LDAP Extensions Working Group Bruce Greenblatt
Internet Draft Pat Richard
<draft-ietf-ldapext-sigops-04.txt>
Expires in six months

An LDAP Control and Schema for Holding Operation Signatures

Status of this Memo

Abstract

In many environments clients require the ability to validiate the source and integrity of information provided by the directory. This document describes an LDAP message control which allows for the retrieval of digitally signed information. This document defines an LDAP v3 based mechanism for signing directory operations in order to create a secure journal of changes that have been made to each directory entry. Both client and server based signatures are supported. An object class for subsequent retrieval are "journal entries" is also defined. This document specifies LDAP v3 controls that enable this functionality. It also defines an LDAP v3 schema that allows for subsequent browsing of the journal information.

1. Introduction

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

In many environments clients require the ability to validiate the source and integrity of information provided by the directory. This document describes an LDAP message control which allows for the retrieval of digitally signed information. The perspective of this document is that the origin of the information that is stored in LDAP v3 accessible directories is the LDAP v3 client that creates the information. The source and integrity of the information is guaranteed by allowing for the digital signing of the operations that make changes to entries in the directory. The source and integrity of an individual LDAP connection can be guaranteed by making use of an underlying session layer that provides such services, such as TLS. Note that the integrity of an individual connection does not, in and of itself guarantee the integrity of the data that comes across the connection. This is due to the fact that the LDAP server is only capable of providing information that it has stored. In distributed and replicated environments, the fact that an entry has been successfully retrieved from a server may not be completely reassuring, if the entry in question was replicated from an untrusted domain.

By making use of public key technology, and creating digitally signed transactions that are created by the LDAP v3 client as entries are created and modified, a complete journal of the history of the entry is available. Since each entry in the journal has been digitally signed with the private key of the creator, or modifier of the entry, the source and integrity of the directory entry can be validated by verifying the signature of each entry in the journal. Note that not all of the journal entries will have been signed by the same user.

1.1. Audit Trail Mechanism

Signed directory operations is a straightforward application of S/MIME technology that also leverages the extensible framework that is provided by LDAP version 3. LDAP version 3 is defined in [4], and S/MIME is defined in [2]. The security used in S/MIME is based in the definitions in [1]. The basic idea is that the submitter of an LDAP operation that changes the directory information includes an LDAP version 3 control that includes either a signature of the operation, or a request that the LDAP server sign the operation on the behalf of the

LDAP client. The result of the operation (in addition to the change of the directory information), is additional information that is attached to directory objects, that includes the audit trail of signed operations. The LDAP control is (OID = 1.2.840.113549.6.0.0):


```
SignedOperation ::= CHOICE {
     signbyServer   NULL,
     signatureIncluded   OCTET STRING
 }
```

If the SignatureIncluded CHOICE is used, then the OCTET string is just an S/MIME message of the multipart/signed variety, that is composed of a single piece, that is the signature of the directory operation. Multipart/signed MIME objects are defined in [3]. If the SignbyServer CHOICE us used, then the LDAP server creates the signature on behalf of the client, using its own identity and not the identity of the client, in order to produce the audit trail entry. In either case the successful result of processing the control is the creation of additional information in the directory entry that is being modified or created. The signature of the LDAP operation is computed on the LDAPMessage prior to the inclusion of the SignedOperation control. The procedure is as follows:

- Build LDAPMessage without the SignedOperation control
- Compute signature on the above LDAPMessage
- Create new LDAPMessage that includes the old MessageID, protocolOp and a control fields from the previous LDAPMessage, plus the computed signatur formatted as an S/MIME message.

No control is defined for the server to return in the LDAPResult as defined in [4]. The LDAP server MAY attempt to parse and verify the signature included in the SignedOperation control, but is not required to. The server can accept the signed operation without verifying the signature. Signature verification can be quite a lengthy operation, requiring complex certificate chain traversals. This allows a more timely creation of the audit trail by the server. Any LDAP client browsing the directory that retrieves the 'Changes' (defined in the following paragraphs) attributes, should verify the signature of each value according to the local signature verification policies. Even if the

LDAP server verifies the signature contained in the singed operation, the LDAP client has no way of knowing what policies were followed by the server in order to verify the signature.

