
S8	A1	S3	
S8	A3	S1	
S8	A8	S8	

S8	A7 and D1=NO	S6	Error: InvalidCredentials
S8	A7 and D1=YES	S7	
S8	A6	S6	

7. Anonymous Authentication

Directory operations that modify entries or access protected attributes or entries generally require client authentication. Clients that do not intend to perform any of these operations typically use anonymous authentication. Servers SHOULD NOT allow clients with anonymous authentication to modify directory entries or access sensitive information in directory entries.

LDAP implementations MUST support anonymous authentication, as defined in section 7.1.

LDAP implementations MAY support anonymous authentication with TLS, as defined in section 7.2.

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.

7.1. Anonymous Authentication Procedure

An LDAPv3 client that has not successfully completed a bind operation on a connection is anonymously authenticated. See section 4.3.3.

An LDAP client MAY also choose to explicitly bind anonymously. A client that wishes to do so MUST choose the simple authentication option in the Bind Request (see section 4.1) and set the password to be of zero length. (This is often done by LDAPv2 clients.) Typically the name is also of zero length.

7.2. Anonymous Authentication and TLS

An LDAP client MAY use the Start TLS operation (section 5) to negotiate the use of TLS security [TLS]. 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 11.

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.

8. Password-based authentication

8.1. Simple authentication

The LDAP "simple" authentication choice is not suitable for authentication in environments where there is no network or transport layer confidentiality. LDAP implementations SHOULD support authentication with the "simple" authentication choice when the connection is protected against eavesdropping using TLS, as defined in section 7.3. LDAP implementations SHOULD NOT support authentication with the "simple" authentication choice unless the data on the connection is protected using TLS or other privacy and data-integrity protection.

8.2. Digest Authentication

LDAP implementations MUST support authentication with a password using the DIGEST-MD5 SASL mechanism for password protection.

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 [RFC2831], 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 [RFC2831]. 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 [RFC2831]. 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.

8.3. "simple" authentication choice under TLS encryption

Following the negotiation of an appropriate TLS ciphersuite providing connection confidentiality [6], a client MAY authenticate to a directory that supports the simple authentication choice by performing a simple bind operation.

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 11.

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 a DN , and the "simple" authentication choice, containing a password.

8.3.1. "simple" Authentication Choice

DSAs that map the DN sent in the bind request to a directory entry with a userPassword attribute 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.

8.4. Other authentication choices with TLS

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

9. Certificate-based authentication

LDAP implementations SHOULD support authentication via a client certificate in TLS, as defined in section 8.1.

9.1. 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 that issued the userÆs certificate must be sufficiently trusted by the directory server in order for the server to 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 the server MUST perform a private key-based encryption, proving it has the private key associated with the certificate.

In deployments that require protection of sensitive data in transit, the client and server MUST negotiate a ciphersuite that contains a bulk encryption algorithm of appropriate strength. Recommendations of cipher suites are given in section 11.

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.

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 day of CPU time on a standard CPU in 2000). These ciphersuites are NOT RECOMMENDED for use in confidentiality protection of passwords or data. Client and server implementers 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 TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA and MAY support other ciphersuites offering equivalent or better protection.

11. 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.

11.1. Start TLS Security Considerations

The goals of using the TLS protocol with LDAP are to ensure connection confidentiality and integrity, and to optionally provide for authentication. TLS expressly provides these capabilities, as described in [TLS].

All security gained via use of the Start TLS operation is gained by the use of TLS itself. The Start TLS operation, on its own, does not provide any additional security.

The use of TLS does not provide or ensure for confidentiality and/or non-repudiation of the data housed by an LDAP-based directory server. Nor does it secure the data from inspection by the server administrators. Once established, TLS only provides for and ensures confidentiality and integrity of the operations and data in transit over the LDAP association, and only if the implementations on the client and server support and negotiate it.

The level of security provided though the use of TLS depends directly on both the quality of the TLS implementation used and the style of usage of that implementation. Additionally, an active-intermediary attacker can remove the Start TLS extended operation from the supportedExtension attribute of the root DSE. Therefore, both parties SHOULD independently ascertain and consent to the security level achieved once TLS is established and before beginning use of the TLS connection. For example, the security level of the TLS connection might have been negotiated down to plaintext.

Clients SHOULD either warn the user when the security level achieved does not provide confidentiality and/or integrity protection, or be configurable to refuse to proceed without an acceptable level of security.

