LDAP Authentication Password Schema

Status of this Memo

Copyright Notice

Abstract

   This document describes schema in support of user/password
   authentication in a LDAP (Lightweight Directory Access Protocol)
   directory including the authPassword attribute type. This attribute
   type holds values derived from the user's password(s) (commonly using
   cryptographic strength one-way hash). authPassword is intended to
   used instead of userPassword.

1. Background and Intended Use

   The userPassword attribute type [RFC2256] is intended to be used to
   support the LDAP [RFC2251] "simple" bind operation. However, values
   of userPassword must be clear text passwords. It is often desirable
   to store values derived from the user's password(s) instead of actual
   passwords.

   The authPassword attribute type is intended to be used to store
   information used to implement simple password based authentication.
   The attribute type may be used by LDAP servers to implement the LDAP
   Bind operation's "simple" authentication method.

   The attribute type supports multiple storage schemes. A matching
   rule is provided for use with extensible search filters to allow
   clients to assert that a clear text password "matches" one of the
   attribute's values.

   Storage schemes often use cryptographic strength one-way hashing.
   Though the use of one-way hashing reduces the potential that exposed
   values will allow unauthorized access to the Directory (unless the

hash algorithm/implementation is flawed), the hashing of passwords is
intended to be as an additional layer of protection. It is
RECOMMENDED that hashed values be protected as if they were clear
text passwords.

This attribute may be used in conjunction with server side password
generation mechanisms (such as the LDAP Password Modify [RFC3062]
extended operation).

Access to this attribute may governed by administrative controls such
as those which implement password change policies.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are
to be interpreted as described in RFC 2119 [RFC2119].

2. Schema Definitions

The following schema definitions are described in terms of LDAPv3
Attribute Syntax Definitions [RFC2252] with specific syntax detailed
using Augmented BNF [RFC2234].

2.1. authPasswordSyntax

```
( 1.3.6.1.4.1.4203.1.1.2
  DESC 'authentication password syntax' )
```

Values of this syntax are encoded according to:

```
authPasswordValue = w scheme s authInfo s authValue w
scheme = %x30-39 / %x41-5A / %x2D-2F / %x5F
      ; 0-9, A-Z, "-", ".", "/", or "_"
authInfo = schemeSpecificValue
authValue = schemeSpecificValue
        schemeSpecificValue = *( %x21-23 / %x25-7E )
      ; printable ASCII less "$" and " "
s = w SEP w
w = *SP
SEP = %x24 ; "$"
SP = %x20 ; " " (space)
```

where scheme describes the mechanism and authInfo and authValue are a
scheme specific. The authInfo field is often a base64 encoded salt.
The authValue field is often a base64 encoded value derived from a
user's password(s). Values of this attribute are case sensitive.

   Transfer of values of this syntax is strongly discouraged where the
   underlying transport service cannot guarantee confidentiality and may
   result in disclosure of the values to unauthorized parties.

   This document describes a number of schemes, as well as requirements
   for the scheme naming, in section 3.

2.2. authPasswordExactMatch

      (1.3.6.1.4.1.4203.1.2.2
        NAME 'authPasswordExactMatch'
        DESC 'authentication password exact matching rule'
        SYNTAX 1.3.6.1.4.1.4203.1.1.2)

   This matching rule allows a client to assert that an asserted
   authPasswordSyntax value matches authPasswordSyntax values. It is
   meant to be used as the EQUALITY matching rule of attributes whose
   SYNTAX is authPasswordSyntax.

   The assertion is "TRUE" if there is an attribute value which has the
   same scheme, authInfo, and authValue components as the asserted
   value; "FALSE" if no attribute value has the same components as the
   asserted value; and "Undefined" otherwise.

2.3. authPasswordMatch

      (1.3.6.1.4.1.4203.1.2.3
        NAME 'authPasswordMatch'
        DESC 'authentication password matching rule'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40{128})

   This matching rule allows a client to assert that a password matches
   values of authPasswordSyntax using an extensibleMatch filter
   component. Each value is matched per its scheme. The assertion is
   "TRUE" if one or more attribute values matches the asserted value,
   "FALSE" if all values do not matches, and "Undefined" otherwise.