If the LDAP server is unable to verify the signature and wishes to return an error then the error code unwillingToPerform(53) should be returned, and the entire LDAP operation fails. In this situation, an appropriate message (e.g. "Unable to verify signature") MAY be included in the errorMessage of the LDAPResult. The SignedOperation Control MAY be marked CRITICAL, and if it is CRITICAL then if the LDAP Server performs the LDAP operation, then must include the signature in the signedAuditTrail information.

The schema definition for the signedAuditTrail information is:

```
( 1.2.840.113549.6.1.0
NAME 'signedAuditTrail'
SUP top
AUXILIARY
MUST (
Changes
)
   )
```

The format of the Changes attribute is:

```
( 1.2.840.113549.6.2.0
NAME 'Changes'
DESC 'a set of changes applied to an entry'
SYNTAX 'Binary' )
```

The actual format of the Changes attribute is:

```
Changes ::= SEQUENCE {
     sequenceNumber [0] INTEGER (0 .. maxInt),
     signedOperation [1] OCTET STRING }
```

The SignedOperation attribute is a multipart/signed S/MIME message. Part 1 of the message is the directory operation, and part 2 is

the signature. Sequence number 0 (if present) always indicates the starting point directory object as represented by the definitions in "A MIME Content-Type for Directory Information", as defined in [5]. Subsequent sequence numbers indicate the sequence of changes that have been made to this directory object. Note that the sequence of the changes can be verified due to the fact that the signed directory object will have a timestamp as part of the signature object, and that the sequence numbering as part of the change attribute should be considered to be an unverified aid to the LDAP client. Sequence numbers are meaningful only within the context of a single directory entry, and LDAP servers are not expected to maintain these sequence numbers across all entries in the directory.

Some LDAP servers will only allow operations that include the SignedOperation control. This is indicated by the inclusion of a 'signedDirectoryOperationSupport' attribute in the rootDSE. This attribute is defined as:


```
( 1.2.840.113549.6.2.2
NAME 'signedDirectoryOperationSupport'
DESC 'how many of the LDAP operations must be signed'
SYNTAX 'Integer' SINGLE-VALUE )
```

The 'signedDirectoryOperationSupport' attribute above may have one of the values, '0', '1' or '2' with the following meanings:

- '0' Directory Operations may be signed
- '1' Directory Operations must always be signed
- '2' Directory Operations must never be signed

Some LDAP servers will desire that the audit trail be continuous, and not contain any gaps that would result from unsigned operations. Such server will include a signature on each LDAP operation that changes a directory entry, even when the LDAP client does not include a signedOperation control.

1.2. Handling the Delete Operation

The LDAP Delete operation represents an interesting case for Signed Directory Operations. This is due to the case that subsequent to the successful completion of the Delete Operation, the object that would have held the latest 'Changes' attribute no longer exists. In order to handle this situation, a new object class is defined to represent a directory object that has been deleted.

```
( 1.2.840.113549.6.1.2
NAME 'zombieObject'
SUP top
STRUCTURAL
MUST (
Cn $ Changes $ OriginalObject
)
   )
```

The format of the OriginalObject attribute is:

```
( 1.2.840.113549.6.2.1
NAME OriginalObject
DESC 'The LDAP URL of an object that has been deleted from the directory'
SYNTAX 'Binary' )
```


The OriginalObject attribute contains the URL of the object that was deleted from the directory. It is formatted in accordance with RFC 2255. Directory servers that comply with this specification SHOULD create a zombieObject when performing the delete Operation that contains a SignedOperation LDAPControl. The Cn attribute of the zombieObject is synthesized by the LDAP server, and may or may not be related to the original name of the directory entry that was deleted. All changes attributes that were attached to the original entry are copied over to the zombieObject. In addition the LDAP Server MUST attach the signature of the Delete operation as the last successful change that was made to the entry.

2. Signed Results Mechanism

A control is also defined that allows the LDAP v3 client to request

that the server sign the results that it returns. It is intended that this control is primarily used in concert with the LDAPSearch operation. This control MAY be marked as CRITICAL. If it is marked as CRITICAL and the LDAP Server supports this operation, then all search results MUST be returned with a signature as attached in the SignedResult control if it is willing to sign results for this user. If the server supports this control but does not wish to sign the results for this user then the error code unwillingToPerform(53) should be returned, and the LDAP search will have failed. In this situation, an appropriate message (e.g. "Unwilling to sign results for you!") MUST be included in the errorMessage of the LDAPResult. If the LDAPSigType has the value FALSE then the client is requesting that the server not sign this operation. This may be done in situations where servers are configured to always sign their operations.