Client and server implementors SHOULD take measures to ensure proper protection of credentials and other confidential data where such measures are not otherwise provided by the TLS implementation.

Server implementors SHOULD allow for server administrators to elect whether and when connection confidentiality and/or integrity is required, as well as elect whether and when client authentication via TLS is required.

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

12. Acknowledgements

This document combines information originally contained in RFC 2829, RFC 2830 and portions of RFC 2251. The author acknowledges the work of Harald Tveit Alvestrand, Jeff Hodges, Tim Howes, Steve Kille, RL "Bob" Morgan , and Mark Wahl, each of whom authored one or more of these documents. RFC 2829 and RFC 2830 were products of the IETF LDAPEXT Working Group. RFC 2251 was a product of the ASID Working Group.

This document is based upon input of the IETF LDAP Revision working group. The contributions of its members is greatly appreciated.

13. Bibliography

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

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

[LDAPDN] Zeilenga, Kurt D., "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", draft-ietf-ldapbis-dn-06.txt, July, 2001.

[LDAPv3] Wahl, M., Kille S. and T. Howes, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC2828] Shirey, R., "Internet Security Glossary", RFC 2828, May 2000.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

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

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

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

14. Author's Address

Roger Harrison
Novell, Inc.
1800 S. Novell Place
Provo, UT 84606
+1 801 861 2642
roger_harrison@novell.com

15. Full Copyright Statement

Appendix A. Example Deployment Scenarios

The following scenarios are typical for LDAP directories on the Internet, and have different security requirements. (In the following discussion, "sensitive data" refers to information whose disclosure, alteration, destruction, or loss would adversely affect the interests or business of its owner or user. Also note that 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 data confidentiality for sensitive authentication information AND data integrity for all authentication information.

(3) A read-only directory containing no sensitive data; and the client needs to ensure the identity of the directory server and that the directory data is not modified while being returned from the server. A data origin authentication service AND data integrity service are required.

(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 data confidentiality for sensitive authentication information AND data integrity for all authentication information.

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

Appendix B. Authentication and Authorization: Definitions and Concepts

This appendix 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.

B.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.

B.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 [LDAPv3]). 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.

B.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.

B.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 [SASL]. 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.

Appendix C. RFC 2829 Change History

This appendix lists the changes made to the text of RFC 2829 in preparing this document.

C.0. General Editorial Changes

Version -00

- Changed other instances of the term LDAP to LDAPv3 where v3 of the protocol is implied. Also made all references to LDAPv3 use the same wording.

- Miscellaneous grammatical changes to improve readability.

- Made capitalization in section headings consistent.

Version -01

- Changed title to reflect inclusion of material from RFC 2830 and 2251.

C.1. Changes to Section 1

Version -01

- Moved conventions used in document to a separate section.

C.2. Changes to Section 2

Version -01

- Moved section to an appendix.

C.3. Changes to Section 3

Version -01

- Moved section to an appendix.

C.4 Changes to Section 4

Version -00

- Changed "Distinguished Name" to "LDAP distinguished name".

C.5. Changes to Section 5

Version -00

- Added the following sentence: "Servers SHOULD NOT allow clients with anonymous authentication to modify directory entries or access sensitive information in directory entries."

C.5.1. Changes to Section 5.1

Version -00

- Replaced the text describing the procedure for performing an anonymous bind (protocol) with a reference to section 4.2 of RFC 2251 (the protocol spec).

Version -01

- Brought text describing procedure for performing an anonymous bind from section 4.2 of RFC 2251 bis. This text will be removed from the draft standard version of that document.

C.6. Changes to Section 6.

Version -00

Reorganized text in section 6.1 as follows:

1. Added a new section (6.1) titled "Simple Authentication" and moved one of two introductory paragraphs for section 6 into section 6.1. Added sentences to the paragraph indicating:

   a. simple authentication is not suitable for environments where confidentiality is not available.

   b. LDAP implementations SHOULD NOT support simple authentication unless confidentiality and data integrity mechanisms are in force.

2. Moved first paragraph of section 6 (beginning with "LDAP implementations MUST support authentication with a passwordà") to section on Digest Authentication (Now section 6.2).

C.6.1. Changes to Section 6.1.

