

# Directory Services for Linux

in comparison with Novell NDS  
and Microsoft Active Directory

by

Norbert Klasen

Matr.-Nr.: 202620

A thesis for submission to the

DEPARTMENT OF COMPUTER SCIENCE

in partial fulfillment of the requirements  
for the degree of “Diplom-Informatiker” at the

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE  
AACHEN, GERMANY

August 2001

Erstgutachter

[Prof. Dr. Dietmar Kaletta](#)

ZENTRUM FÜR DATENVERARBEITUNG

UNIVERSITÄT TÜBINGEN

Betreuer

[Peter Gietz](#)

[Dr. Kurt Spanier](#)

Zweitgutachter

[Prof. Dr. Ir. Boudewijn R. Haverkort](#)

LEHR- UND FORSCHUNGSGEBIET

LEISTUNGSBEWERTUNG UND VERTEILTE SYSTEME

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Tübingen, den 9. August 2001

(Norbert Klasen)



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Why Directories? . . . . .	1
1.2	Layout of this Thesis . . . . .	2
<b>2</b>	<b>Directory Services Overview</b>	<b>3</b>
2.1	Information Model . . . . .	3
2.2	Naming Model . . . . .	5
2.3	Functional Model . . . . .	5
2.3.1	Interrogation . . . . .	6
2.3.2	Modification . . . . .	7
2.3.3	Authentication and Control . . . . .	7
2.4	Distribution Model . . . . .	7
<b>3</b>	<b>A History of Directory Standards</b>	<b>9</b>
3.1	Early Electronic Directories . . . . .	9
3.2	Special Purpose Directories . . . . .	9
3.3	General Purpose Directories – X.500 . . . . .	10
3.4	Lightweight X.500 – LDAP . . . . .	11
3.5	LDAPv3 . . . . .	14
3.6	Text-based Standards . . . . .	16
<b>4</b>	<b>Access Control and Security Layers</b>	<b>17</b>
4.1	Authentication . . . . .	17
4.1.1	Cryptographic background . . . . .	18
4.1.2	Authentication Methods . . . . .	19
4.2	Authorization . . . . .	22
4.3	Security Layers . . . . .	23

4.4	SASL	23
4.5	Authentication Methods for LDAP	24
<b>5</b>	<b>Developing LDAP-enabled Software</b>	<b>25</b>
5.1	C	25
5.2	Perl	27
5.3	Java	27
5.4	PHP	28
5.5	Python	28
5.6	ADSI	28
<b>6</b>	<b>Directory Software for Linux</b>	<b>30</b>
6.1	Directory Servers	30
6.1.1	OpenLDAP	30
6.1.2	Netscape	32
6.1.3	IBM	33
6.1.4	MessagingDirect	33
6.1.5	Novell	33
6.2	Administration Tools	34
6.2.1	Command line tools	34
6.2.2	lbe	34
6.2.3	gq	34
6.3	Directory-enabled Applications	35
6.3.1	Name Service Switch	35
6.3.2	Pluggable Authentication Modules	36
6.3.3	Sendmail	36
6.3.4	Apache	37
<b>7</b>	<b>Novell Directory Services</b>	<b>38</b>
7.1	Directory Server	38
7.2	Administration Tools	40
7.3	Directory-enabled Applications	40
<b>8</b>	<b>Microsoft Active Directory</b>	<b>42</b>
8.1	Directory Server	42
8.2	Administration Tools	44
8.3	Directory-enabled Applications	45

<b>9</b>	<b>LDAP-enabled user management for Linux</b>	<b>47</b>
9.1	Requirements	47
9.2	Implementation	48
9.2.1	Schema	48
9.2.2	Directory Service	49
9.2.3	Authentication Considerations	50
9.2.4	Single Sign-On	50
<b>10</b>	<b>Conformance and Interoperability</b>	<b>53</b>
10.1	Interoperability Tests	53
10.2	Supported features in DSAs	54
10.3	Support for standard schema items	54
10.4	SASL GSSAPI	57
<b>11</b>	<b>Performance</b>	<b>60</b>
11.1	Test Environment	60
11.2	Tests	61
11.2.1	Loading - Initial directory population	62
11.2.2	Messaging	63
11.2.3	Address Look-up	63
11.2.4	Authentication	64
11.3	Results	65
11.4	Comparison with real-life requirements	65
<b>12</b>	<b>BibTeX records in LDAP</b>	<b>67</b>
12.1	Schema and DIT	67
12.2	Converting bibliographies to LDIF	69
12.3	Retrieving BibTeX records from LDAP	69
12.4	A web front-end with Java Servlets	69
12.5	Conclusions and future work	70
<b>13</b>	<b>Conclusions and Future Work</b>	<b>71</b>
<b>A</b>	<b>Password Synchronization</b>	<b>73</b>
A.1	Changes originating in Linux	73
A.2	Changes originating in Windows 2000	74

<b>B LanManager Hashes in OpenLDAP</b>	<b>75</b>
<b>C Abbreviations</b>	<b>77</b>
<b>Bibliography</b>	<b>86</b>

# Chapter 1

## Introduction

### 1.1 Why Directories?

The X.500 standard gives the following characterisation of a directory [38]: “The Directory is a collection of open systems which cooperate to hold a logical database of information about a set of objects in the real world.” Directories have some properties that set them apart from relational databases [68]:

- Directories are organised in an object-oriented and hierarchical way. Information about a real-world object is stored in the entry that represents this object in the directory. To mirror the relationships of their respective objects, entries can be organised in a tree structure.
- Directory services provide a common schema for what can/must be stored for a certain class of objects and a standard access protocol, which greatly facilitates interoperability.
- Directories services offer a fine-grained security model. For example, access restrictions can be specified for one entry and then inherited by all entries below this entry in the directory tree.
- Directory services do not support transactions. Instead, they adopt a loose-consistency model. This allows for improved local availability of the service in a distributed environment.

The most common areas of application for directories are *white pages* and *yellow pages* services. In white-pages services such as a phone book, information about an object is accessed by the object’s name, whereas yellow pages allow for information to be searched or browsed by specifying categories. Due to their flexibility, directories are being used

in other applications as well, for example, as information repository for users and resources in computer networks.

The successful standardisation efforts on directories have created a highly interoperable software landscape. Support for directory protocols has become a standard feature in many programs. Standard-compliant general-purpose directories are therefore often used to consolidate information germane to multiple applications into a single repository. In an economic viability-analysis the impact of the introduction of a directory based white-pages and Single Sign-On service in six banks and insurance companies has been researched. This case study estimated that “the benefit of the directory would be 11 times that of its cost” which would lead to savings of about 23 Mil. DM [100].

## 1.2 Layout of this Thesis

The main aim of this thesis is to show that directories can be a viable tool for managing user accounts and resources in computer networks of academic institutions. Furthermore, criteria for choosing between different directory products will be given and the currently available software packages for Linux will be evaluated to these criteria.

Chapter 2 will give a description of the fundamental aspects of a general-purpose directory service. In Chapter 3 a historic review of different standards for directory services will be given. It will concentrate on X.500 and will point out the reasons why a lightweight version of this standard was conceived.

Security considerations for accessing directory services have been analysed in the frame of this thesis and will be presented in Chapter 4.

Chapter 5 will give an overview how support for directory services can be integrated into applications.

In Chapters 6, 7 and 8 a description of current directory server products available for Linux as well as the products from Novell and Microsoft will be given.

Chapter 9 will describe which requirements would be posed to a directory-enabled user management system and how such a system has been implemented in the frame of this thesis.

In Chapters 10 and 11 an analysis of directory servers with regard to their suitability as a back-end for a user management and a white pages service will be presented.

This thesis was typeset with LaTeX. A schema and several utility programs have been developed to store and manage BibTeX bibliographical references in a directory. This work will be presented in Chapter 12.

Chapter 13 will give a summary of the work done and a perspective on possible future work items.

In Appendix B and A ways to synchronise passwords between Linux and Windows 2000 will be described.

# Chapter 2

## Directory Services Overview

This chapter describes the fundamental concepts of X.500 and related directories services. Each aspect of the directory service is outlined by a pertaining model [30].

### 2.1 Information Model

Directories contain information about real world objects. Information concerning one object is stored in an *entry*—the basic building block of a directory (see Figure 2.1).

An entry is a collection of name-value pairs called *attributes*. There can be more than one value associated with an attribute name—a person might have multiple first names, for example. However, some attributes may only have one value—a person definitely would only have one date of birth. Attributes of that kind are said to be *single-valued*. The others are called *multi-valued*.

Each attribute has a *syntax*, which describes the type of information that can be stored in this attribute. This might be a character string, a number, a photo or some complex data type, e.g. a certificate for digital signatures. The definition of an attribute also lists its *matching rules*. These rules govern how values of this attribute type should be compared. It is possible to specify rules for *equality* and *substring* matching as well as specifying rules that determine how values of this attribute should be ordered—the *ordering* matching rule.

Objects in the real world often show similarities and are therefore said to be of a particular kind. This grouping principle is met in the directory world by the notion of *object classes*. Object classes specify the attributes that an entry must or may contain. Attributes that an entry must contain are called *mandatory*, attributes that may be present *optional*. It is possible to derive an object class from another one. This allows

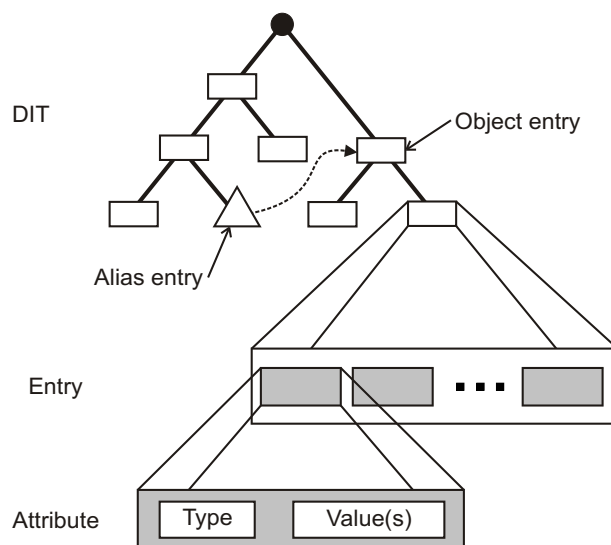


Figure 2.1: DIT and Entry Structure [38]

the design of an object class hierarchy. Subclasses inherit all mandatory and optional attributes of their superior class and can incorporate additional ones. It is also possible to declare an inherited optional attribute as mandatory for the subclass.

Three types of object classes exist: *abstract*, *structural* and *auxiliary*. Object classes of the abstract type are used to form the upper levels of an object class hierarchy. Entries can only be added to the directory if they meet the requirements of at least one structural object class. Structural object classes reflect the principal fabric of an object. Common structural object classes are, e.g. “person” or “organization”. The object classes to which an entry conforms are listed in its “objectClass” attribute. The “objectClass” attribute is introduced as a mandatory attribute by the “top” object class. “top” forms the root of the object class hierarchy. All other object classes are directly or indirectly derived from it. This ensures that every entry has at least one object class. Sometimes the need arises to store additional data that is not strictly tied to the structure of an object or which may not be present for all objects of a particular class. This additional data can be stored in attributes, which are governed by auxiliary object classes. With auxiliary classes, attributes can be introduced as a mandatory requirement to a subset of entries that have the same structural class. An auxiliary class can also be added to entries that have a different structural classes, e.g. both a person and an organization could have a homepage, which would be stored in the common “labeledUri” attribute.

When more applications with partly different needs employ the directory as their data storage, the advantage of introducing auxiliary classes over deriving from structural

object classes becomes obvious: The latter approach would require a structural class for every combination of attributes (or sets of attributes) whereas auxiliary classes allow the addition of object classes to existing entries as needed.

There is one very special structural object class: *alias*. Entries of this type do not actually contain information on objects but rather act as placeholders that point to other entries. With the use of aliases it is possible to access the same data under a different name.

The collection of syntax, matching rule, attribute type and object class definitions is called the *schema*. This meta information governs what might be stored in the directory.

## 2.2 Naming Model

All entries in the directory are arranged in a hierarchical manner—forming the *Directory Information Tree* (DIT) (see Figure 2.1). In this tree, directories are similar to file systems. There is however one decisive difference: An entry in a directory can simultaneously hold information itself in addition to being a container for other entries, whereas objects in a file system can either be a directory or a file<sup>1</sup>.

Each entry in a directory is identified by its *Distinguished Name* (DN). It is composed by concatenating the entry's *Relative Distinguished Name* (RDN) with those of its superiors along the path to root entry, whose RDN is a hypothetical empty string (“”). Figure 2.2 shows how entries in a directory could be named. An RDN consists of an attribute name, the equal sign and the attribute value (e.g. “cn=Sam Carter”)<sup>2</sup>. The attribute used in the RDN is called a *naming attribute*. RDNs must be different for all siblings in a tree. This ensures that all entries will have a unique DN.

