Deficiencies in LDAP when used to support Public Key Infrastructures

Author: David Chadwick, University of Salford, Salford M5 4WT, England.
Email: d.w.chadwick@salford.ac.uk

Abstract

This paper discusses the limitations and deficiencies in the LDAPv2 and LDAPv3 protocols when they are used to retrieve certificates and CRLs in support of a public key infrastructure. Deficiencies in both centralised and distributed directories are described. The paper then introduces current work-around solutions that implementers are employing, and the modifications and enhancements that are being and have been defined within the IETF in order to rectify these shortcomings. Finally the paper describes an enhancement that is being added to Release 2.1 of OpenLDAP to allow authentication in a distributed directory. The enhancement will allow the local server to act as a proxy when forwarding a user's request to a remote LDAP server in cases where the local server is not able to execute the LDAP query.

Keywords

Lightweight Directory Access Protocol, Public Key Infrastructure, X.500, Certificates, Certificate Revocation Lists, Internet Standards, Internet Engineering Task Force

Introduction

The lightweight directory access protocol (LDAP) is the de-facto and de-jure standard way of accessing directory services that conform to the X.500 data model [ISO1]. It is very widely supported and deployed by all the leading software vendors, and is part of the latest version of Windows 2000 Active Directory. LDAP comes in two versions:

> LDAPv2 [Yeong] - the truly lightweight variation of the Directory Access Protocol (DAP) [ISO1], and
> LDAPv3 [Wahl] - the heavyweight version (occasionally known as the Leadweight Directory Access Protocol).

Whilst the DAP was designed from its inception to support public key infrastructures (PKIs), being part of the same X.500 family of standards as X.509 [ISO1], LDAP was not. LDAP has however become the predominant protocol in support of PKIs accessing directory services, but because of its lineage, it has some deficiencies. This paper describes the deficiencies in both the LDAPv2 and v3 protocols, along with the solutions that have been and are currently being standardised within the IETF in order to rectify them. The deficiencies will be documented firstly for a centralised directory service, in which a single standalone LDAP server is being used to support a single PKI, and secondly for a distributed directory service, in which there are many LDAP servers that need to co-operate in order to support a network of interconnected PKIs.

Centralised LDAP Deficiencies

Certificate and CRL retrieval

The first attempt to support certificate retrieval using LDAPv2, was documented in [Howes1]. Unfortunately it did not work. The reason for this was that the LDAP encoding of a certificate, from its ASN.1 [ISO2] type, length, value (TLV) binary encodings, into a simple LDAP ASCII string, is an irreversible process for distinguished names. There is a one to many decoding as relative distinguished names can be of type teletex, printable or universal strings. Consequently when the LDAP client tries to reconstitute the ASN.1 binary in order to validate the signature on the certificate, it proves impossible to do without producing possibly hundreds of different ASN.1 encodings, and checking the signature against each. This is not practical. The obvious solution is to transfer the certificate or CRL in its binary format so that the ASN.1 encoding is preserved, and not to convert it into ASCII strings. This solution is documented in the LDAPv2 profile for PKIs [Boeyen]. The solution by and large works, and is used by PKI LDAP clients successfully. However, some LDAP clients, such as Netscape Communicator, still do not interpret the certificate properly when it is retrieved. They attempt to display it as a single stream of binary characters in the browser window, rather than decoding the ASN.1 it into its constituent parts.

When LDAPv3 was specified, it overcame the encoding bug of LDAPv2, by introducing the concept of attribute descriptions. Attribute descriptions are used to optionally qualify attribute type definitions, and one special option, the *;binary* option, is built into the base LDAPv3 specification [Wahl]. The ;binary option, when used by a client to describe an attribute that it wants to retrieve, mandates that the LDAPv3 server must return the attribute values encoded using the ASN.1 Basic Encoding Rules (BER) rather than in their LDAP defined string encodings. In this way, when used to retrieve signed attributes such as certificates and CRLs, the client can validate the signatures against the binary values.