Version -00 Renamed section to 6.2

- Added sentence from original section 6 indicating that the DIGEST-MD5 SASL mechanism is required for all conforming LDAPv3 implementations

C.6.2. Changes to Section 6.2

Version -00

- Renamed section to 6.3

- Reworded first paragraph to remove reference to user and the userPassword password attribute Made the first paragraph more general by simply saying that if a directory supports simple authentication that the simple bind operation MAY performed following negotiation of a TLS ciphersuite that supports confidentiality.

- Replaced "the name of the user's entry" with "a DN" since not all bind operations are performed on behalf of a "user."

- Added Section 6.3.1 heading just prior to paragraph 5.

- Paragraph 5: replaced "The server" with "DSAs that map the DN sent in the bind request to a directory entry with a userPassword attribute."

C.6.3. Changes to section 6.3.

Version -00

- Renamed to section 6.4.

C.7. Changes to section 7.

none

C.7.1. Changes to section 7.1.

Version -00

- Clarified the entity issuing a certificate by moving the phrase "to have issued the certificate" immediately after "Certification Authority."

C.8. Changes to section 8.

Version -00

- Removed the first paragraph because simple authentication is covered explicitly in section 6.

- Added section 8.1. heading just prior to second paragraph.

- Added section 8.2. heading just prior to third paragraph.

- Added section 8.3. heading just prior to fourth paragraph.

Version -01

- Moved entire section 8 of RFC 2829 into section 3.4 (Using SASL for Other Security Services) to bring material on SASL mechanisms together into one location.

C.9. Changes to section 9.

Version -00

- Paragraph 2: changed "EXTERNAL mechanism" to "EXTERNAL SASL mechanism."

- Added section 9.1. heading.

- Modified a comment in the ABNF from "unspecified userid" to "unspecified authz id".

- Deleted sentence, "A utf8string is defined to be the UTF-8 encoding of one or more ISO 10646 characters," because it is redundant.

- Added section 9.1.1. heading.

- Added section 9.1.2. heading.

Version -01

- Moved entire section 9 to become section 3.5 so that it would be with other SASL material.

C.10. Changes to Section 10.

Version -00

- Updated reference to cracking from a week of CPU time in 1997 to be a day of CPU time in 2000.

- Added text: "These ciphersuites are NOT RECOMMENDED for use... and server implementers SHOULD" to sentence just prior the second list of ciphersuites.

- Added text: "and MAY support other ciphersuites offering equivalent or better protection," to the last paragraph of the section.

C.11. Changes to Section 11.

Version -01

- Moved to section 3.6 to be with other SASL material.

C.12. Changes to Section 12.

Version -00

- Inserted new section 12 that specifies when SASL protections begin following SASL negotiation, etc. The original section 12 is renumbered to become section 13.

Version -01

- Moved to section 3.7 to be with other SASL material.

C.13. Changes to Section 13 (original section 12).

None

Appendix D. RFC 2830 Change History

This appendix lists the changes made to the text of RFC 2830 in preparing this document.

D.0. General Editorial Changes

- Material showing the PDUs for the Start TLS response was broken out into a new section.

- The wording of the definition of the Start TLS request and Start TLS response was changed to make them parallel. NO changes were made to the ASN.1 definition or the associated values of the parameters.

- A separate section heading for graceful TLS closure was added for parallelism with section on abrupt TLS closure.

Appendix E. RFC 2251 Change History

This appendix lists the changes made to the text of RFC 2251 in preparing this document.

E.0. General Editorial Changes

- All material from section 4.2 of RFC 2251 was moved into this document.

- A new section was created for the Bind Request

- Section 4.2.1 of RFC 2251 (Sequencing Bind Request) was moved after the section on the Bind Response for parallelism with the presentation of the Start TLS operations. The section was also subdivided to explicitly call out the various effects being described within it.

- All SASL profile information from RFC 2829 was brought within the discussion of the Bind operation (primarily sections 4.4 - 4.7).

Appendix F. Change History to Combined Document

F.1. Changes for draft-ldap-bis-authmeth-02.doc

Section 1.
- Added glossary of terms and added sub-section headings

Section 2.
- Clarified security mechanisms 3, 4, & 5 and brought language in line with IETF security glossary.

Section 3.
- Brought language in requirement (3) in line with security glossary.

- Clarified that information fetched prior to initiation of TLS negotiation must be discarded

-Clarified that information fetched prior to initiation of SASL negotiation must be discarded

- Rewrote paragraph on SASL negotiation requirements to clarify intent

Section 4.4.
- Added stipulation that sasl choice allows for any SASL mechanism not prohibited by this document. (Resolved conflict between this statement and one that prohibited use of ANONYMOUS and PLAIN SASL mechanisms.)