X.500 initially followed a naming-scheme based on geographic or national regions. Since acquiring a registered name in this scheme proved cumbersome, a new naming-scheme based on the Domain Name System (DNS) was introduced [50]. To map a DNS name to a DN, the *dc* attribute—short for domain component—is used: “directory.dfn.de” thus maps to “dc=directory, dc=dfn, dc=de”.

## 2.3 Functional Model

The functional model describes the means by which information is accessed in the directory. It defines the operations by which the user—by means of a program called

---

<sup>1</sup>Not counting operational attributes such as modification time.

<sup>2</sup>The standard allows for multi-valued RDNs such as “cn='Sam Carter' + uid=scarter”; however these are rarely used and sometimes not supported by a directory server, e.g. in Active Directory.

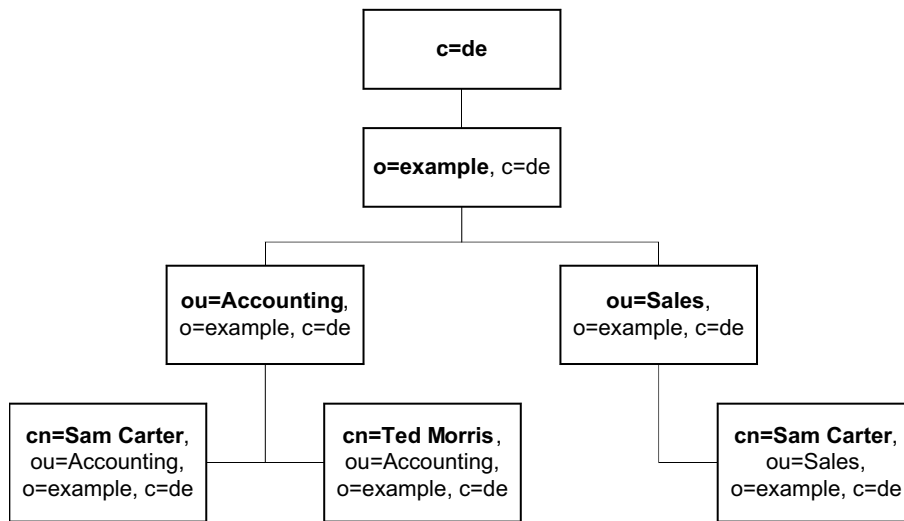


Figure 2.2: A namespace example

*Directory User Agent* (DUA)—interacts with the application providing the directory service—the *Directory System Agent* (DSA). Operations can be classified into three groups [30].

### 2.3.1 Interrogation

The primary use of directories is to provide information. To request such information the *search* operation is used. How a search is actually performed is controlled by the following parameters:

- *baseDN*—gives the node in the DIT from which the search starts.
- *scope*—either “base”, “one-level” or “subtree”. If “base” is specified, only the entry specified by *baseDN* is searched. With “one-level” only entries directly below the *baseDN* entry are considered as candidates in the search. As the name implies, “subtree” searches through all entries in the subtree whose root is formed by the *baseDN* entry.
- *filter*—what is actually searched for. This could be as simple as “givenname = Sam” or a complex nested logical expression, e.g.:  
“( & (objectclass=person) (modifyTimeStamp>=200101010000Z) ( | (gn=Sam) (gn=Ted) ) )”<sup>3</sup>

<sup>3</sup>A search with this filter will return all person entries with a first name of Sam or Ted that have been modified since January 1st, 2001.

- Other parameters determine which attributes should be returned, whether time and size limits would have to be observed, and if aliases should to be de-referenced.

The *compare* operation checks, if a specific entry contains an attribute of a given value. It is still included in the standard because of historic reasons, since its semantics can be reproduced by a tailored search operation.

### 2.3.2 Modification

Before one can retrieve information from the directory it first has to be populated with entries. Two function exists to add and delete entries in the directory, namely the *add* and *delete* functions. Furthermore, there is the function *modifyDN* to rename and/or move an entry in the DIT. But entries can also be changed on the attribute level. Using the *modify* operation, new values can be added to an attribute, particular values removed from an attribute, or all values replaced by new ones.

Although directories offer no support for transaction mechanisms, operations that modify an entry are atomic. Either all attribute changes can be committed to the entry or none at all.

### 2.3.3 Authentication and Control

Some operations in the directory require special privileges. Before these can be carried out, the identity of the client has to be ascertained. This is done using the *bind* operation. (See also Section 4.1) A client can terminate a connection by issuing the *unbind* request. Upon receiving such a request the DSA will discard all outstanding requests from this client and close the connection. *abandon* is sent by the DUA if the results of an operation initiated earlier are not required anymore—e.g. when a user has clicked on a “Cancel” button in a graphical user interface (GUI).

## 2.4 Distribution Model

Due to the hierarchical structure of directories, directory services are well suited to be implemented as distributed systems. To achieve this, the DIT is *partitioned* into smaller areas, each being a connected subtree, which do not overlap with other partitions.

Figure 2.3 shows this approach. A separate server would master each such *partition*. The links between partitions are known as *knowledge information* and can themselves

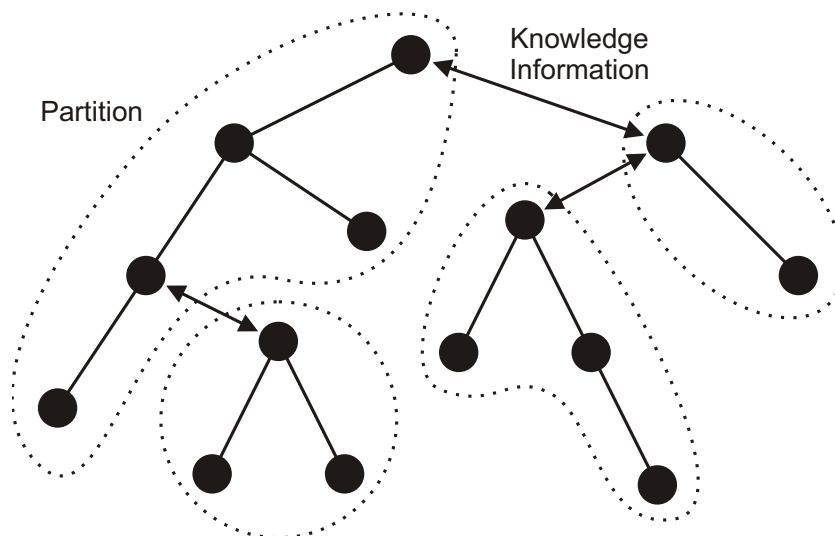


Figure 2.3: DIT Partitioning and Knowledge Information [4]

be stored in the directory. If a client issues a request, which effects entries that do not fall into the partition mastered by the server, the server has two options. The first one is to retrieve this information from the responsible server on behalf of the client. This feature is called *chaining*. The other option is to *refer* the client to the right server. It is then the client's task to follow this *referral* and contact the new server to retrieve the information.

To increase the availability of a directory service, multiple hosts can run directory servers that hold the data of a partition. Such a set up would also be feasible if a directory service is to be provided to two different sites, which are connected by a WAN link. The clients at each site can then access the local server, thus avoiding the use of the slower inter-site connection. As the data needs to be kept in sync, changes made by clients would have to be propagated to all other servers holding a copy of the effected entries. This procedure is called *replication*. The current standard only defines a replication architecture, where only one server masters the data in one partition. In such a *single-master* environment, all clients have to contact the master server if the need to update the directory would arise. Some proprietary implementations such as NDS or Active Directory have *multi-master* capabilities: Clients can connect to any server to request operations that will modify entries in the directory.

# Chapter 3

## A History of Directory Standards

### 3.1 Early Electronic Directories

In the early 1980s research was carried out into distributed computing systems at the Xerox Palo Alto Research Center and led to the development of Grapevine [5]. Later, the Xerox *Clearinghouse* was built on the results of this research. It was the first distributed directory to share user account information. The Grapevine network spanned 1500 computers in more than 50 local networks. Thanks to Clearinghouse a user could log in both at his usual desk, and at any of the Xerox overseas offices.

### 3.2 Special Purpose Directories

Along with the Internet came another distributed directory—the *Domain Name System* (DNS). DNS was designed to fulfil a specific task. The IP protocol used in all Internet transmissions employs numeric addresses. Since these are quite hard to remember for humans, a level of indirection was added: a symbolic name was associated with each IP address. At first these mappings were kept in plain text files that were centrally maintained for the whole Internet. But as the Internet grew, this soon proved unfeasible. Therefore the hierarchical DNS was introduced in which the various branches of the DNS were delegated to organizations, who are responsible for their domain.

The name-to-address mapping tables are kept by so-called *name servers*. If a client requests an IP address for a non local hostname, the local name server will then retrieve the address of the name server for the target's domain and in turn get the requested IP address from this server on behalf of its client. [16]

### 3.3 General Purpose Directories – X.500

X.500 was developed jointly by the International Telegraph and Telephone Consultative Committee (CCITT), which is now known as the International Telecommunications Union (ITU), and the International Organization for Standardization (ISO) [6]. Their aim was to define a general-purpose directory standard, which would cater for different needs. The ITU-T members, mostly national telecommunications operators, looked for a system, which would provide a white pages service for phone numbers and X.400 [37] email addresses, whereas the ISO was interested mainly in providing a name service for Open Systems Interconnect (OSI) applications. The idea was to build a globally distributed system that would offer homogenous access to the information [6]. That is why in X.500 terms directories are often referred to in the singular, i.e. there exists one and only one directory—called “THE Directory”.<sup>1</sup>

The first joint meeting was held in April 1986 and the standard documents—the so-called ’88 edition—were published in January 1990 by CCITT as “X.500 Blue Book Recommendations” and in January 1991 by ISO as “ISO/IEC 9594 - The Directory”. X.500 consists of multiple parts:

- X.500 Overview of concepts, models and services [38]
- X.501 Models [39]
- X.509 Authentication framework [40]
- X.511 Abstract service definition [41]
- X.518 Procedures for distributed operation [42]
- X.519 Protocol specifications [43]
- X.520 Selected attribute types [44]
- X.521 Selected object classes [45]
- X.525 Replication [46]
- X.530 Use of systems management for administration of the Directory [47]

The X.500 standard was updated in 1993 and 1997. Currently the 4th edition is being drafted. While the core functionality was defined in the 1988 edition, some areas (e.g. replication and access control) were left to be addressed later. The 1993 edition then brought the addition of an access control model and a replication mechanism called

---

<sup>1</sup>These thoughts were prevalent at a time when the notion of privatisation and de-regulation in the communications sector were still distant in the future. But with the advent of the World Wide Web, which first developed as little information islands then later becoming the true globally interlinked service, the hopes of X.500’s founding fathers were destroyed. Not at least because the idea of a centrally controlled system did not appeal to the “Internet community”.

*shadowing*. Furthermore the original information model specification was significantly revised [98]. Object classes were divided into *abstract*, *structural* and *auxiliary* types; attributes into *operational* and *user* types. *Name forms*, *DIT Structure and Content Rules* were added; as were *Matching Rules*. A definition of the directory schema itself was now stored in the directory. This allowed applications to retrieve information about supported schema items.

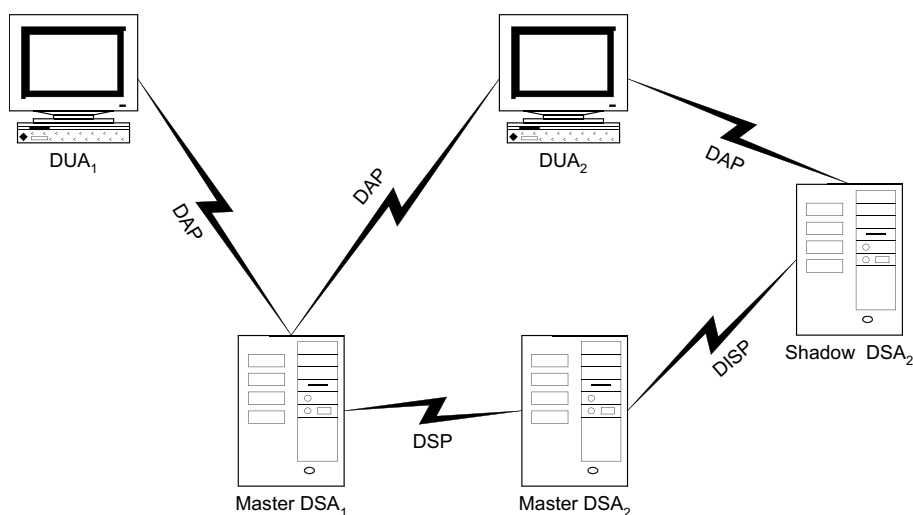


Figure 3.1: Components of an X.500 directory service [30]

As shown in Figure 3.1 several different protocols are used: A client (DUA) will contact a server (DSA) using the *Directory Access Protocol* (DAP). If a DUA requests information that is not held by the DSA itself but the DSA knows where in the directory it may be stored, the DSA can retrieve this information on behalf of its clients. This *chaining* is carried out via the *Directory System Protocol* (DSP). The replication mechanism introduced in the second edition defined yet another protocol: the *Directory Information Shadowing Protocol* (DISP).