The LDAP control to include in the LDAP request is (OID = 1.2.840.113549.6.0.1):

```
DemandSignedResult ::=  LDAPSigType

LDAPSigType ::= BOOLEAN
```

In response to a DemandSignedResult control, the LDAP v3 server will return a SignedResult control in addition to the normal result as defined by the operation (assuming that the server understands the control, and is willing to perform it). The SignedResult control MUST NOT be marked CRITICAL. Some LDAP v3 servers may be configured to sign all of their operations. In this situation the server always returns a SignedResult control, unless instructed otherwise by the DemandSignedResult Control. Since the SignedResult control is not marked critical, the LDAP client is allowed to ignore it. The signature field below includes the signature of the enitre LDAPResult formatted as an S/MIME

pkcs-7/signature object, as defined in [2]. The procedure for creating the signature of the signedResult control is the same as the procedure for the creation of the signedOperation control. The LDAP control in the LDAP response is (OID = 1.2.840.113549.6.0.2):

```
SignedResult ::= CHOICE {
     signature     OCTET STRING }
```

3. Security Considerations and Other Notes

The base OIDs are:

```
rsadsiLdap ::= {1 2 840 113549 6}
rsadsiLdapControls ::=  {1 2 840 113549 6 0}
rsadsiLdapObjectClasses ::= {1 2 840 113549 6 1}
rsadsiLdapAttributes ::= {1 2 840 113549 6 2}
```

The complete ASN.1 module for this specification is:

```
SIGNEDOPERATIONS DEFINITIONS ::=
BEGIN

SignedOperation ::= CHOICE {
     signbyServer   NULL,
     signatureIncluded   OCTET STRING
 }

Changes ::= SEQUENCE {
     sequenceNumber [0] INTEGER (0 .. maxInt),
     signedOperation [1] OCTET STRING }

DemandSignedResult ::=  LDAPSigType

LDAPSigType ::= BOOLEAN

SignedResult ::= CHOICE {
     signature     OCTET STRING }


END
```

If any of the controls in this specification are supported by an LDAP v3 server then that server MUST make available its certificate (if any) in the userCertificate attribute of its rootDSE object. The

UserCertificate attribute is defined in [6], and contains the public key

of the server that is used in the creation of the various signatures defined in this specification.

It is not the intention of this specification to provide a mechanism that guarantees the origin and integrity of LDAP v3 operations. Such a service is best provided by the use of an underlying protocol such as TLS [8]. TLS defines additional features such as encryption and compression. This specification does not define support for encrypted operations.

This draft proposes protocol elements for transmission and storage of the digital signatures of LDAP operations. Though the LDAP server may have verified the operation signatures prior to their storage and subsequent retrieval, it is prudent for LDAP clients to verify the signatures contained in the chained attribute upon their retrieval. The issuing Certification Authorities of the signer's certificate should also be consulted in order to determine if the signer's private key has been compromised or the certificate has been otherwise revoked. Security considerations are discussed throughout this draft.

4. References

[1] RFC 2315 PKCS 7: Cryptographic Message Syntax Version 1-5. B. Kaliski. March 1998.

[2] RFC 2311 S/MIME Version 2 Message Specification. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka. March 1998.

[3] RFC 1847 Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. J. Galvin, S. Murphy, S. Crocker & N. Freed. October 1995.

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

[5] Internet Draft, work in progress, "A MIME Content-Type for Directory Information", Tim Howes, Mark Smith, Frank Dawson Jr., 04/22/1998.

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

[7] RFC 2255 The LDAP URL Format. T. Howes, M. Smith. December 1997.

[8] Internet Draft, work in progress, "The TLS Protocol Version 1.0", Tim Dierks, Chris Allen., 11/12/1997.

5. Author's Addresses

Bruce Greenblatt
San Jose, CA 95119
USA
Email: bgreenblatt@directory-applications.com
Phone: +1-408-224-5349

Pat Richard
Xcert Software, Inc.
Suite 1001 - 701 W. Georgia
Vancouver, BC
CANADA V6G 1C9
Email: patr@xcert.com

TTaabbllee ooff CCoonnttteennttss