4.2 LDAPv3 complexity

Unfortunately LDAPv3 is a complex protocol. It has many sophisticated features built into it, so that it can incorporate all the richness needed of a general purpose directory access protocol (just like the original X.500 DAP). Many of these features are not needed for simple PKI certificate and CRL retrieval, and so an LDAPv3 profile [Chadwick1] has been specified by the PKIX working group. The profile specifies the features of LDAPv3 that it is mandatory to support for a PKI, those features that optionally may be supported for a PKI, and those features of LDAPv3 that are not needed if support of a PKI is the only concern of the client. This Internet Draft is about to go to Last-Call in the PKIX working group. (Last-Call is the penultimate stage to becoming a proposed standard in the IETF standardisation process.)

Searching for the correct certificate or CRL

Simple PKIs usually only store single certificates in each user's directory entry, and only one CRL in each distribution point. However, how does the LDAP client know which entry this is, in order to be able to ask for the certificate (or CRL) to be returned? In short, the client needs to be able to specify filtering criteria that allow the LDAP server to search through its database and select only those entry or entries and

attribute(s) that match the client's criteria. For example, "Find the entry for the person named Carly Simon and return her userCertificate attribute", or "Find the userCertificate attribute containing the email address inet.person@someorg.com". Unfortunately no certificate or CRL matching rules have been defined for LDAP (although they do exist for X.500 DAP). Thus the latter search cannot be requested without some work-around.

So how are PKI implementers coping with this problem today? Basically, the directory server administrator has to create one or more new attributes in the user's entry, in parallel with the certificate attribute. These new attributes each contain the contents of one of the fields of the user's certificate that are to be searched for e.g. the mail attribute holds the contents of the rfc822 value from the subjectAltName certificate extension, or the serialNumber attribute holds the certificate serial number. The client can then search for the entry containing this new attribute (using LDAPv2 or LDAPv3), and ask for the accompanying userCertificate attribute to be returned. Hopefully the certificate will be the one that matches the user's search criteria. But this places a large burden on the directory administrator, as he has to ensure that the new attributes are kept synchronised with the contents of the user's certificate. Furthermore, this solution has security implications, since whilst the certificate is digitally signed to prevent unauthorised tampering, the new attributes are not signed, and can thus be tampered with without it necessarily being immediately obvious to the client or the directory administrator. (The client would have to check that the contents of the retrieved certificate truly did match his filtering criteria. The directory administrator would continually have to check that the new attributes matched the contents of their peer certificate.) Finally, this solution does not work well when the user has two or more certificates, since there is no way of pairing the new attributes with particular certificate values held in the userCertificate attribute. (This is because attribute values are held as an unordered set and not in any defined order.) Clearly a better solution is required.

Consequently the IETF PKIX group has published an Internet Draft [Chadwick2] that specifies matching rules for both certificates and CRLs. This will allow the client to specify his filtering criteria, and the server will match directly on the contents of the certificate or CRL, rather than on the contents of some peer attributes. The client can then be assured that the returned certificate or CRL matches the one(s) requested. As this standardisation work only started during 2000, it is unlikely that a proposed standard will be issued much before the end of 2001. Besides, before the proposed standard is published, ideally the community should have some experience of using it. This requires that at least one supplier should implement the proposed matching rules in a server, and should also produce a client interface for specifying the filtering criteria. In this way users will be able to check that the matching rules are really the most useful ones in practice.

Users with multiple certificates

As PKIs become more advanced, users will start to be issued with several certificates. For example, a user might have a certificate for creating digitally signed messages, a certificate to be used by others for sending encrypted S/MIME email to the user, a certificate to be used for e-commerce transactions, and a certificate that gives the user special privileges on the corporate LAN. If all these certificates are stored in the user's entry - which is the natural place to store them - then LDAP clients have an