X.500 is defined in terms of the *Abstract Syntax Notation* (ASN.1) [36]. This notation is used to specify the protocol data units (PDU) of the X.500 protocols as well as the X.500 information model.

### 3.4 Lightweight X.500 – LDAP

Before this new service could be successfully implemented an important barrier needed to be overcome: its dependency on the OSI protocol stack. Due to being a protocol

residing at the OSI application level, DAP required a lot of resources. But the desktop systems of the time did not have the software or the necessary computing power to implement the full OSI protocol stack. Since these were the systems that should provide access to the directory, research was carried out into finding other means of access. These development efforts resulted in two protocols, which used the readily available TCP/IP stack: the *Directory Assistance Service* (DAS) [82] and the *Directory Interface to X.500 Implemented Efficiently* (DIXIE) [29]. DIXIE had been designed at the University of Michigan (UMich) and quickly became the protocol of choice. The disadvantage of these two protocols was that they were built to interact with one particular X.500 implementation<sup>2</sup> of the original 1988 standard. To overcome this restriction, members of the Open Systems Interconnection – Directory Services (OSI-DS) working group of the Internet Engineering Task Force (IETF) met together and developed the X.500 *Lightweight Directory Access Protocol*. In July 1993 their first RFCs were published [104, 25]. This identified the Proposed Standard for LDAPv2, which was later to be revised by the Access and Searching of Internet Directories (ASID) working group which eventually become a Draft Standard. The resulting RFCs [105, 26, 49] were published in March 1995. LDAPv1 was never published as RFC.

To summarize, the key aspects of LDAP in comparison with X.500 are:

- LDAP is run directly over TCP bypassing session and presentation layers.
- While the actual messages in the LDAP protocol are still ASN.1 structures, most protocol data elements are carried as ordinary strings.
- A lightweight version of BER<sup>3</sup> is used.
- LDAP does not support chaining. Instead, clients are directly referred to a server, which has more information about the requested entry.

The first implementation of LDAP was carried out at University of Michigan and was released in mid 1992. It contained the LDAP daemon (`ldapd`), i.e. a gateway that translated between LDAP and DAP (see Figure 3.2), a client library and also several programs, which were built upon this library.

After noticing that the overwhelming majority of queries<sup>4</sup> to X.500 DSAs came via LDAP, a new server software was written. It incorporated the actual data store mechanism into the LDAP daemon and was named “standalone ldap daemon” (`slapd`, see Figure 3.3). In addition to a performance boost this had the convenient side effect of eliminating the need to set up and maintain a cumbersome X.500 DSA in order to provide directory services.

---

<sup>2</sup>The *Quipu* software package from ISODE

<sup>3</sup> The *Basic Encoding Rules* (BER) define how ASN.1 structures are encoded for storage or transmission in computer systems.

<sup>4</sup>99% at UMich according to [30]

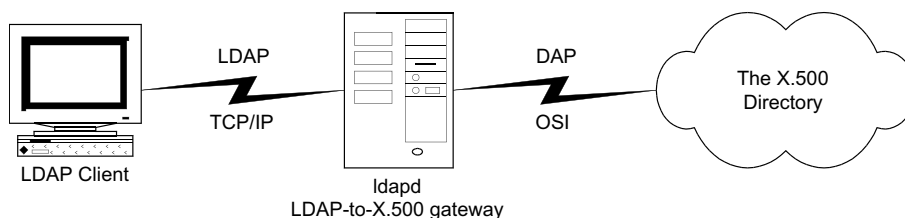


Figure 3.2: LDAP as gateway to X.500

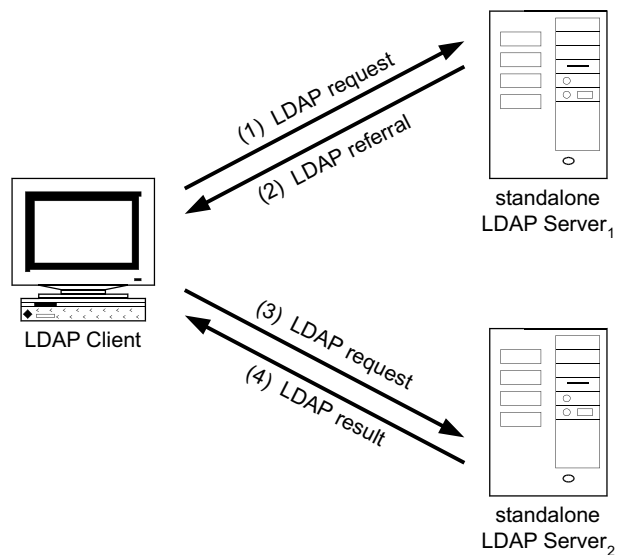


Figure 3.3: Directory architecture with standalone LDAP servers [30]

The source code for the University of Michigan software has always been freely available. The release of the client library especially added to the success of the software package and of LDAP in general. This development kit made it very easy for programmers add LDAP into their applications. (see [chapter 5](#))

## 3.5 LDAPv3

The next step in LDAP development was LDAPv3, an outcome of the Access, Searching and Indexing of Directories (ASID) working group within the IETF. The wide use of LDAPv2 revealed some short comings of the protocol which were addressed by the following new features:

- Schema as of X.500 (93)—LDAPv3 incorporated most of the X.500 (93) schema.
- Feature and schema discovery—A special entry that lists which optional features are supported by a server was defined. This entry is called *rootDSE* (DSA Specific Entry). Furthermore a description of the schema used by the DSA was made available to clients in another special entry—the *subschema subentry*.
- Internationalisation—LDAPv2 used T.61 as its character set. However, in many deployments this requirement has often been disregarded, and data was stored in the local character set.<sup>5</sup> To allow all characters to be stored unambiguously, LDAPv3 uses the UTF-8 encoding scheme of Unicode.
- Referrals—While they were a proprietary extension to LDAPv2 in the University of Michigan code, referrals are now a standardized element of the protocol.
- SASL—The LDAP bind operation was modified to support the *Simple Authentication and Security Layer*. It offers advanced authentication mechanisms as well as integrity and privacy protection. (see [Section 4.4](#))
- Extension mechanisms—by means of *extended operations* and *controls*. Extended operations allow for new features to be implemented without the need to modify the protocol itself. Existing operations may now be augmented with controls. They allow for altering the semantics of an operation, for example, to instruct the server to sort the result set from of a search request before sending it to the client.

LDAPv3 maintains downward compatibility with LDAPv2, i.e. an LDAPv2 client can still access an LDAPv3 server. The RFCs defining LDAPv3 were published in December 1997:

---

<sup>5</sup>In West-European countries mostly ISO-8859-1, also known as Latin-1, was used.

- RFC 2251: “Lightweight Directory Access Protocol (v3)” includes the directory information model and describes the format of the LDAP PDUs. [96]
- RFC 2252: “Attribute Syntax Definitions” defines how attribute types and object classes are to be constructed. [94]
- RFC 2253: “UTF-8 String Representation of Distinguished Names”. [97]
- RFC 2254: “A String Representation of LDAP Search Filters”. [24]
- RFC 2255: “The LDAP URL Format” specifies how to represent references to directory data as a Universal Resource Locator (URL). [28]
- RFC 2256: “A Summary of the X.500(96) User Schema for use with LDAPv3” lists which common X.500 schema elements should be supported by LDAP implementations. [92]

The development effort of LDAP did not stop with the publication of these RFCs. Two new working groups were formed in the IETF: *LDAP Duplication / Replication / Update Protocols* (LDUP<sup>6</sup>) and *LDAP Extension* (LDAPext<sup>7</sup>). LDUP concerns itself with designing non-proprietary replication mechanisms. Progress in this workgroup has been slow. Chartered in 1998, no document from this working group has reached RFC status yet. Arguments centre on how to retain LDAP semantics in multi-master environments. No unilateral agreement has been reached to date as to how this could be achieved.

An important area of LDAPext’s work is concerned with security: Although LDAPv3 includes support for SASL, no mechanisms which avoid plain text passwords were mandated by the 1997 RFCs. They thus carry a note of the IESG<sup>8</sup>, which says that the update operation should not be used until appropriate authentication mechanisms have been defined. These concerns have been addressed recently. [93] discusses which kind of protection would be needed for a specific operation and [21] gives the details on the implementation of data protection on the network by means of TLS (see Section 4.3). These additional RFCs are now considered to be part of the core standard [20]. Other work items of LDAPext include:

- Access control—To define a model by which access control information can be stored in the directory.
- Language tags—The definition of attribute subtypes, which identify the language of a value.
- Server-side sorting and paged results—The definition of controls, which instruct a server to sort the results of a query, or to return the results in small parts at a time.

---

<sup>6</sup><http://www.ietf.org/html.charters/ldup-charter.html>

<sup>7</sup><http://www.ietf.org/html.charters/ldapext-charter.html>

<sup>8</sup>Internet Engineering Steering Group

- Referral management—How to store knowledge information about data held by other servers.
- APIs— A definition of a Java API and an update of the LDAPv2 C API specification to support the new LDAPv3 features.
- CLDAP—A specification of connection-less LDAP using the UDP transport layer.

In December 2000 another LDAP working group has been formed in the IETF: *LDAP (v3) Revision* (LDAPbis<sup>9</sup>). The purpose of this working group is to revise the current documents so that they become suitable for being considered as “Draft Standard”—the next step from “Proposed Standard” in the Internet Standards Process

### 3.6 Text-based Standards

There were some other standards that attempted to become the “lingua franca” for directory access—namely *RWhois* [103] and *Whois++* [7]— but they never gained the support and wide spread use that LDAP has. These are text-based protocols, i.e. the data is carried over a network in human-readable form. *RWhois* and *Whois++* are derived from *Whois* [18] which itself is a descendant of the *Finger* [108] protocol. [16]

One area in which the *Whois* protocol is still used, is the registry service for the DNS system. But there are ongoing experiments to store this information in LDAP servers. The company NSI maintains the registry for the .com, .org and .net domains. Since the autumn of 2000 the Applied Research Department of VeriSign, NSI’s parent company, offers an LDAP access<sup>10</sup> to this information.

---

<sup>9</sup><http://www.ietf.org/html.charters/ldapbis-charter.html>

<sup>10</sup><http://www.ldap.research.netsol.com>

# Chapter 4

## Access Control and Security Layers

Not all directories provide a read-only white-pages service that should be publicly available. Access might be limited to subscribers of the service or only available on a pay-per-view basis, in which case appropriate accounting mechanisms would have to be in place. The data contained in the directory could be of a sensitive nature so that access by non-authorized parties would not be allowed.

The security model for directory services relies on the fact that the protocols, which are used to access the directory, are connection-oriented. It is assumed, that peer entities do not change after a connection has been established. To ensure this in an insecure environment, *security layers* have to be deployed. All operations, which are initiated over a connection, can rely on data that was exchanged earlier. This is used to implement access control mechanisms. At the beginning of a connection the client's identity is established during the *authentication* stage. For each subsequent operation requested by the client, the server checks whether the client is *authorized* to do so.

### 4.1 Authentication

*Authentication* is the process of verifying that a peer entity in a connection is genuine. For use in computer systems the identity of a peer entity is generally represented by an alphanumeric string of some form, for example, DNs, fully qualified domain names, user ids or social security numbers. To assert the claimed identity, credentials are passed between the communication parties. [48]

The following section presents different authentication mechanisms grouped by the type of credentials and their underlying cryptographic properties. All mechanisms described here deal with client to server authentication.

## 4.1.1 Cryptographic background

### 4.1.1.1 One-way-hash-functions

A *message digest*—or *secure hash*—is the result of a so called one-way-hash-function. These functions have the property that it is “computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest” [72]. In other words,  $hash = f(password)$  can be computed easily but not the inverse  $password = f^{-1}(hash)$ .

### 4.1.1.2 Symmetric cryptosystems

In *symmetric cryptosystems*, the same key is used for encrypting and decrypting a message. Such a key must be kept secret between the communications parties, as its disclosure would reveal any transmission encrypted with this key. Symmetric cryptosystems are therefore also called *secret-key cryptosystems*.

Two problems exist with the practical use of secret-key cryptosystems. First, keys have to be transferred over a secure channel. This brings up the problem of initially establishing a secure channel. Secondly, a separate key has to be agreed upon for each pair or communication peers. In a network with  $n$  participants, this would require  $(n(n - 1))/2$  keys to be generated and distributed.

