Network Working Group
Internet-Draft
Updates: 2307 (if approved)
Intended status: Informational
Expires: September 14, 2013

M. Stroeder
Independent consultant

March 13, 2013

Lightweight Directory Access Protocol (LDAP): Hashed Attribute values for 'userPassword' draft-stroeder-hashed-userpassword-values-01

Abstract

This document describes the widely used syntax for storing hashed passwords in LDAP attribute 'userPassword'. Furthermore it points out some of the deficiencies of the approach. Its purpose is solely to document current practice, it does not define a new standard.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on September 14, 2013.

Copyright Notice

Table of Contents

1. Introduction 2
2. Syntax for attribute 'userPassword' 3
3. Implementation Issues 4
4. Acknowledgements 5
5. IANA Considerations 5
6. Security Considerations 5
7. References 6
7.1. Normative References 6
7.2. Informative References 6
Author's Address 7

1. Introduction

This document does not define a new standard. Its purpose is solely to correctly document what is widely implemented in LDAP servers and clients to avoid interop issues in case implementors have to support old legacy systems using this scheme.

Strictly speaking the 'userPassword' attribute type, intended to be used to support the LDAP [RFC4510] "simple" bind operation, was meant to only store clear text passwords [RFC4519].

Although [RFC3112] defined a more versatile password attribute 'authPassword' for storing hashed passwords this was not widely implemented in server and client implementations. Instead current LDAP deployments still rely on the password hashing scheme for attribute 'userPassword' introduced in [RFC2307] especially since this attribute type is directly used in various object classes.

The specification in [RFC2307] is missing some formal aspects potentially leading to interop issues. Furthermore new hash algorithms are used today by various implementors which were not mentioned in [RFC2307].

Therefore this document also updates [RFC2307] by fully specifying how to store hashed password values in attribute 'userPassword'

optionally using the SHA-2 hash algorithms [FIPS-180-4]. For this it focuses on documenting already implemented server and client implementations. The password hashing scheme {crypt} was left out from the syntax definition because there are many platform-specific variants of possible values.

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 [RFC2119].

This document is being discussed on the ldapext@ietf.org mailing list.

2. Syntax for attribute 'userPassword'

userPassword values MUST be represented by following syntax:

```
userpasswordvalue  = cleartext-password / prefix b64-hashandsalt

prefix       = "{" scheme "}"
scheme = %x30-39 / %x41-5A / %x61-7a / %x2D-2F / %x5F
     ;0-9, A-Z, a-z, "-", ".", "/", or "_"

b64-hashandsalt = <base64 of hashandsalt>

hashandsalt = password-hash salt

password-hash = <digest of cleartext-password salt>

cleartext-password = %x00-FF

salt = %x00-FF
```

Field salt SHALL be treated as a randomely chosen sequence of bytes. The length of the salt SHOULD be at least 64 bits but other salt length MAY be recommended for specific hash algorithms. Implementations MUST be prepared of handling field salt of arbitrary length. salt MUST be empty for non-salted hashing schemes.

The field hashandsalt is generated by concatening the field password-hash and the salt. The field password-hash is generated by calculating the digest over the concatenation of password followed by salt.

b64-hashandsalt is generated by encoding hashandsalt according to the base64 algorithm as specified in [RFC4648].

The field scheme MUST be treated case-insensitive by all server and client implementations. Applications SHALL handle prefix "x-" in scheme just like any other scheme value without this prefix [RFC6648]. Other specifications MAY update this document by defining other values for scheme.

prefix	Description	Algorithm reference
{MD5}	MD-5 without salt	[RFC1321]
{SMD5}	salted MD-5	[RFC1321]
{SHA}	SHA-1 without salt	[FIPS-180-4]
{SSHA}	salted SHA-1	[FIPS-180-4]
{SHA256}	SHA-256 without salt	[FIPS-180-4]
{SSHA256}	salted SHA-256	[FIPS-180-4]
{SHA384}	SHA-384 without salt	[FIPS-180-4]
{SSHA384}	salted SHA-384	[FIPS-180-4]
{SHA512}	SHA-512 without salt	[FIPS-180-4]
{SSHA512}	salted SHA-512	[FIPS-180-4]

Table 1: Currently used hash schemes

3. Implementation Issues

All implementations SHOULD prepare textual strings used as field password like described for clear-text storage in section 2.41 in [RFC4519] before deriving a hash value from them. It is up to the implementation to determine what a textual password is.