additional problem (this is assuming that the matching rules mentioned above have been implemented so that the client can search for a particular certificate e.g. find the S/MIME encryption certificate for A.Person@e-commerce.site.com). The problem is that there is no standard way to ask the LDAP server to only return one certificate from the user's entry. Both LDAPv2 and LDAPv3 protocols are only capable of returning all the attribute values from a particular attribute, or none of them. It is currently not possible for a user to specify a subset of the attribute values. This causes the client software a problem if several certificates are returned when only one was being sought. How is the correct one to be chosen? The client software will have to implement complex matching rules to parse each certificate and find the one that fulfils the user's criteria. This is both time consuming and unnecessary. If the LDAP server has already implemented the certificate matching rules mentioned above in order to find the correct certificate in the first place, it should surely be able to return just the desired certificate(s) to the client, rather than all or none of them. This is the subject of the matched values Internet Draft [Chadwick3], which is now ready to enter last call in the LDAPExt working group. The draft specifies an extension to the LDAPv3 protocol, that allows the user to say which value or values should be returned. Unfortunately it is not possible to specify this extension in LDAPv2, as LDAPv2 is a fixed protocol with no possibility of extending it.

So how do PKI implementers cope with this problem today. Basically they have to ensure that each certificate attribute only holds a single certificate attribute value. This can be done in one of two ways, either create new certificate attributes for each type of certificate, or alter the structure of the directory information tree (DIT) to fit the PKI applications. The former method is problematical, as some PKIs are not capable of changing the attribute type used for publishing certificates, but must always use the X.509 standard attribute type (userCertificate). The latter method is sure to work with all PKI implementations, and is usually executed in one of three ways as shown in Figures 1-3 below. (Note that whether the organisation uses DC or X.521 naming is immaterial to the example.)