   Servers which support use of this matching rule SHOULD publish
   appropriate matchingRuleUse values per [RFC2252], 4.4.

   Transfer of authPasswordMatch assertion values is strongly
   discouraged where the underlying transport service cannot guarantee
   confidentiality and may result in disclosure of the values to
   unauthorized parties.

2.4. supportedAuthPasswordSchemes

        (1.3.6.1.4.1.4203.1.3.3
          NAME 'supportedAuthPasswordSchemes'
          DESC 'supported password storage schemes'
          EQUALITY caseExactIA5Match
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{32}
          USAGE dSAOperation)

   The values of this attribute are names of supported authentication
   password schemes which the server supports. The syntax of a scheme
   name is described in section 2.1. This attribute may only be present
   in the root DSE. If the server does not support any password
   schemes, this attribute will not be present.

2.5. authPassword

        (1.3.6.1.4.1.4203.1.3.4 NAME 'authPassword'
          DESC 'password authentication information'
          EQUALITY 1.3.6.1.4.1.4203.1.2.2
          SYNTAX 1.3.6.1.4.1.4203.1.1.2)

   The values of this attribute are representative of the user's
   password(s) and conform to the authPasswordSyntax described in 2.1.
   The values of this attribute may be used for authentication purposes.

   Transfer of authPassword values is strongly discouraged where the
   underlying transport service cannot guarantee confidentiality and may
   result in disclosure of the values to unauthorized parties.

2.6. authPasswordObject

        (1.3.6.1.4.1.4203.1.4.7 NAME 'authPasswordObject'
          DESC 'authentication password mix in class'
          MAY 'authPassword'
          AUXILIARY)

   Entries of this object class may contain authPassword attribute
   types.

3. Schemes

   This section describes the "MD5" and "SHA1" schemes. Other schemes
   may be defined by other documents. Schemes which are not described
   in an RFC SHOULD be named with a leading "X-" to indicate they are a
   private or implementation specific scheme, or may be named using the
   dotted-decimal representation [RFC2252] of an OID assigned to the
   scheme.

3.1. MD5 scheme

   The MD5 [RFC1321] scheme name is "MD5".

   The authValue is the base64 encoding of an MD5 digest of the
   concatenation the user password and salt. The base64 encoding of the
   salt is provided in the authInfo field. The salt MUST be at least 64
   bits long. Implementations of this scheme MUST support salts up to
   128 bits in length.

   Example:
      Given a user "joe" who's password is "mary" and a salt of "salt",
      the authInfo field would be the base64 encoding of "salt" and the
      authValue field would be the base64 encoding of the MD5 digest of
      "marysalt".

   A match against an asserted password and an attribute value of this
   scheme SHALL be true if and only if the MD5 digest of concatenation
   of the asserted value and the salt is equal to the MD5 digest
   contained in AuthValue. The match SHALL be undefined if the server
   is unable to complete the equality test for any reason. Otherwise
   the match SHALL be false.

   Values of this scheme SHOULD only be used to implement simple
   user/password authentication.

3.2. SHA1 scheme

   The SHA1 [SHA1] scheme name is "SHA1".

   The authValue is the base64 encoding of a SHA1 digest of the
   concatenation the user password and the salt. The base64 encoding of
   the salt is provided in the authInfo field. The salt MUST be at
   least 64 bits long. Implementations of this scheme MUST support
   salts up to 128 bits in length.

   Example:
      Given a user "joe" who's password is "mary" and a salt of "salt",
      the authInfo field would be the base64 encoding of "salt" and the
      authValue field would be the base64 encoding of the SHA1 digest of
      "marysalt".

   A match against an asserted password and an attribute value of this
   scheme SHALL be true if and only if the SHA1 digest of concatenation
   of the asserted value and the salt is equal to the SHA1 digest
   contained in AuthValue. The match SHALL be undefined if the server
   is unable to complete the equality test for any reason. Otherwise
   the match SHALL be false.