Section 5.3.6
- Added a.x.bar.com to wildcard matching example on hostname check.

Section 6
- Added LDAP Association State Transition Tables to show the various states through which an LDAP association may pass along with the actions and decisions required to traverse from state to state.

Appendix A

- Brought security terminology in line with IETF security glossary throughout the appendix.

General

- Added references to other LDAP standard documents, to sections within the document, and fixed broken references.

- General editorial changesùpunctuation, spelling, formatting, etc.

Appendix G. Issues to be Resolved

This appendix lists open questions and issues that need to be resolved before work on this document is deemed complete.

G.1.

Section 1 lists 6 security mechanisms that can be used by LDAP servers. I'm not sure what mechanism 5, "Resource limitation by means of administrative limits on service controls" means.

Status: resolved. Changed wording to "administrative service limits" to clarify meaning.

G.2.

Section 2 paragraph 1 defines the term, "sensitive." Do we want to bring this term and other security-related terms in alignment with usage with the IETF security glossary (RFC 2828)?

Status: resolved. WG input at IETF 51 was that we should do this, so the appropriate changes have been made.

G.3.

Section 2, deployment scenario 2: What is meant by the term "secure authentication function?"

Status: resolved. Based on the idea that a "secure authentication function" could be provided by TLS, I changed the wording to require data confidentiality for sensitive authentication information and data integrity for all authentication information.

G.4.

Section 3, deployment scenario 3: What is meant by the phrase, "directory data is authenticated by the server?"

Status: resolved. I interpreted this to mean the ability to ensure the identity of the directory server and the integrity of the data sent from that server to the client, and explictly stated such.

G.5.

Section 4 paragraph 3: What is meant by the phrase, "this means that either this data is useless for faking authentication (like the Unix "/etc/passwd" file format used to be)?"

G.6.

Section 4 paragraph 7 begins: "For a directory needing session protection..." Is this referring to data confidentiality or data integrity or both?

Status: resolved. Changed wording to say, "For a directory needing data security (both data integrity and data confidentiality)..."

G.7.

Section 4 paragraph 8 indicates that "information about the server fetched fetched prior to the TLS negotiation" must be discarded. Do we want to explicitly state that this applies to information fetched prior to the *completion* of the TLS negotiation or is this going too far?

Status: resolved. Based on comments in the IETF 51 LDAPBIS WG meeting, this has been changed to explicitly state, "fetched prior to the initiation of the TLS negotiation..."

G.8.

Section 4 paragraph 9 indicates that clients SHOULD check the supportedSASLMechanisms list both before and after a SASL security layer is negotiated to ensure that they are using the best available security mechanism supported mutually by the client and server. A note at the end of the paragraph indicates that this is a SHOULD since there are environments where the client might get a list of supported SASL mechanisms from a different trusted source.

I wonder if the intent of this could be restated more plainly using one of these two approaches (I've paraphrased for the sake of brevity):

> Approach 1: Clients SHOULD check the supportedSASLMechanisms list both before and after SASL negotiation or clients SHOULD use a different trusted source to determine available supported SASL mechanisms.
>
> Approach 2: Clients MUST check the supportedSASLMechanisms list both before and after SASL negotiation UNLESS they use a different trusted source to determine available supported SASL mechanisms.

Status: Resolved. WG input at IETF 51 was that Approach 1 was probably best. I ended up keeping the basic structure similar to the original to meet this intent.

G.9.

Section 6.3.1 states: "DSAs that map the DN sent in the bind request to a directory entry with a userPassword attribute will... compare [each value in the named user's entry]... with the presented password." This implies that this this applies only to user entries with userPassword attributes. What about other types of entries that might allow passwords and might store in the password information in other attributes? Do we want to make this text more general?

G.10 userPassword and simple bind

We need to be sure that we don't require userPassword to be the only attribute used for authenticating via simple bind. (See 2251 sec 4.2 and authmeth 6.3.1. Work with Jim Sermersheim on resolution to this. On publication state something like: "This is the specific implementation of what we discussed in our general reorg conversation on the list." (Source: Kurt Zeilenga)

G.11. Meaning of LDAP Association

The original RFC 2830 uses the term "LDAP association" in describing a connection between an LDAP client and server regardless of the state of TLS on that connection. This term needs to be defined or possibly changed.