### 4.1.1.3 Asymmetric cryptosystems

The concept of asymmetric cryptosystems—also known as public-key cryptosystems (PKCS)—was first published in [9]. It builds on the idea that keys are used in pairs. Every participant in such a system has two keys. A message encrypted with one of the keys can only be decrypted with the other one. In asymmetric crypto systems one of the keys can be openly distributed—the *public key*. The matching *private key* must be kept secret. For such a system to provide strong security, it is vital that computing the private from the public key is next to impossible, i.e. very time consuming. If a message is to be sent securely, it is encrypted with the addressee’s public key. Then only the recipient that has the matching private key can read it.

Public-Key algorithms are slower than symmetric algorithms by orders of magnitude. In data encryption applications, a hybrid approach is therefore taken. A random key—the *session key*—is generated and encrypted with the asymmetric algorithm. The actual message is then encrypted with a fast symmetric algorithm using the session key.

In asymmetric cryptosystems, keys do not have to be transferred over secure channels since public keys are openly available. However, it remains to be ensured, that

a public key really belongs to its apparent owner. This is accomplished by a *certificate*. In a certificate, a trusted third party certifies the link between public key and its owner. Certificates make use of the second application of asymmetric cryptosystems—*signing*. To sign a message, a digest of the message is generated and encrypted with the sender’s private key. The recipient receives both the original message and the encrypted digest. He then decrypts the digest using the sender’s public key and compares it to the digest generated from the message. If these values match, it is ensured that the message was not changed during transit (integrity protection) and actually comes from the designated sender (non-repudiation).

## 4.1.2 Authentication Methods

### 4.1.2.1 Anonymous

In many applications, it is not necessary—and sometimes not desirable—that clients authenticate themselves to the server. This is usually true for read access to public data. Since the data is public, there is no need to restrict access to specific users. It should be in the service provider’s interest to fashion the access for customers as easy as possible. Any uncalled for efforts might drive the customer elsewhere. From the client’s perspective, not giving away too much information about the user helps in upholding privacy. This prevents the server’s operator from building detailed customer profiles.

If no authentication is performed, the client is said to be “anonymous”. This is also the initial state for the client, after a connection has been opened to the server.

### 4.1.2.2 Plain Text Passwords

The authentication protocol with plain text passwords works as follows:

1. The server asks the client for username and password.
2. The client responds with username and password.
3. The host compares the password with the value on record for the specified username.

The major disadvantage of plain text passwords is that they are easily to compromise. Anyone who can physically eavesdrop on the communication between client and server will learn the password (and the authentication identifier). Once obtained, the credentials can be reused until the password is changed. The attacker might even change the password himself and thus lockout its original holder.

Great care must also be taken to protect the passwords stored on the server. Anybody who is able to read these secrets will get instantaneous access to all accounts on the system. As the server must only be able to verify that a password is correct, there is no need to store the actual passwords themselves. To accomplish this, only hashes of passwords are stored. The initial plain text password protocol mentioned above is thus changed in step 3:

3. a) The host applies the hash function to the transmitted password –and–
3. b) compares the result to the stored hash value.

Unix systems make use of the `crypt(3)` function to store password hashes in the system wide `/etc/passwd` file. `crypt(3)` is a variation of the DES [73] algorithm, where a known plaintext is encrypted with a key generated from the user's password [84]. The resulting cipher text is the output of the hash function. Due to ever-increasing computing power, the `crypt(3)` has become susceptible to brute-force attacks<sup>1</sup> or a refined form thereof—the dictionary attack. In a dictionary attack, not every possible combination of characters is tried out. Instead only common words that are often chosen as passwords and their variations are considered. To strengthen security one needs to make it difficult for a lot of passwords to be tried in a short time. This can be achieved by the following measures:

- Restrict access to hashed passwords. In Unix systems the password hashes are not now stored in the world-readable `/etc/passwd` file but are located in a so-called `shadow` file that is only readable by the super-user.
- Enforce the use of longer and more complex passwords. The key space of 8-character alphanumeric password is 18 million times the size of 5-character lower-case passwords.
- Employ computationally more expensive hash algorithms like MD5 [81] or SHA-1 [72].
- Use salted hash functions: a random string is used as an additional parameter to the hash function. This reduces the probability of attacks with pre-build dictionaries. In such a scenario, the attacker builds a database of inputs and corresponding hash values. Now if a hash value becomes known, only a database lookup is needed to find the matching password. However, with every bit of additional input from the salt, the size of such a database would double. The standard Unix `crypt` system uses 12 bits of salt, which would require 4096 combinations for each password.

---

<sup>1</sup>An average PC (AMD 750 MHz Athlon) can compute the `crypt(3)` hash value of about 100,000 passwords per second. An exhaustive search of passwords consisting of five lower-case characters can thus be performed in about two minutes. [19]

However, all these precautions are useless when plain text passwords can easily be obtained by wiretapping. Their use on the Internet without additional protection mechanism is therefore strongly discouraged.

#### 4.1.2.3 Challenge-Response Mechanisms

A server task is to verify that a client knows the shared secret; the secret itself need not necessarily be transmitted to prove this. Such a protocol can be outlined as follows:

1. The server generates a random string and issues it as challenge to the client.
2. The client computes a response from the challenge using a value derived from its password and sends the response along with its authentication identifier to the server.
3. The server performs the same calculation and compares its result to the response sent by the client.

By choosing a new challenge every time, the protocol prohibits replay attacks. Instead of the actual password, the derived value mentioned in step 2 is stored on the server. As such a value is password equivalent, i.e. its knowledge is sufficient to authenticate, it still must be adequately protected.

Examples for challenge-response mechanisms include *CRAM-MD5* [51] and *DIGEST-MD5* [56]. In contrast to CRAM-MD5, DIGEST-MD5 prevents chosen plaintext attacks<sup>2</sup>.

#### 4.1.2.4 Kerberos

Kerberos [69] is a network authentication protocol that provides strong authentication for client/server environments. It makes use of symmetric encryption for authentication purposes [74]. Each participant in the system—a *principal* in Kerberos terms—shares a secret key with the central *Key Distribution Centre* (KDC). The KDC actually consists of two components: the *Authentication Service* (AS) and the *Ticket-Granting Service* (TGS).

The credential in Kerberos is the *ticket*. To gain access to a service, the client requests a *Service Ticket* from the TGS. The client can then prove its identity to the service with this ticket. However, the client must first establish its identity with the TGS before any Service Tickets will be issued. To this end, the client authenticates with the AS and

---

<sup>2</sup>In a chosen plaintext attack, the attacker has the opportunity to provide the encryption system with arbitrary plain text and then examine the resulting crypt text.

receives an initial ticket—the *Ticket-Granting Ticket* (TGT). The TGT is then used in all subsequent requests to the TGS. This procedure makes Kerberos a *Single Sign-On* protocol. The user only has to provide his password once when authenticating to the AS. To prevent misuse of tickets, they expire after a defined period<sup>3</sup>. [58]

Version 5 [52] is the current version of the Kerberos standard.<sup>4</sup> Applications usually do not call Kerberos functions directly. Instead they use the *Generic Security Service Application Program Interface* (GSSAPI) [60]. GSSAPI is a security framework that abstracts from underlying protocols. A GSSAPI mechanism that uses Kerberos 5 as underlying protocol (GSS-KRB5) has been defined in [59].

#### 4.1.2.5 X.509

X.509 [40] was originally designed as authentication framework for X.500 directories. The strong authentication method defined therein makes use of public key cryptosystems and is now widely used for secure email (S/MIME) and secure connections (SSL and TLS). The trusted third party is the *Certification Authority* (CA). The CA is responsible for creating *certificates*—the credentials in X.509. A certificate contains a public key, the distinguished name of the entity that is to be associated with this key, and the CA's distinguished name. These elements are then signed by the CA with its private key, thus making the certificate unforgeable.

## 4.2 Authorization

*Authorization* is the process of determining, whether the requested access to a resource may be granted. The rules—often called *Access Control Lists* (ACL)—that govern this process are expressed in terms of *Access Control Factors* (ACF). Common ACFs in the LDAP context are [93]:

- The authentication method.
- The *authorization identity*.
- The kind of requested operation (e.g. search or modify).
- The entries and attributes affected by the operation.

---

<sup>3</sup>The default ticket lifetime in Kerberos 5 is ten hours.

<sup>4</sup>Free implementations for Unix systems are available from the Massachusetts Institute of Technology, <http://web.mit.edu/kerberos/www/>, and from the Royal Institute of Technology, Stockholm, Sweden, <http://www.pdc.kth.se/heimdal/>. In Windows 2000, Microsoft has adopted Kerberos as primary authentication mechanism. [63]

- Factors outside the LDAP protocol itself, like source IP-address or encryption strength.

Generally, the authorization identity is equal to authentication identity. However, this does not necessarily have to be the case. A *proxy policy* may allow certain users to act on behalf of others. They authenticate as themselves but then take the role of another user for the purpose of authorization.

## 4.3 Security Layers

*Security layers* protect data on its way between the communication peers from

1. being altered in transit. This is known as *integrity protection*.
2. being overheard by an unauthorized person. This is known as *privacy protection*.

Security layers are commonly provided at the transport layer. The relevant standards in this field are the *Secure Socket Layer* (SSL) [13] and its successor, *Transport Layer Security* (TLS) [8]. But also Kerberos and SASL include the ability to establish a security layer. With *IPsec* a standard has emerged, which offers support for a security layer at the network layer of the Internet Protocol.

LDAP over SSL (LDAPS<sup>5</sup>) is available in most mainstream products. Support for TLS via the StartTLS extended request [21] is so far only available in OpenLDAP and MessagingDirect products. (See also Figure 10.1)

## 4.4 SASL

The *Simple Authentication and Security Layer* (SASL) provides “a method for adding authentication support to connection-based protocols” [71]. SASL allows application layer protocols to make use of different authentication mechanisms. To implement SASL, a protocol needs to define an operation by which a user can authenticate himself to a server. This operation must include the name of the authentication mechanism to use and may include an authorization identifier if it is different from the authentication identifier.

Often some way is provided, by which a client can determine which mechanisms the server supports. In LDAP, these can be requested by retrieving the “supported-SaslMechanisms” attribute from the server’s rootDSE.

SASL mechanisms that have been defined include:

---

<sup>5</sup>LDAPS uses the well-known TCP port 636.

- ANONYMOUS (defined in [76], see also 4.1.2.1)
- PLAIN (defined in [77], see also 4.1.2.2)
- DIGEST-MD5 (defined in [56], see also 4.1.2.3)
- GSSAPI (defined in [71], see also 4.1.2.4)
- EXTERNAL (defined in [71]). By using the EXTERNAL mechanism, the authentication identity is determined from an external source, e.g. from certificates that have been exchanged during the negotiation of a security layer with SSL or TLS.

## 4.5 Authentication Methods for LDAP

[93] identifies three levels of protection and their appropriate authentication mechanisms:

- Public read-only directories do not require any authentication, i.e. anonymous authentication is sufficient.
- For connections that require authentication, the DIGEST-MD5 mechanism must be used to protect the password from passive eavesdropping attacks.
- If, in addition to the password, the whole session is to be protected, the use of TLS in combination with plain-text passwords or the SASL EXTERNAL mechanism is recommended. TLS also provides protection against active man-in-the-middle attacks.

# Chapter 5

## Developing LDAP-enabled Software

Software development kits (SDK) greatly facilitate a programmer's job. They implement a protocol and allow access to it by a set of high-level functions—often called an *Application Programming Interface* (API). The programmer can thus concentrate on writing his application and does not need to worry about lower layer aspects of the protocol. LDAP SDKs are available for a wide range of programming languages<sup>1</sup>. The more important ones will be dealt with in the following section.

### 5.1 C

The ancestor of LDAP SDKs is the SDK included in the University of Michigan LDAP distribution. Its API was published in an informational RFC [27] which has been the normative reference for most SDKs—not just those for the C programming language. Since it describes how to build client applications that access an LDAPv2 service, work is underway to standardise an API for the additional elements of LDAPv3 [31]. Although this document is still in draft status, the functions described therein have been implemented by all current SDKs.