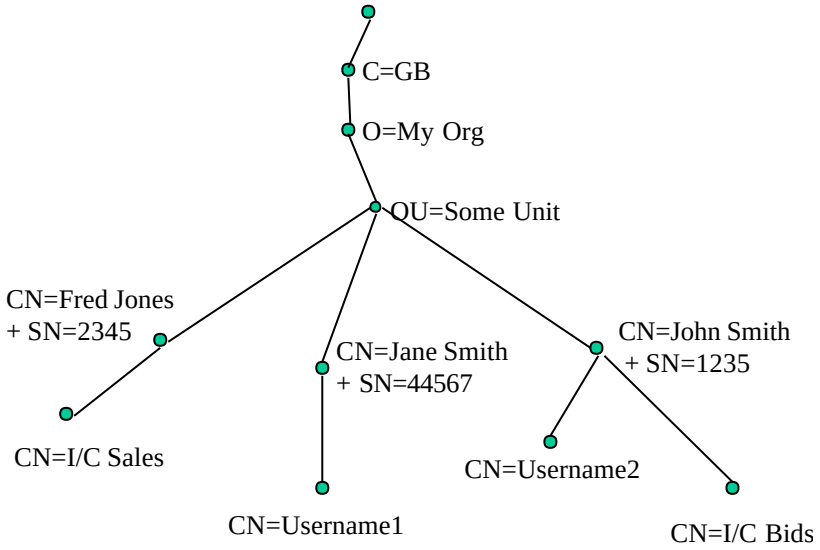

Figure 1. Child entries

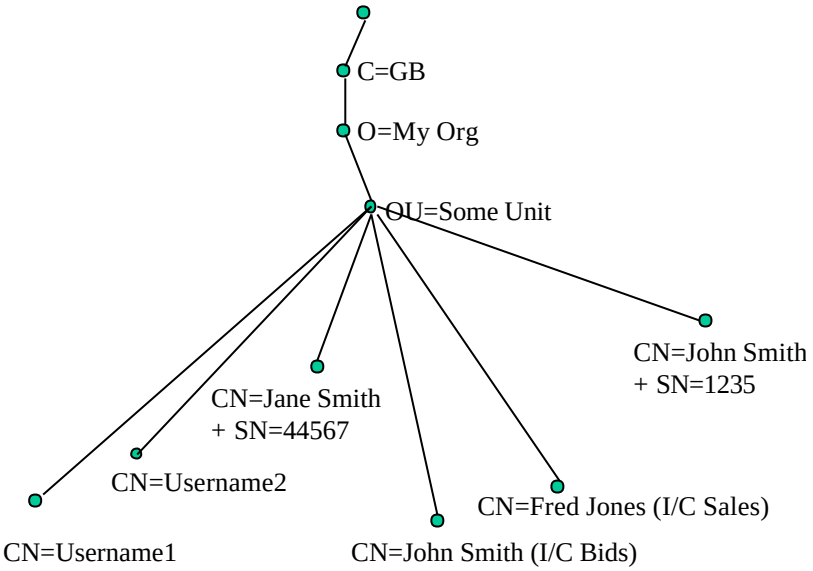

Figure 2. Sibling Entries

The first and second ways keep the existing DIT structure intact, but add extra entries for each user, as new certificates are issued. This ensures that only a single certificate is ever stored in each entry. The new entries are created either subordinate to the user's existing entry (Figure 1) or as siblings of the existing entries (Figure 2). The entries are given names that indicate the type of certificate stored within them. The third way creates a new DIT subtree for each PKI application, and users have separate entries in each parallel application tree (Figure 3).

A disadvantage of the first way is that some LDAP browsers expect common name (CN) entries to be leaf entries, and cannot cope with them being non-leaves. A disadvantage of the second way is that a user's name is prefixed with the type of certificate contained within the entry (although this is the solution we have adopted in our PKI infrastructure). A disadvantage of the third way is that new tree structures have to built for each application (although this can be an advantage is the application is to be the sole user of the new subtree).

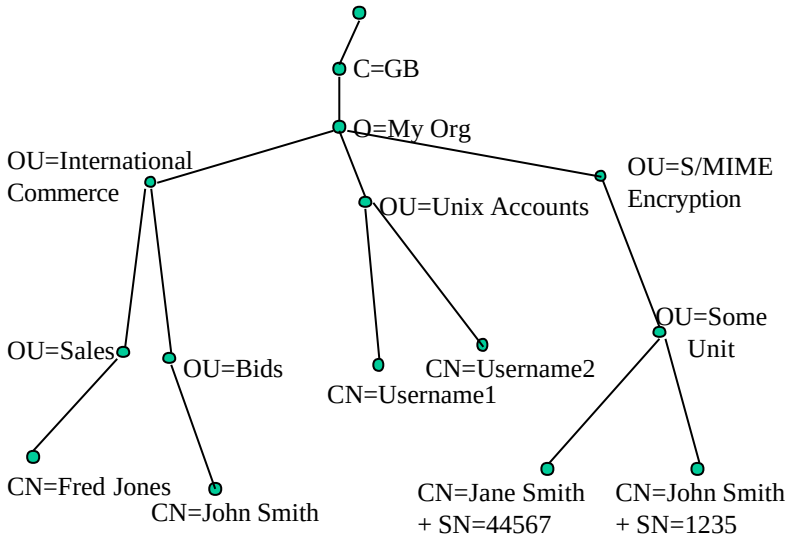

Figure 3. Application based subtrees

Distributed LDAP Deficiencies

Once PKIs start to link together, via either CA hierarchies, cross certification or bridge CAs, then PKI users will need to have access to the certificates and CRLs contained in each of the PKI directory services. Thus ideally we need a distributed directory service made up of all the individual LDAP directories in order to support inter-linked PKIs. Experience to date, for example with the US Federal PKI [Burr], suggests that an X.500 based distributed directory with chaining, using LDAP client access, provides the only workable solution. LDAP only servers are currently not up to the task. Why is this?

LDAP was initially conceived as a lightweight front-end protocol to a distributed X.500 directory service. Consequently LDAPv2 had no need to support distribution, as the X.500 servers used the Directory System Protocol (DSP) [ISO1]. The types of feature that a distributed directory service needs are:

- the ability of the servers to know about other servers (in X.500 these are called knowledge references, in the DNS these are the NS and A resource records) and how to interact with them; otherwise the client has to know the whereabouts of all the LDAP servers that it wishes to contact, and has to choose the correct one for each request that it issues,
- the ability of the servers to pass client requests between themselves (in X.500 these mechanisms are called chaining and referrals, in the DNS these features are called the recursive and iterative modes respectively), otherwise the client has to contact each server separately and track its own progress,
- the ability of the servers to perform distributed authentication, otherwise the client has to have separate authentication tokens at each LDAP server that it wishes to contact.