Status: Resolved. at IETF 51 Bob Morgan indicated that the term "LDAP association" was intended to distinguish the LDAP-level connection from the TLS-level connection. This still needs to be clarified somewhere in the draft. Added "LDAP association" to a glossary in section 1.

G.12. Is DIGEST-MD5 mandatory for all implementations?

Reading 2829bis I think DIGEST-MD5 is mandatory ONLY IF your server supports password based authentication...but the following makes it sound mandatory to provide BOTH password authentication AND DIGEST-MD5:

"6.2. Digest authentication

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

The thing is for acl it would be nice (though not critical) to be able to default the required authentication level for a subject to a single "fairly secure" mechanism--if there is no such mandatory authentication scheme then you cannot do that. (Source: Rob Byrne)

G.13. Ordering of authentication levels requested

Again on the subject of authentication level, is it possible to define an ordering on authentication levels which defines their relative "strengths" ? This would be useful in acl as you could say things like"a given aci grants access to a given subject at this authentication level AND ABOVE". David Chadwick raised this before in the context of denying access to a subject at a given authentication level, in which case he wanted to express "deny access to this subject at this authentication level AND TO ALL IDENTITIES AUTHENTICATED BELOW THAT LEVEL". (Source: Rob Byrne)

G.14. Document vulnerabilities of various mechanisms

While I'm here...in 2829, I think it would be good to have some comments or explicit reference to a place where the security properties of the particular mandatory authentication schemes are outlined. When I say "security properties" I mean stuff like "This scheme is vulnerable to such and such attacks, is only safe if the key size is > 50, this hash is widely considered the best, etc...". I think an LDAP implementor is likely to be interested in that information, without having to wade through the security RFCs. (Source: Rob Byrne)

G.15. Include a StartTLS state transition table

The pictoral representation it is nominally based on is here (URL possibly folded):

http://www.stanford.edu/~hodges/doc/LDAPAssociationStateDiagram-1999-12-14.html

(Source: Jeff Hodges)

Status: Table provided. Final review of content for accuracy is still needed.

G.16. Empty sasl credentials question

I spent some more time looking microscopically at ldap-auth-methods and ldap-ext-tls drafts. The drafts say that the credential must have the form dn:xxx or u:xxx or be absent, and although they don't say what to do in the case of an empty octet string I would say that we could send protocolError (claim it is a bad PDU).

There is still the question of what to do if the credential is 'dn:' (or 'u:') followed by the empty string. (Source: ariel@columbia.edu via Jeff Hodges)

G.17. Hostname check from MUST to SHOULD?

I am uneasy about the hostname check. My experience from PKI with HTTP probably is a contributing factor; we have people using the short hostname to get to a server which naturally has the FQDN in the certificate, no end of problems. I have a certificate on my laptop which has the FQDN for the casse when the system is on our Columbia network with a fixed IP; when I dial in however, I have some horrible dialup name, and using the local https server becomes annoying. Issuing a certificate in the name 'localhost' is not a solution! Wildcard match does not solve this problem. For these reasons I am inclined to argue for 'SHOULD' instead of 'MUST' in paragraph...

Also, The hostname check against the name in the certificate is a very weak means of preventing man-in-the-middle attacks; the proper solution is not here yet (SecureDNS or some equivalent). Faking out DNS is not so hard, and we see this sort of thing in the press on a pretty regular basis, where site A hijacks the DNS server for site B and gets all their requests. Some mention of this should be made in the draft. (Source: ariel@columbia.edu via Jeff Hodges)

G.18. Must SASL DN exist in the directory?

If the 'dn:' form of sasl creds is used, is it the intention of the draft(ers) that this DN must exist in the directory and the client will have the privileges associated with that entry, or can the server map the sasl DN to perhaps some other DN in the directory, in an implementation-dependent fashion?

We already know that if *no* sasl credentials are presented, the DN or altname in the client certificate may be mapped to a DN in an implementation-dependent fashion, or indeed to something not in the directory at all. (Right?) (Source: ariel@columbia.edu via Jeff Hodges)

G.19. DN used in conjunction with SASL mechanism

We need to specify whether the DN field in Bind operation can/cannot be used when SASL mechanism is specified. (source: RL Bob)

G.20. Bind states

Differences between unauthenticated and anonymous. four states you can get into. One is completely undefined (this is now explicitly called out in document). This text needs to be moved from RFC2251bis to this draft. (source: Jim Sermersheim)