The C API closely follows the LDAP functional model. For each operation in LDAP (bind, search, compare, add, modify and delete) there exists a corresponding C function. For instance, a skeleton program to search the directory would look like this [87]:

---

<sup>1</sup>[102] gives examples for nine different languages.

```

LDAP *ld;
LDAPMessage *result, *e;
BerElement *ber;
char *a;
char **vals;

/* Initialize the connection to the LDAP server */
ld = ldap_init( HOSTNAME, PORT_NUMBER );

/* Authenticate with a plain text password */
ldap_simple_bind_s( ld, BINDDN, PASSWORD );

/* Search for entries that match FILTER and retrieve all attributes in ATTRIBUTES. */
ldap_search_ext_s( ld, FIND_DN, LDAP_SCOPE_SUB, FILTER,
    ATTRIBUTES, 0, NULL, NULL, LDAP_NO_LIMIT,
    LDAP_NO_LIMIT, &result );

/* Iterate over the entries returned. */
for ( e = ldap_first_entry( ld, result );
    e != NULL; e = ldap_next_entry( ld, result ) ) {
    /* Iterate over the attributes for this entry */
    for ( a = ldap_first_attribute( ld, e, &ber );
        a != NULL; a = ldap_next_attribute( ld, e, ber ) ) {
        /* For each attribute, print the attribute name and values. */
        if ( (vals = ldap_get_values( ld, e, a )) != NULL ) {
            for ( int i = 0; vals[i] != NULL; i++ ) {
                printf( "%s: %s\n", a, vals[i] );
            }
        }
    }
}
}

```

Most of the functions exist in a synchronous and asynchronous form. Upon calling a synchronous function the application will block until the result becomes available. Since this should not happen in applications that have a graphical user interface, the programmer should make use of the asynchronous interface in this case. A program written this way can initiate further requests, while it is still waiting for incoming results. It also enables the user to abort an ongoing operation that is taking longer than intended. The asynchronous interface however is by its very nature more complex to program.

## 5.2 Perl

Two libraries are available for developing LDAP-enabled software with Perl. First there is *PerLDAP*<sup>2</sup>, a Perl wrapper around the C SDK. Then there is *Perl-LDAP*<sup>3</sup> which is written purely in Perl. While both modules support the basic LDAP operations as well as LDAP over SSL and import/export of LDIF<sup>4</sup> files, only Perl-LDAP offers support for LDAPv3 controls, SASL authentication and schema management. It also includes a module for writing and reading data in the Directory Service Markup Language (DSML<sup>5</sup>), which is an XML dialect for representing directory information.

## 5.3 Java

Two SDKs are available for Java, which facilitate the writing LDAP-enabled programs. The first one is the *Netscape Directory SDK for Java* [75, 101], which is modeled after the C API. It is also available in source code from the Mozilla project<sup>6</sup>. Standardisation of its API is currently a task for the LDAPext working group [32].

The other way to access LDAP directory services is by means of the *Java Naming and Directory Interface* (JNDI) [57]. It takes a more abstract approach than the Netscape SDK. JNDI provides a unified interface which can be used to access different directory services. For each such service a separate module must exist, which handles the underlying protocol (see Figure 5.1). In JNDI 1.2 these so-called “providers” are available for LDAP, a CORBA naming service, NIS, DSML, DNS, native NDS and the local file system.

JNDI takes the job of protocol management from the developer. In areas where high performance solutions are required, this additional abstraction layer might not be advisable. With the Netscape SDK the programmer has the choice in deciding when new connections to the server need to be opened. It also allows for a pool of pre-connected associations with the server. This can, for example, be used in Java Servlets to minimize the overhead of processing HTTP requests.

---

<sup>2</sup><http://www.perldap.org>

<sup>3</sup><http://perl-ldap.sourceforge.net>

<sup>4</sup>LDAP Data Interchange Format [15]

<sup>5</sup><http://www.dsml.org>

<sup>6</sup><http://www.mozilla.org/directory>

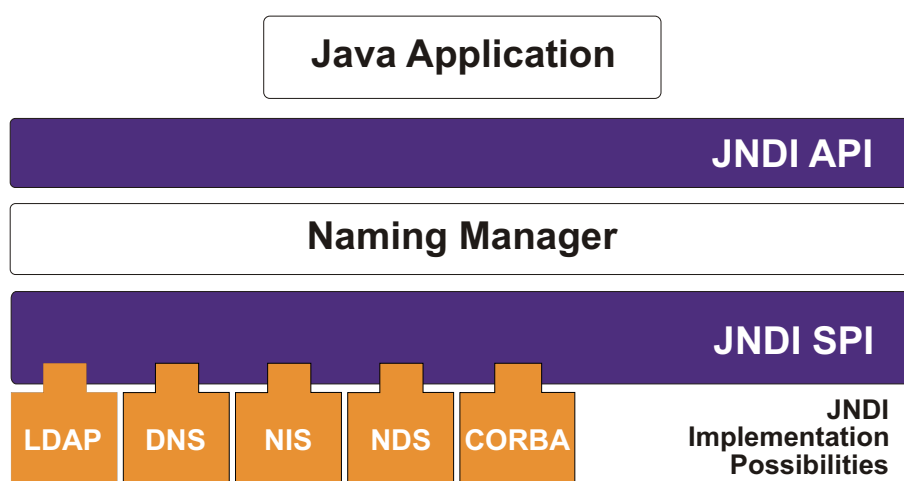


Figure 5.1: JNDI architecture [88]

## 5.4 PHP

PHP<sup>7</sup> is a server-based scripting language which is mainly used for building dynamic web sites. Its LDAP module is built using a C SDK and to date only supports LDAPv2. However, this is still sufficient to create a web front-end to a white-pages service.

## 5.5 Python

Python is an “interpreted, interactive, object-oriented programming language”<sup>8</sup>. The ldap module for Python<sup>9</sup> is a wrapper around an LDAPv2 C SDK and follows the API defined in [27]. However, return codes indicating an error are converted to a Python exception. Likewise, the native list datatype is used for arguments and return values of functions.

## 5.6 ADSI

ADSI is the acronym for *Active Directory Service Interfaces* [64] and is only available for the Windows platform. It is based on Microsoft’s Component Object Model

<sup>7</sup><http://www.php.net>

<sup>8</sup><http://www.python.org/>

<sup>9</sup><http://python-ldap.sourceforge.net/>

(COM) and thus can be used by any application that supports COM. Besides traditional development environments such as Visual C++ or Visual Basic this also allows macros in Microsoft Word and Excel or scripting languages like VBscript to access directory services.

Much like JNDI, ADSI offers an abstract object oriented interface that uses providers to access various data sources. Besides the ubiquitous LDAP support, providers exist for Windows NT 4.0 domain controllers, Novell Directory Services and IIS, Microsoft's Internet Information Server.

# Chapter 6

## Directory Software for Linux

### 6.1 Directory Servers

This section describes LDAP server software available for Linux from different vendors.

#### 6.1.1 OpenLDAP

The OpenLDAP Project is dedicated “to develop a robust, commercial-grade, fully featured, and open source LDAP suite of applications and development tools.”<sup>1</sup> OpenLDAP’s code is derived from the University of Michigan LDAP v3.3 distribution<sup>2</sup> and is developed under a BSD style licence<sup>3</sup>. Two major versions exist in the OpenLDAP Project: OpenLDAP 1.x, the latest release being 1.2.11, is LDAPv2 only. The actively maintained branch that strives for full LDAPv3 compliance is 2.x. OpenLDAP 2.0 was first released in August 2000. The latest version (2.0.7) from the stable development branch is the version used in this thesis. There is also a “head” branch, which holds the more experimental extensions, which are not yet considered stable enough for general usage.

All applications reside on top of two libraries: *liblber*, which handles the ASN.1 encoding, and *libldap*, which handles the application layer protocol. The package includes

---

<sup>1</sup>OpenLDAP Project Overview, <http://www.openldap.org/project/>

<sup>2</sup><http://www.umich.edu/~dirsvcs/ldap/>

<sup>3</sup>This kind of license basically means that distribution, use and modification of the software is allowed as long as the original copyright holder is properly credited and all warranties regarding the functionality of the software are disclaimed.

all the applications, which were already present in the distribution from the University of Michigan, namely the command line tools to search and update the directory, a standalone directory server (*slapd*) and *slurpd*, the standalone ldap update replication daemon. *slurpd* can be used to implement a single-master replication architecture.<sup>4</sup> OpenLDAP does not have the ability to do online back-ups. The server has to be shut-down before administrative utilities can be used to dump or restore the database to or from an LDIF file.

The *slapd* directory server consists of a front-end, which handles the communication with a client according to the LDAP protocol specification, and a number of back-ends, which implement non-volatile data storage. The most commonly used back-end is called *gdbm*. It stores directory information using the Berkeley embedded database system from Sleepycat.<sup>5</sup> Other back-end modules have been written to allow LDAP access to various existing data sources:

- *back-passwd* provides read-only access to `/etc/passwd`.
- *back-shell*, *-perl*, *-tcl* call external programs or scripts to access data.
- *back-java* allows interaction with Java classes using the Java Native Interface (JNI). [85]
- *back-sql* provides access to relational databases via ODBC.
- *back-ldap* forwards requests to other LDAP servers, optionally rewriting queries and results. This module could be used to build an LDAP proxy server.
- *back-whois* provides a gateway between LDAP and the whois service for DNS registry information.<sup>6</sup>
- *back-dnssrv* is intended to build a root LDAP server for the DC style naming schema. On a query for “dc=example, dc=com” the server will look up the LDAP server for the “example.com” domain using DNS SRV records [17] and then refer the client to it.

OpenLDAP offers a wide range of security features. It makes use of the libraries from the OpenSSL<sup>7</sup> and Cyrus SASL<sup>8</sup> projects to support transport layer security and offer the most complete support for authentication features.

Some features, which would be needed to make OpenLDAP a “really grown up” server, are currently only available from the head branch. This includes support for multi-master replication and the ability to store access control information in the directory

---

<sup>4</sup>An experimental multi-master implementation exists in the head branch.

<sup>5</sup><http://www.sleepycat.com/>

<sup>6</sup><http://www.usrlocalsrc.org/BACK-WHOIS/index.html>

<sup>7</sup><http://www.openssl.org>

<sup>8</sup><http://asg.web.cmu.edu/sasl/sasl-library.html>

itself. Until the latter feature becomes available, access control lists will have to be statically configured in a config file. Changing one of these directives requires a restart of the server, as do changes to the schema. As the directives are evaluated on a first match basis, great care must be taken in writing them to achieve the desired result. The common approach is to write rules that affect large parts of the directory and then to specify stricter or more relaxed rules for subordinate areas. However, this scenario is not possible because in practice, evaluation is stopped at the first matching rule.

Directories for large enterprises or directories with a flat DIT structure can contain thousands of entries in a container. Clients trying to browse such a server might not have the required resources to handle this kind of result set or could be connected to the server over a low-bandwidth connection. LDAP controls have therefore been defined which allow the client to specify that the server should only return parts of the complete result set at a time. The OpenLDAP slapd does not support either of the two proposed mechanisms [99, 1]; however, they are supported in the client library. In a similar vein, the SDK supports a control that instructs the server to sort matching entries before returning them [33], while the OpenLDAP server does not.

Another area which could be improved, is documentation—especially documentation that keeps up with rapid development. To fully keep up with the most recent versions, one needs to monitor the relevant mailing lists. They are a valuable source of information and can offer help in the case of problems arising as the developers themselves respond to postings on these lists.

### 6.1.2 Netscape

Most of the people who originally developed LDAP at the University of Michigan were later employed by Netscape to develop one of the first commercial LDAPv3 servers. Version 1.0 of the *Netscape Directory Server* was released in 1996 [35].

Today the directory activities lie with “iPlanet e-commerce solutions”, an alliance formed by Netscape, Sun and AOL in 1999. The current release of the directory server software is version 4.13. In March 2000 Sun Microsystems also acquired Innosoft International, Inc., whose products included *Innosoft Distributed Directory Server*. The upcoming 5.0 release of the iPlanet directory server will incorporate Innosoft’s technology and will offer advanced features such as multi-master replication and LDAP chaining. The beta version of this product has been available since December 2000, not for the Linux platform however, and is therefore not been dealt with by this thesis.

iPlanet offers a plug-in interface [34] to extend the capabilities of the server. With it, a developer can write modules that provide additional checks on LDAP operations, add new syntaxes and matching rules, implement support for extended operations or new authentication mechanisms, and can even use a custom database back-end.

The Netscape Directory Server comes with a Java based management tool and a HTTP-to-LDAP gateway.

### 6.1.3 IBM

IBM's *SecureWay Directory* 3.2 is "provided at no cost"<sup>9</sup> and runs on AIX, OS/390, OS/400, Solaris and Windows NT. A "Technology Preview" version for Linux is currently available on request. This version lacks support for SSL/TLS and GSSAPI because the required software packages (IBM Global Security KIT and IBM Network Authentication Service) are not yet available for the Linux platform.

The provided plug-in API is compatible to that which would be found in the Netscape Directory Server.

IBM uses the relational DB2 Universal Database as a back-end storage mechanism for the SecureWay directory server. To administrate the server a web based tool is provided, which is not yet available for Linux. Additionally the software package includes a schema-aware Java application to manage the directory.