Chaining and Referrals

LDAPv3 has made a start on providing a distributed directory service by adding referrals to the LDAPv3 protocol. (It is not possible to add referrals to the fixed LDAPv2 protocol.) Some LDAP server suppliers, mainly those that have built their LDAP service around their X.500 servers, also provide LDAP chaining. However, LDAP has not yet standardised knowledge references or distributed authentication, so building a distributed LDAP service is still extremely difficult. Referrals on their own are insufficient, since servers need a standard way of using them.

Knowledge references

There have been several attempts to define standard LDAP knowledge references [e.g. Howes2]. Quite why these have failed to progress past Internet Draft stage is difficult to assess. Perhaps it is because the LDAP server suppliers saw this feature as low priority, given that the vast majority of their market was providing centralised LDAP servers into a corporate environment, where a distributed directory service was not needed. Perhaps it is because each supplier has his own way of storing these attributes, based on either X.500 knowledge references or some proprietary format. Whatever the reason, until recently the work seemed to have stalled. However, during 2000 there has been a resurgence of interest in this topic, and a new Internet Draft [Hedberg] has been published. Perhaps the IETF LDAPExt working group will reach

consensus on a standard format for knowledge references and how they are to be used in distributed name resolution during 2001.

Distributed authentication

Concerning distributed authentication, two mechanisms have been defined by X.500. Both rely on the directory servers having a trust relationship between themselves, and on them having previously exchanged and stored the authentication tokens of each other. In the first mechanism, the user binds to his local server and is authenticated by it. The user then sends its request to the local server. If the local server is unable to fulfil the request, it binds to the remote server and chains the user's request to the latter. It tells the remote server the name of the user and the level of authentication that it has performed. The remote server then allows the user to have access to the appropriate amount of directory data, as directed by its access controls. In the second mechanism, the user binds to the remote server using the same credentials that he would normally use for his local LDAP server. The remote server binds to the local server, and after establishing a trusted link, issues a Compare operation to the local server, passing the users credentials. The local server is then in a position to check these credentials and pass a True/False reply back to the remote server. Note that this latter method can only work securely if the link is secure, or the credentials are encrypted prior to transfer.

LDAP does not have a standardised way of performing distributed authentication, however, the following mechanism is currently being implemented for release 2.1 of the OpenLDAP implementation (http://www.openldap.org). It uses the proxying feature of the Simple Authentication and Security Layer (SASL) [Myers]. This allows a client to specify an authorisation identity different from the authentication identity provided on the Bind operation, and therefore the client can act as a proxy for an authorised user (see Figure 4).

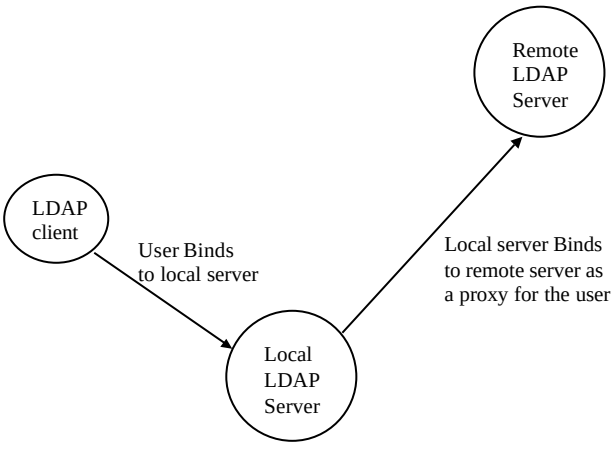

Figure 4. LDAP server proxying

In order for this to work, the OpenLDAP solution requires a remote server to hold the following:

- authentication information for the local server, so that it can authenticate the latter at bind time;
- authorisation identity information for the local server, so that it can verify that the latter is authorised to assert the user's identity.