      Values of this scheme SHOULD only be used to implement simple
      user/password authentication.

4. Implementation Issues

   For all implementations of this specification:

      Servers MAY restrict which schemes are used in conjunction with a
      particular authentication process but SHOULD use all values of
      selected schemes. If the asserted password matches any of the
      stored values, the asserted password SHOULD be considered valid.
      Servers MAY use other authentication storage mechanisms, such as
      userPassword or an external password store, in conjunction with
      authPassword to support the authentication process.

      Servers that support simple bind MUST support the SHA1 scheme and
      SHOULD support the MD5 scheme.

      Servers SHOULD NOT publish values of authPassword nor allow
      operations which expose authPassword values or AuthPasswordMatch
      assertions to unless confidentiality protection is in place.

      Clients SHOULD NOT initiate operations which provide or request
      values of authPassword or make authPasswordMatch assertions unless
      confidentiality protection is in place.

      Clients SHOULD NOT assume that a successful AuthPasswordMatch,
      whether by compare or search, is sufficient to gain directory
      access. The bind operation MUST be used to authenticate to the
      directory.

5. Security Considerations

   This document describes how authentication information may be stored
   in a directory. Authentication information MUST be adequately
   protected as unintended disclosure will allow attackers to gain
   immediate access to the directory as described by [RFC2829].

   As flaws may be discovered in the hashing algorithm or with a
   particular implementation of the algorithm or values could be subject
   to various attacks if exposed, values of AuthPassword SHOULD be
   protected as if they were clear text passwords. When values are
   transferred, privacy protections, such as IPSEC or TLS, SHOULD be in
   place.

   Clients SHOULD use strong authentication mechanisms [RFC2829].

AuthPasswordMatch matching rule allows applications to test the
validity of a user password and, hence, may be used to mount an
attack. Servers SHOULD take appropriate measures to protect the
directory from such attacks.

Some password schemes may require CPU intensive operations. Servers
SHOULD take appropriate measures to protect against Denial of Service
attacks.

AuthPassword does not restrict an authentication identity to a single
password. An attacker who gains write access to this attribute may
store additional values without disabling the user's true
password(s). Use of policy aware clients and servers is RECOMMENDED.

The level of protection offered against various attacks differ from
scheme to scheme. It is RECOMMENDED that servers support scheme
selection as a configuration item. This allows for a scheme to be
easily disabled if a significant security flaw is discovered.

6. Acknowledgment

   This document borrows from a number of IETF documents and is based
   upon input from the IETF LDAPext working group.

7. Bibliography

   [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321,
             April 1992

   [RFC2219] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2234] Crocker, D., Editor, P. Overell, "Augmented BNF for Syntax
             Specifications: ABNF", RFC 2234, November 1997.

   [RFC2251] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory
             Access Protocol (v3)", RFC 2251, December 1997.

   [RFC2252] Wahl, M., Coulbeck, A., Howes, T., and S. Kille,
             "Lightweight Directory Access Protocol (v3): Attribute
             Syntax Definitions", RFC 2252, December 1997.

   [RFC2256] Wahl, A., "A Summary of the X.500(96) User Schema for use
             with LDAPv3", RFC 2256, December 1997.

   [RFC2307] Howard, L., "An Approach for Using LDAP as a Network
             Information Service", RFC 2307, March 1998.

   [RFC2829] Wahl, M., Alvestrand, H., Hodges, J. and R. Morgan,
              "Authentication Methods for LDAP", RFC 2829, June 2000.

   [RFC3062] Zeilenga, K., "LDAP Password Modify Extended Operation",
              RFC 3062, February 2001.

   [SHA1]   NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.

8. Author's Address

   Kurt D. Zeilenga
   OpenLDAP Foundation

   EMail: Kurt@OpenLDAP.org

9. Full Copyright Statement

Acknowledgement