### 6.1.4 MessagingDirect

The *MessagingDirect M-Vault*<sup>10</sup> directory server has its roots in the ISODE software. ISODE stands for *ISO Development Environment* and is a software development kit for OSI systems. Among other services provided, it also includes support for X.500. The implementation of the X.500 (88) standard (*Quipu*) is freely available<sup>11</sup>. Later releases by the ISODE Consortium (and now MessagingDirect) are available to the members of the German academic research network under a special license.

The current source code release R6.0v3p2 implements both the X.500 (93) standard and LDAPv3. Like OpenLDAP, it uses the Berkeley database system as a back-end storage mechanism. An LDAP-to-HTML gateway (WebXS) is included in the package as well as two TCL script based administration tools, one for the server administration (EDM) and one for data maintenance (DDM).

### 6.1.5 Novell

Originally designed as an embedded directory for NetWare, Novell's directory server is now available as standalone product for range of operation systems including Linux. It will be discussed in [chapter 7](#).

---

<sup>9</sup><http://www-4.ibm.com/software/network/directory/>

<sup>10</sup><http://www.messagingdirect.com/products/IC-6097.html>

<sup>11</sup><http://www.directory.dfn.de/isode/isode-frei.html>

## 6.2 Administration Tools

### 6.2.1 Command line tools

All directory servers mentioned above come with a set of command line tools including *ldapsearch* and *ldapmodify*. While the average user will probably prefer an application with a GUI, these programs can be valuable tools in the hands of an experienced administrator. They are especially helpful in narrowing down the source of a problem, as problems that originate on the client side can be reduced to a minimum.

### 6.2.2 lbe

*lbe*<sup>12</sup> stands for LDAP Browser/Editor. It is written in Java and therefore not specific to Linux. Originally a standalone application, recent versions can also be used as an applet. The user interface follows the classic file manager layout: A tree pane on the left, which displays the DIT hierarchy, and a tabular pane on the right showing the attributes of the currently selected entry.

Outstanding is *lbe*'s drag&drop support. This feature can be used to move or copy entries in the DIT but also to add the dragged entry's DN to an attribute of another entry. This way, an administrator can intuitively manage group memberships or role assignments. *lbe* lacks support for schema information offered by the directory server. Instead, it employs a template mechanism to aid the administrator in creating new entries.

### 6.2.3 gq

*gq*<sup>13</sup> is GTK-based LDAP client for the X Windows System and builds upon the OpenLDAP 2.0 client libraries. It supports TLS and is able to make use the schema information available in LDAPv3 servers. In its current version 0.4.0, *gq* is not able to handle UTF-8 correctly. Each byte of a multi-byte character is displayed as a separate character.

---

<sup>12</sup><http://www.iit.edu/~gawojar/ldap/>

<sup>13</sup><http://biot.com/gq/>

## 6.3 Directory-enabled Applications

### 6.3.1 Name Service Switch

Applications under Linux interact with the system by calling functions of the C library. As part of the C library an API exists, which allows data to be searched for in administrative system databases. This API includes functions to enumerate (e.g. `getpwent` to consecutively list all users) and search (e.g. `getpwnam` to search for a user by name) for users and groups as well as IP services, protocols, hosts and networks.

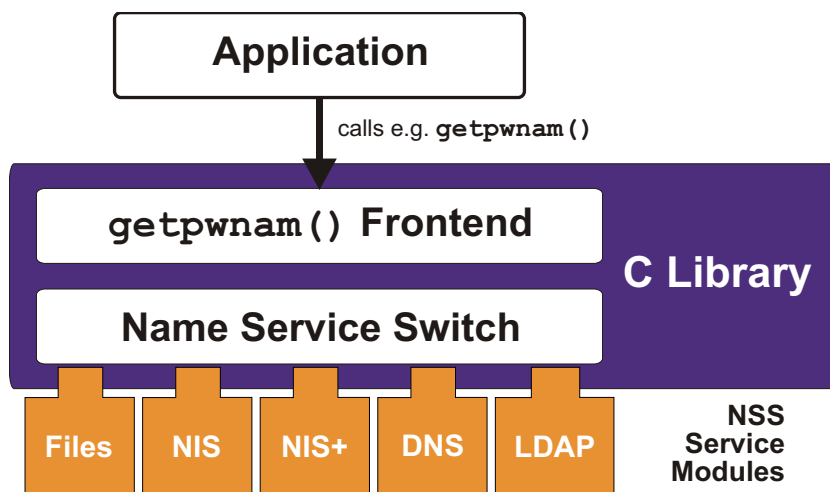


Figure 6.1: Name Service Switch Architecture

Originally, these databases were stored as plain text files on a host's local file system (e.g., `/etc/passwd` or `/etc/groups`). This was followed by proprietary extensions that used a special entry in these files to signal that data should instead be retrieved from other sources. For Solaris 2, Sun designed a modular architecture—called *Name Service Switch*—that defines an interface between the core C library and a service module that implements a particular type of information storage (see Figure 6.1). The GNU C library, which Linux uses, follows this concept.

`nss_ldap`<sup>14</sup> is a NSS service module that retrieves information from an LDAP server. It is the reference implementation of [23], which proposes a schema for storing name service related data in a directory.

<sup>14</sup>[http://www.padl.com/nss\\_ldap.html](http://www.padl.com/nss_ldap.html)

### 6.3.2 Pluggable Authentication Modules

The idea behind *Pluggable Authentication Modules* (PAM) [83] is to provide a framework for login services. PAM defines a unified interface, which applications can use to authenticate and authorize users. In its design, PAM allows for a fine-grained configuration. It can be configured on a per application basis and allows multiple authentication methods to be used in sequence. PAM's functionality is divided into four areas:

- *Authentication management* includes functions to identify and to authorize a user.
- *Account management* provides authorization functions. Generally, this includes checking for password and account expiration.
- *Session management* handles tasks like accounting.
- *Password management* provides the user with the ability to change his authentication credentials, e.g. his password.

*pam\_ldap*<sup>15</sup> implements authentication, account and password management with LDAP as data store. To authenticate a user, *pam\_ldap* first searches for the provided username in the directory. If a match is found, *pam\_ldap* then tries to bind to the entry found. It does this with call to `ldap_simple_bind`. To protect the transferred password, security layers provided by LDAPS or StartTLS are supported.

In the account management stage, *pam\_ldap* retrieves the information stored under the user's entry to be able to check for account and password expiration, groupmembership and host restrictions.

Finally, the password management section provides the user with the ability to change his password on the LDAP server. Besides a plain replace on the "userPassword" attribute, *pam\_ldap* supports the LDAP password modify extended operation [106] as well as password changing for NDS and Active Directory (see Section A.1).

### 6.3.3 Sendmail

Sendmail<sup>16</sup> is the default *Message Transfer Agent* (MTA) on most Linux systems. Since version 8.10 it includes the ability to route email traffic based in information stored on an LDAP server. By default this `ldap_routing` feature implements the schema proposed in [53]. However, it can be configured to support arbitrary attribute names if necessary, and if enabled for a domain, sendmail will do an LDAP lookup

---

<sup>15</sup>[http://www.padl.com/nss\\_ldap.html](http://www.padl.com/nss_ldap.html)

<sup>16</sup><http://www.sendmail.org>

for every recipient from such a domain. It first searches for full email address (e.g., `user@example.com`) and then, if nothing was found, for the domain portion (i.e. `@example.com`). Based on the return values, Sendmail will either forward the unchanged mail to the specified mail host or substitute the recipient address with the new value and attempt a new delivery.

The upcoming 8.12 release will also be able to natively use LDAP look-ups for a wide range of configuration properties (i.e. Aliases, Maps and Classes).

### 6.3.4 Apache

Apache<sup>17</sup> is the most popular web server on the Internet<sup>18</sup>. Apache can make use of a directory service to verify passwords transmitted in the *HTTP Basic Authentication Scheme* [12]. To this end several modules have been written, the most advanced of which being *auth\_ldap*<sup>19</sup>. *auth\_ldap* supports LDAPS and TLS and implements a cache, which reduces the load on the LDAP server and gives better response times. Its authentication functionality is identical to that of *pam\_ldap*. To specify access restrictions, ACLs are added to the Apache configuration file. Authorized access to web pages can be granted to any user who has successfully authenticated, to a list of users and by group membership.

---

<sup>17</sup><http://www.apache.org>

<sup>18</sup><http://www.netcraft.com/survey/>

<sup>19</sup>[http://www.rudedog.org/auth\\_ldap/](http://www.rudedog.org/auth_ldap/)

# Chapter 7

## Novell Directory Services

### 7.1 Directory Server

For a long time, *Novell NetWare* has been the Network Operating System (NOS) of choice for most networks of IBM-compatible PCs that used to run Microsoft DOS or Windows 3.1. A dedicated server offered file- and print-services for a number of client PCs. All administrative information about objects in such a network such as users, groups, printers and print queues was stored in a proprietary directory called the *Bindery*. However, this information was not shared between servers. If a user needed access to multiple servers, his account had to be created on all of them separately. This changed in 1993 when *Novell Directory Services* (NDS) [78] were introduced with version 4 of NetWare. Although NDS is not an X.500 DSA, its design is based heavily on concepts laid out in X.500. NDS follows the models specified therein but uses the *Novell Directory Access Protocol* (NDAP), which is built on top of the *NetWare Core Protocol* (NCP). With NDS, centralized management of large computer networks, which could expand over workgroups, organizational units and sites, from a single administrative point, became feasible [54].

An additional component called *LDAP services for NDS* was made available in 1997, which allowed LDAP access to the directory. The LDAP services for NDS would translate LDAP operations into native NDS calls. This resulted in a degraded performance and the need arose to create mappings between LDAP object classes and attributes types to their respective NDS equivalents.

Due to strong competition in its core business of file- and print-services from Windows NT and increasingly Unix—especially Linux—based solutions with Samba, Novell has lost a lot of its market share in this sector. The strategy in recent years has thus been to expand into new business areas where existing technology and available knowledge could be put to good use.

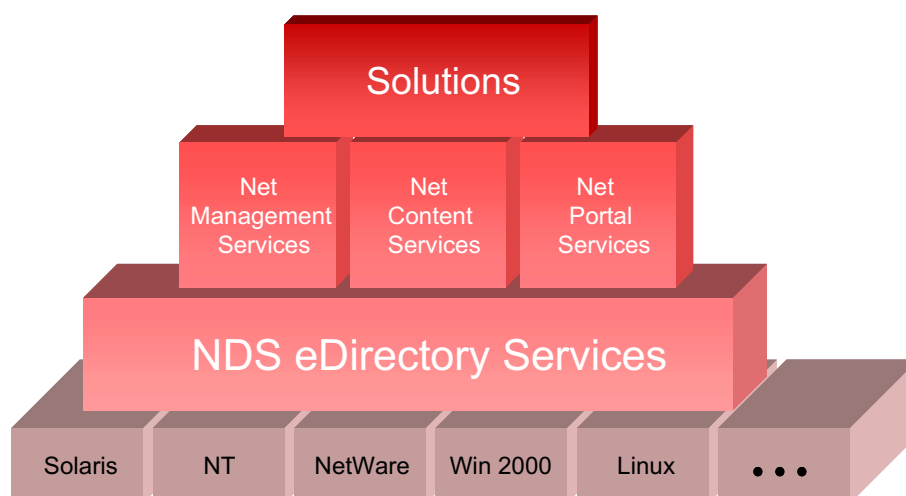


Figure 7.1: DENIM Architecture [79]

This approach has been called *Directory-Enabled Net Infrastructure Model* (DENIM, see Figure 7.1). Building on existing products, new value added services and solutions are provided. The key component in this model is NDS. All higher-level services rely on it for data management. To expand the usage of NDS, it has been ported to platforms other than NetWare and has subsequently been renamed to *Novell eDirectory*. eDirectory, whose first version (Version 8.5) was released in October 2000, is a standalone LDAP server that is available for NetWare, Windows NT/2000, Solaris, Tru64 and Linux.

eDirectory possesses a very versatile replication architecture. Generally, all servers have multi-master capabilities but can become a read-only replica if necessary. Also, so-called *filtered replicas* are possible. Such a replica will only hold a subset of attributes. Such a replica can be used to create a publicly accessible server, which holds only non-confidential information such as email addresses and telephone numbers. In this way the directory as a whole can still be used for more sensitive data like, for example, social security numbers. Another way to distribute a directory is by subtrees—*partitions* in Novell terms. Each host that runs eDirectory can handle multiple partitions, i.e. it is not limited to a single subtree of the DIT.

Another interesting feature is eDirectory's ability to automatically generate indices. It analyzes query patterns by looking for frequent searches on unindexed attributes and will then create new indices for the effected attributes to increase performance.

## 7.2 Administration Tools

Novell's administration utilities have come a long way, i.e. from DOS based applications like *pconsole* to the now very powerful *ConsoleOne*. *ConsoleOne* is a Java application, which underlines Novell's cross platform strategy. With all operating systems where eDirectory is available, it presents the same user interface to the administrator. Besides the normal management tasks, the administrator can use *ConsoleOne* to access the PKI component of eDirectory for issuing or revoking X.509 certificates. The application further allows for the management of DirXML settings.