Hashed 'userPassword' values are only suitable for directly comparing it to a clear-text password. They SHOULD NOT be used in challenge-response authentication schemes.

It is clear from the syntax specification that distinguishing clear-text passwords and hashed passwords is somewhat ambigous which is a well-known deficiency of this approach. Therefore implementing [RFC3112] is RECOMMENDED if a better solution is strictly required.

Implementations SHALL first check whether a value stored in attribute 'userPassword' adheres to the syntax specified above. Syntax checking SHALL be implemented by checking hash and optional salt following the algorithm-specific rules. Any value which do not adhere to this syntax MAY be treated as clear-text password by the DSA when processing a LDAP simple bind request or LDAP compare request.

Servers MAY provide local configuration items to limit the set of hash schemes to be processed and for completely disabling use of clear-text passwords in attribute 'userPassword'.

4. Acknowledgements

The syntax definition for 'userPassword' values in this document is based on and supersedes the specification in section 5.3 of [RFC2307] by L. Howard.

Some basics of the formal specification and security considerations are based on text in [RFC3112] by K. Zeilenga.

5. IANA Considerations

There are no identifiers defined herein to be reserved by IANA.

6. 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].

Hashed values in attribute 'userPassword' SHOULD be protected as if they were clear text passwords because they are subject to dictionary or other attacks and flaws may be discovered in the hashing algorithm or with a particular implementation of the algorithm.

Especially it is RECOMMENDED to avoid using hashing schemes based on MD-5 because of known weaknesses of this digest algorithm [RFC6151].

Applications SHOULD NOT use the non-salted password hash schemes and SHOULD use a sufficiently long salt value.

When values are transferred, privacy protections, such as IPSEC or TLS, SHOULD be in place.

Clients SHOULD use stronger authentication mechanisms like defined in [RFC5802].

Servers SHOULD use stronger credential storage mechanisms like defined in [RFC5803].

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

Using 'userPassword' attribute type description as defined in [RFC4519] 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). If a server supports defining additional local constraints to limit the count of attribute values it is RECOMMENDED to define such a constraint for 'userPassword' values to be limited to one without having to change the standard schema.

Use of policy aware clients and servers is RECOMMENDED (see example in [I-D.behera-ldap-password-policy]).

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.

7. References

7.1. Normative References

[FIPS-180-4] National Institute of Standards and Technology (NIST) , "Secure Hash Standard (SHS)", FIPS PUB 180-4, March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[RFC4519] Sciberras, A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

7.2. Informative References

[I-D.behera-ldap-password-policy]

Sermersheim, J., Poitou, L., and H. Chu, "Password Policy for LDAP Directories", draft-behera-ldap-password-policy-10 (work in progress), August 2009.

[I-D.hoffman-schneier-4270bis]
Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", draft-hoffman-schneier-4270bis-01 (work in progress), November 2012.

[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, May 2000.

[RFC3112] Zeilenga, K., "LDAP Authentication Password Schema", RFC 3112, May 2001.

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

[RFC5803] Melnikov, A., "Lightweight Directory Access Protocol (LDAP) Schema for Storing Salted Challenge Response Authentication Mechanism (SCRAM) Secrets", RFC 5803, July 2010.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

[RFC6648] Saint-Andre, P., Crocker, D., and M. Nottingham, "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", BCP 178, RFC 6648, June 2012.

Author's Address

Michael Stroeder
Independent consultant
Klauprechtstr. 11
Karlsruhe 76137
DE

Email: michael@stroeder.com
URI: http://www.stroeder.com