This requires some new LDAP authorisation schema to be defined. When a user binds to his local server he authenticates to it and is associated with some authorisation identity (this will usually be the same as or similar to his authentication identity but may be different from it). If the local server is unable to fulfil the user's request, it needs to pass the request to a remote server. The local server binds to the remote server using its own authentication information, but in addition, it asserts the authorisation identity of the user. The remote server then checks its authorisation schema to see if this authorisation assertion is allowed, and if it is, executes the operation providing the local server with the information that the user is authorised to access. The local server is therefore acting as a proxy for the user when it chains the LDAP operation to a remote server.

Note that one draw back to this approach is that the local server must have separate sessions to the remote server for each user that it is concurrently acting as a proxy for. This disadvantage is not suffered by the X.500 distributed authentication schemes, as the local server does not act as a proxy for the user, but rather chains the user's request via DSP. The DSP carries the user's DN in its chaining arguments. To resolve this issue, LDAP will need to mirror this feature of DSP, by adding a control to each LDAPv3 chained operation to present the client's authorisation identity. This will then allow LDAP chained operations from multiple clients to share a common connection between the local and remote servers. Note that standard LDAP extensions currently have not been defined, even in draft form, for either of these features, so that using pure LDAP servers to build a distributed directory service is still a long way off.

Conclusions

This paper has highlighted the many deficiencies in the LDAP protocols that arise when they are used to access PKI related information. These are mainly a result of the fact that LDAP was not initially conceived as a protocol to support PKIs. However, the IETF has been steadily addressing these deficiencies over the past few years, with the consequence that by the end of 2001 it is anticipated that standard solutions should be available to address most of them. Unfortunately, some of the extensions needed for LDAP to support a distributed directory service still have not been specified.

Acknowledgements

The author would like to thank the EC IV Framework ICE-TEL and ICE-CAR projects, the TERENA managed SCIMITAR2 project, and the UK EPSRC 3D and ICT projects for funding his attendance at various IETF meetings during the past three years. The author would also like to thank Kurt Zeilenga of OpenLDAP for describing the distributed authentication method that is currently being implemented.

References

(Boeyen) Boeyen, S., Howes, T., Richard, P. "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2." RFC 2559, April 1999
(Burr) Burr, W.E. "Proposed Federal PKI Architecture," 19 May 1998
(Chadwick1) Chadwick, D.W. "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv3". <draft-pkix-ldap-v3-03.txt>, September 2000.
(Chadwick2) Chadwick, D.W., Legg, S. "Internet X.509 Public Key Infrastructure, Additional LDAP Schema for PKIs and PMIs", <draft-pkix-ldap-schema-01.txt>, September 2000.

(Chadwick3) Chadwick, D.W., Mullan, S. “Returning Matched Values with LDAPv3”, Internet Draft <draft-ldapext-matchedval-03.txt>, September 2000
(Hedberg) Hedberg, R. “Referrals in LDAP Directories”, <draft-ietf-ldapext-refer-00.txt>, July 2000
(Howes1) Howes, T., Kille, S., Yeong, W., Robbins, C. “The String Representation of Standard Attribute Syntaxes”. RFC 1778, March 1995
(Howes2) Howes, T., Wahl, M., Chadwick, D.W. “Referrals and Knowledge References in LDAP Directories”, <draft-ietf-ldapext-referral-01.txt>, March 1998
(ISO1) ISO/ITU-T X.500 (1993) Series of Recommendations on The Directory, Parts 1-8
(ISO2) ISO/ITU-T Rec. X.690, “Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules”, 1994.
(Myers) Myers, J. “Simple Authentication and Security Layer (SASL)”. RFC 2222, October 1997
(Wahl) Wahl, M., Howes, T., Kille, S. “Lightweight Directory Access Protocol (v3)”, Dec. 1997, RFC 2251
(Yeong) Yeong, W., Howes, T., and Kille, S. “Lightweight Directory Access Protocol”, RFC 1777, March 1995.