*iMonitor* is another tool that comes with NDS. It is an embedded web server and allows browsing of the server's configuration and statistics as well as the directory data itself.

## 7.3 Directory-enabled Applications

Additional software from Novell that relies on eDirectory is *GroupWise*, Novell's messaging and time management solution, and *BorderManager*, the company's firewall suite, and *DirXML*.

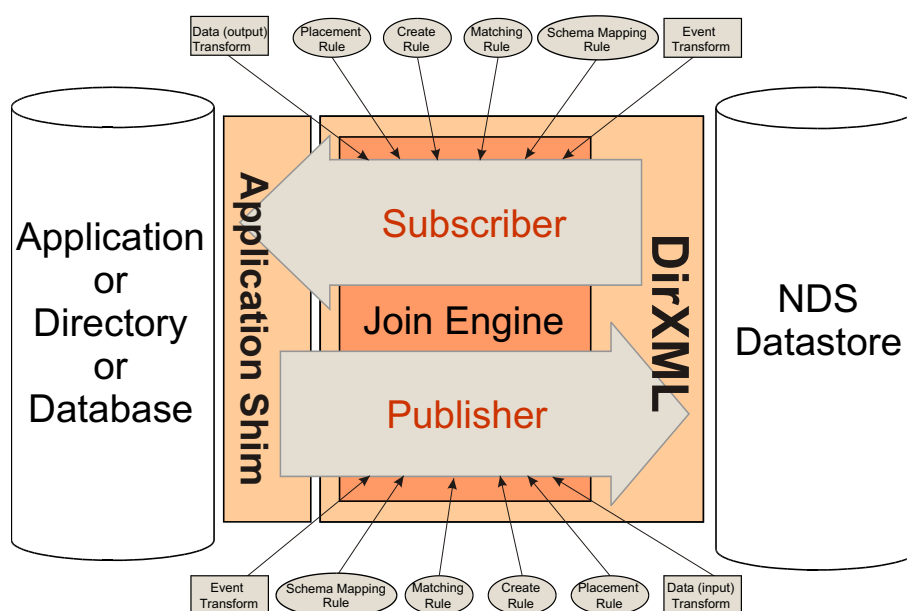


Figure 7.2: DirXML

DirXML [80] is a framework that allows synchronization of different data sources. The “XML” in its name stems from the fact that the rules and style sheets (see Figure 7.2) that govern the data exchange, are written in XML. The publisher/subscriber

schema enables DirXML to preserve data ownership: employee's personal information is usually managed by the human resource department, whereas email addresses and network accounts fall into the responsibility of the IT department. To keep a department from writing to data it is not authoritative for, the join engine will accept updates only from publishers. DirXML ships with drivers for Lotus Notes, Microsoft Active Directory, Microsoft Exchange, NDS and Netscape's directory server. It also includes a C++ and Java API which allows the development of drivers for legacy applications.

# Chapter 8

## Microsoft Active Directory

### 8.1 Directory Server

Active Directory is the integrated directory server in Microsoft's line of Windows 2000 Server operating systems. It replaces the domain model in Windows NT 4.0 [61] and is based on the data models and concepts of X.500. A domain in Windows NT is a collection with a flat hierarchy and is limited to hold about 40,000 objects. In contrast, tests have shown that Active Directory can, on appropriate hardware<sup>1</sup>, scale up to more than 100 million entries [3].

In Windows 2000 Microsoft has taken a step away from proprietary mechanisms. Data in Active Directory can be accessed by any LDAPv3 compliant application. In another area, DNS replaces NetBIOS as the the standard protocol for service location. Native Windows 2000 clients now issue a request for a DNS SRV record to find a host offering a specific service. The Windows 2000 namespace is therefore the same as the one defined by the DNS. As a result, Microsoft has chosen to adopt the domain component naming scheme [50] for Active Directory.<sup>2</sup>

The smallest area of partitioning in Active Directory is the *domain*. Domains with a coherent naming scheme form a *domain tree*. Multiple trees can be joined in a *forest*, (see Figure 8.1) and will share the same schema and have a mutual trust relationship.<sup>3</sup> All Windows 2000 servers that run Active Directory are called *domain controllers*. Active Directory is a pure multi-master environment. Each domain controller maintains

---

<sup>1</sup> A Compaq Proliant 8500 8-way PIII Xeon with 2GB RAM and an ESA 12000 (Fibre channel) RAID controller with 48 18GB hard disks

<sup>2</sup>The "organizationalUnit" object class is retained to group entries within a Windows 2000 domain.

<sup>3</sup> These trust relationships allow that objects from one domain might be granted access to resources in another domain. For example, the administrator of domain "campusA" can allow users from "campusB" to use printers located in "campusA".

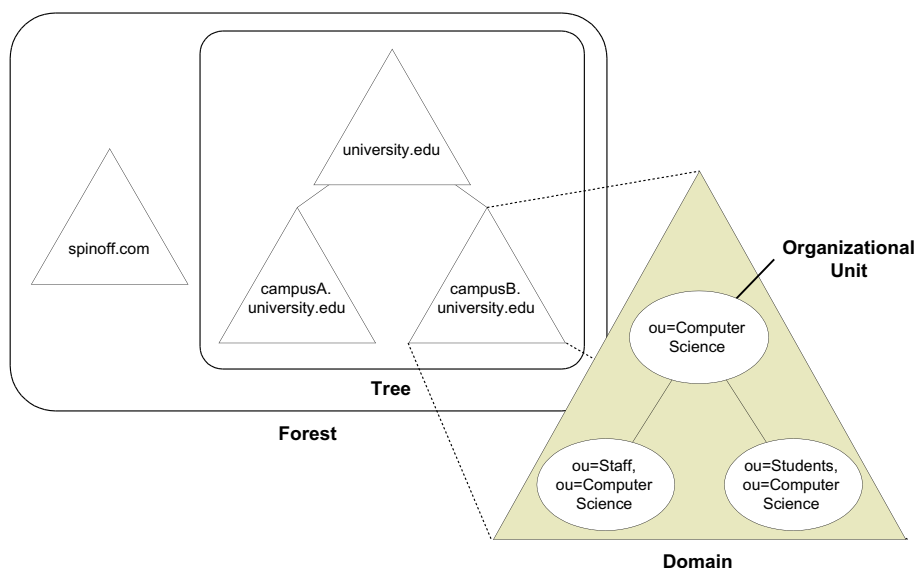


Figure 8.1: Active Directory Structure

a read-writable version of the directory for its domain. A Windows 2000 domain controller cannot manage directories for more than one domain. If an online backup is required for a domain, a dedicated host has to be set up. However, domain controllers can hold a copy of the forest wide *global catalog*. In the global catalog a read-only subset of commonly used attributes for all entries in the forest is stored and indexed. This allows for forest wide searches to be performed without the penalty of having to contact multiple domain controllers.

Computers managed by Active Directory can be assigned to a *site*. A site is a location where all hosts are connected by a fast interlink—usually an Ethernet LAN. To ensure that changes are properly replicated to all domain controllers, the replication topology is maintained automatically by the *Knowledge Consistency Checker (KCC)*. The KCC will create a bi-directional replication ring for all intra-site domain controllers. Inter-site connections will only be created if two sites have been marked as connected by the administrator.

There are five areas in Active Directory that need to be governed by a single master. Normally, operations affecting these areas are carried out by the first ever created domain controller. As these areas are critical to the functionality of Active Directory, mechanisms exist for other domain controllers to obtain authority over these areas. A domain controller responsible for such an area is called the *Flexible Single Master Operation (FSMO)* role owner. The FSMO roles in Windows 2000 are [61]:

- Only the server that owns the *Schema Master* role is allowed to extend the schema.

- The *Domain Naming Master* controls the Active Directory namespace. It can add or remove new sub-domains.
- To allow coexistence with legacy Windows NT servers, one Windows 2000 domain controller—the *PDC Advertiser*—needs to play the role of the Primary Domain Controller (PDC) in the Windows NT domain. It will synchronise Active Directory with the remaining Windows NT Backup Domain Controllers (BDC).
- Each security-enabled object in Active Directory has a unique security identifier (SID). These identifiers are assigned by the domain controller in which the entry is created. To ensure uniqueness, each server requests a pool of Ids from the *RID Master* in advance, which can later be used for newly created entries.
- The *Infrastructure Master* in a domain maintains references to objects in other domains.

The schema in Active Directory differs in some ways from that given in the X.500 Information Model. First, auxiliary classes do not exist as independent classes in Active Directory. Instead, they are incorporated into structural classes when the domain controller loads the schema. Their attributes become part of the structural class and they do not appear as values in the “objectClass” attribute. As a result, auxiliary classes cannot be searched for and, more importantly, they cannot be dynamically added to selected entries. Secondly, RDNs must be single-valued in Active Directory. Thirdly, Active Directory does not publish any matching rules in its schema although searching is obviously possible. Before attempting to change the schema one should be aware that extensions to Active Directory are irreversible. Once an object class or attribute type has been added, it cannot be removed but only marked as deactivated. [22]

Windows 2000 also includes a PKI component—the *Enterprise Certification Authority* (ECA). It can automatically issue certificates for users and computers and will publishes them in the directory. An ECA must be available for generating X.509 server certificates, if SSL protected LDAP connections are needed. [62]

## 8.2 Administration Tools

Administration of Windows 2000 and Active Directory is done with the Microsoft Management Console (MMC). MMC is an umbrella application that offers a consistent look and feel. Modules—so-called *SnapIns*— are used to handle specific tasks. For the management of Active Directory the following SnapIns are available: [61]

- *Users and Computers* to manage organizational units and accounts for computers and users.

- With *Sites and Services* the administrator can create sites and define which IP subnets should belong to them. The replication topology that is automatically created by the KCC can be manually tuned in here.
- Extensions to Active Directory's schema can be made with the *Schema SnapIn*.
- The *Domains and Trust SnapIn* is used to manually establish trust relationships between domains and to manage meta-information of the domain itself.
- *ADSI Edit* is a low-level tool for Active Directory. Unlike the other SnapIns it does not offer dialogues that present attributes in context, but rather provides a complete listing of attributes and their values for a given entry, thereby giving access to attributes that are otherwise hidden.
- The *Group Policy SnapIn* is used to manage *Group Policy Objects* (GPOs).

Windows 2000 uses GPOs to apply settings to multiple objects. A section exists, which deals with the settings for computers, and another section deals with user settings. GPOs can be assigned to sites, domains and organizational units. Typically GPOs are applied during boot-up for computers or log-on for users but can also be periodically be re-applied if a refresh policy is in place. In a homogeneous Windows 2000 environment virtually all settings of the client PCs can be controlled with GPOs, here, for example, are a few of the important ones:

- Scripts that should be run during user logon / logoff or computer boot-up respectively.
- Software that should be available on computers/to users can be assigned, and will be installed during the next application of the GPO.
- Settings in the *Registry*, Windows' database for application configuration data.
- Security settings like password, audit or public key policies. A GPO can for example be used to install a new X.509 trusted root certificate on all systems.
- Services on computers that should be started or must remain quiescent.

## 8.3 Directory-enabled Applications

The DNS server included in Windows 2000 can be integrated with Active Directory.<sup>4</sup> If this feature is activated, all DNS records would be stored as entries in the directory.

---

<sup>4</sup>It is possible to use an existing name server with support for DNS SRV records, e.g. *bind8*. However, this requires the manual addition of the Windows 2000 specific records to the zone file. Another issue arises from the fact, that a Windows 2000 client will try to register itself with the DNS server using dynamic DNS [91] by default.

The SMTP server that comes with Windows 2000 is also directory-enabled. It can use the information stored in the directory to decide whether to forward incoming email.

Since its 2000 version, Microsoft's messaging solution *Exchange* requires Active Directory. While earlier versions of Exchange used a private directory service, which was accessible via LDAP, it is now fully integrated with Active Directory. In fact, Active Directory uses the database technology that was developed for Exchange. During its installation Exchange extends the Active Directory schema and adds user interface components to manage the new attributes. These components appear as additional property pages in the Users and Computers Snap-In of the MMC.

# Chapter 9

## LDAP-enabled user management for Linux

Due to their ever increasing size, management of computer networks and providing users with easy access to them has become a daunting task. Many of the problems inherent to large networks can be solved with a centralized user management. The following sections show which requirements would be posed to a centralized user management and how such a system was implemented using directory technology during the course of this thesis.

### 9.1 Requirements

A typical organization that runs a pool of Linux PCs would require the following services to make use of a central directory:

**R1.1** Basic operating system functions for user and group look-up.

**R1.2** User authentication for console logins, secure remote shells (SSH), email submission and retrieval (SMTP and IMAP), as well as for web pages access.

**R1.3** A white-pages service in the form of an organization wide address book.

**R1.4** An MTA to determine email routing information.

In the outlined network, the directory service is a mission critical application. It must therefore:

**R2.1** be highly available,

**R2.2** still allow for system maintenance of the workstations without network connectivity.<sup>1</sup>

The idea of **R1.2** is to provide the user with a unified login environment. His username and password will be the same for all accessed services. This requires some special precautions to be taken:

**R3.1** Passwords must not be send over the network in clear text.

**R3.2** A password policy, which defines password complexity and renewal requirements, should be enforced.

**R3.3** To further simplify the login process, authentication should be a Single Sign-On operation.

## 9.2 Implementation

### 9.2.1 Schema

User and group information were made available to the operating system by means of `nss_ldap` (see 6.3.1). [23] defines the schema items to use with this module:

- The auxiliary “`posixAccount`” object class includes attributes for the user’s unique identifier and number (“`uid`” and “`uidNumber`”), his primary group (“`gidNumber`”) as well as his “`homeDirectory`” and “`loginShell`”.
- The auxiliary “`shadowAccount`” object class adds attributes, which are used to manage account and password expiration.

This fulfills requirements **R1.1** as well as **R1.2** since `pam_ldap`, which is used to handle password verification (see 6.3.2), relies on the same schema items as `nss_ldap`.

[23] suggests the “`account`” object class as the structural class for a user’s entry. With respect to **R1.3**, “`inetOrgPerson`” has been chosen as structural object class instead. “`inetOrgPerson`” [86] is a descendant of “`organizationalPerson`” and “`person`”. It includes all the attributes that are required to build a state of the art white-pages service. Role accounts, which do not map to a real world person, would still created with “`account`” as structural object class.

---

<sup>1</sup>Support for user accounts in case of a network failure is not necessary, as under such circumstances the users’ home directories, which would be located on a central file server in the outlined environment, would also become unavailable. This will generally preclude any further work to be done on a Unix workstation.

Sendmail (see 6.3.3) was used as the MTA in this thesis. By default, it makes use of the schema items specified in [53]. i.e. for an incoming email, the MTA tries to find an entry that has a matching “mailLocalAddress” and then forwards the mail to the value stored in “mailRoutingAddress”. To avoid redundant storage of information in order to meet requirement **R1.4**, the “mail” attribute from “inetOrgPerson” was used instead of “mailLocalAddress”. The final recipient, which would normally be retrieved from “mailRoutingAddress”, would now be determined from the “userId” attribute.

### 9.2.2 Directory Service

For the directory service to be highly available (**R2.1**), directory servers must be run redundantly on more than one host. The data can be automatically synchronized between the different hosts by the servers’ replication mechanism. Clients can make use of such redundant server pool in a number of different ways:

- In *Round Robin DNS*, multiple IP addresses are assigned a to hostname. However, a failing LDAP server will require manual intervention since the DNS server does not check if the LDAP servers are actually accessible.
- Two servers can be configured as a *Cold Standby Cluster*<sup>2</sup>. The passive server regularly checks that the active server is still alive. If this *heartbeat* check fails, the passive server reconfigures itself with the IP addresses used by the active server (*IP Take-over*) and starts the services. As LDAP servers need to be running to allow for replication, this solution does not fit well for a directory service.
- Another possible scenario would be to set up *load balancing* with *Linux Virtual Server*<sup>3</sup>. A host called *director* would redirect incoming requests to a number of back-end servers based on a scheduling algorithm. This software would also regularly check the back-end servers’ availability and configure the redirection process accordingly. To avoid a single point of failure, a back-up for the director is required. This solution offers the best performance and fault-tolerance but would require at least four hosts.
- The first three options described here allow for redundancy that is implemented transparently to the clients. If one has access to all involved clients, another solution is possible. In most LDAP client libraries, the `ldap_init` function can take a list of hosts as argument. If one server is down, the SDK will automatically try to connect to the next one.<sup>4</sup>

---

<sup>2</sup><http://www.linux-ha.org/>

<sup>3</sup><http://www.linuxvirtualserver.org/>

<sup>4</sup>A connection attempt to a non-reachable host will generally timeout after 60 seconds. Such a delay will lead to unreasonably long response time. The current Netscape SDK therefore introduced a so-

In this thesis, the latter option was chosen to implement redundancy.

To fulfil requirement **R2.2**, system accounts are kept local and are not included in the directory. Both NSS and PAM should therefore be configured to first look at the local files (e.g. `/etc/passwd`) and to fall back to LDAP if no match is found. For example, PAM could be set up like this by stacking the `pam_pwdb` and the `pam_ldap` module.<sup>5</sup>

### 9.2.3 Authentication Considerations

To prevent passwords from being sent over the network in clear text, the connection between the client running `pam_ldap` and the directory server must be encrypted. In a similar vein, the replication traffic among directory server must also be protected. In both instances, SSL or TLS can be employed to guarantee privacy protection. However, all these precautions in the authentication back-end are useless if unprotected passwords are transmitted by front-end protocols, e.g. in Telnet or by using the IMAP login command without a security layer.

If password changes are only initiated by the Linux `passwd` command, the `pam_pw-check` module can be used to enforce a password policy. Some directory servers can also check the quality of a password themselves. If password changes can originate from other sources than `passwd`, a product offering such a feature (see 10.2) should be used.

### 9.2.4 Single Sign-On

LDAP is an access protocol to general-purpose directory services. While it can be conveniently used as an authentication backend, it was not designed as an authentication service and its application in this regard is sometimes seen as an abuse of the system. LDAP works well for networks, where plain text passwords need to be centrally verified (and managed), to allow unified logins. However, LDAP does not provide the necessary semantics to implement a Single Sign-On (SSO) system (requirement **R3.4**). In an SSO system the user logs in only once at the beginning of a session—usually by entering his username and password; for advanced security, a hardware token might be required additionally. During the initial logon process, the user obtains a set of credentials that are used in all further authentication events. These operations are handled

---

called *parallel connect* feature. The SDK tries to connect to all servers at the same time and abandons all outstanding attempts once a connection has been established.

<sup>5</sup>By default, the used SuSE Linux distribution is set up to use the `pam_unix` module. However, that module makes use of NSS, which in turn will access LDAP. This leads to undesired results. Instead, the recommended `pam_pwdb` module can be configured to only look at the local files.

transparently by the SSO framework so that the user can seamlessly move from one application to another.

The design presented here uses Kerberos 5 (see Section 4.1.2.4) as authentication system to provide an SSO environment. It is based on the design which can be found in Windows 2000. In Windows 2000, Microsoft has integrated the KDC with an LDAP server. This retains the possibility to verify a plain text passwords through an LDAP bind attempt while at the same time providing an advanced SSO environment.

In fact, the system that was implemented uses a Windows 2000 Server to provide the KDC functionality. A second server has been set up, so that the service is still accessible in case the first server becomes unavailable. Due to its redefinition of standard attributes from multi-valued to single-valued, Active Directory cannot fully support the schema specified in [23]. It was therefore decided to use Active Directory for authentication purposes only, and to keep OpenLDAP as repository for application specific data.

All programs that use PAM to verify plain text passwords with an LDAP server by means of `pam.ldap` can easily be migrated to Kerberos. `pam.krb5` is the respective PAM module that uses Kerberos as authentication back-end. This module verifies the user's password by requesting an initial Kerberos ticket. If this operation is successful, the ticket is stored in the user's credentials file for further use. To switch from LDAP to Kerberos password checking, the administrator only has to replace `pam.ldap` with `pam.krb5` in the authentication section of the PAM configuration files. The other sections remain unchanged. To enable a user to change his password, support for the Active Directory specific password change mechanism has been added to `pam.ldap` (see Appendix A.1).

To make use of the SSO functionality, applications need to support Kerberos—either directly or via GSSAPI. On the server side, the following implementations exist:

- Active Directory.
- OpenLDAP, Sendmail and Cyrus IMAPD. All these programs link with the Cyrus SASL library and can therefore make use of its GSSAPI plug-in.
- University of Washington IMAP server.
- SSH and OpenSSH. For OpenSSH to support Kerberos, an additional patch is required<sup>6</sup>.
- Concurrent Version System (CVS).
- Kerberos has also been added to the Telnet and FTP servers that are included in the MIT and Heimdal Kerberos distributions.

---

<sup>6</sup><http://www.sxw.org.uk/computing/patches/openssh.html>

All of the Unix applications listed above support Kerberos in their command line or console applications. However, support for Kerberos in GUI applications is not very widespread. *Mulberry*<sup>7</sup> is a notable exception. This IMAP client supports the SASL GSSAPI mechanism in IMAP and SMTP. Its LDAP client is still LDAPv2 and therefore only supports simple binds.

---

<sup>7</sup><http://www.cyrusoft.com/mulberry/>

# Chapter 10

## Conformance and Interoperability

### 10.1 Interoperability Tests

Unfortunately, conformance to a standard is no guarantee that two different implementations will inter-operate with each other. In practice, standards suffer from errata or wordings that can be interpreted in different ways.<sup>1</sup>

The *Open Group Directory Interoperability Forum*<sup>2</sup> holds the annual *DirConnect* conference where directory vendors come together to test the interoperability of their products. To provide some guidance for these tests, the *Basic LDAPv3 Interoperability Test Suite* (BLITS)<sup>3</sup> is being used. The Open Group also offers the *The Open Brand for LDAP 2000* certificate. This certificate is awarded to directory servers, which pass the VSLDAP [89] test suite.

Another implementation of BLITS is available from Public Works and Government Services of Canada<sup>4</sup>. This *Test Suite Utility* (TSU) implements a subset of BLITS version 2.3 and additionally, the EuroSInet interoperability test for LDAPv2 [11] and X.500 (93) [10] are supported.

A subset of BLITS has also been used in compiling the overview presented in the next chapter.

---

<sup>1</sup>The IETF takes a very practical approach to quality assurance of its standards. A standard can only be “promoted” from Proposed to Draft Standard, if two interoperable implementations exist, which were developed independantly.

<sup>2</sup><http://www.opengroup.org/directory/>

<sup>3</sup> The first version of BLITS was edited by Chris Apple, AT&T. Revisions were made for the *Connectathon* interoperability events. Later versions, the latest being 2.5 Draft 1 [90], were made by the Open Group.

<sup>4</sup><http://cagc.srv.gc.ca/>

## 10.2 Supported features in DSAs

The tables (Figure 10.1 and Figure 10.2) on the following two pages show which features the directory servers that have been introduced in chapter 6, 7 and 8 support.

## 10.3 Support for standard schema items

To evaluate how the different directory servers support the schema items defined for X.500 [44] and their application in LDAP [92], a migration of the AMBIX<sup>5</sup> data set was attempted. The data is currently mastered in a slapd from the University of Michigan LDAP 3.3 distribution, which only supports LDAPv2 and does only minimal verification of attribute syntaxes.<sup>6</sup> Therefore, the data had to be cleaned up first before it could be imported to an LDAPv3 compliant server. All entries and attributes that did not match the following schema items were removed to find a common ground. To structure the DIT, “country”, “locality”, “organization” and “organizationalUnit” as well as the auxiliary object class “labeledUriObject” were used. All entries referring to person have been modified to fit the “inetOrgPerson” object class [86]. Where non-ASCII characters appeared, the attribute value has been converted to the UTF-8 encoding as required by LDAPv3. Additionally, German Umlauts, which had been converted to the expanded representation, have been converted back to their canonical form for most non-ambiguous cases, e.g. from “Universitaet” to “Universität”.

An initial attempt to load this data into eDirectory resulted in a large number of errors. Organizational units in Germany tend to have names, which are quite long. However, [44, Annex C] specifies that values of the “ou” attribute may at most have 64 characters. The same problems exist for the “businessCategory” attribute, whose length restriction of 128 characters could also be easily violated. eDirectory and Active Directory strictly uphold these restrictions. A manual relaxation is not possible. SecureWay Directory limits “ou” to 128 characters.<sup>7</sup> To allow further testing, all organizations and organizational units that did not meet the length restrictions were removed along with any “person” entries stored below them. While such a procedure is acceptable for a test, the need to include these organizations would preclude the use of Active Directory or eDirectory as a directory server for AMBIX.

---

<sup>5</sup> “Aufnahme von Mail Benutzern in das X.500-Verzeichnis”, German for “Incorporation of Mail Users into the X.500 Directory”. AMBIX provides a white-pages service for the German academic community. <http://www.directory.dfn.de/ambix/> [14]

<sup>6</sup>Attributes for fax numbers did e.g. contain the organization’s name in addition to the number. Even more common was the use of ISO-Latin-1 characters to represent German Umlauts.

<sup>7</sup>According to an email from IBM support, it should be possible to extend the field lengths. This would be done by editing the schema files before creating the tables for the directory server in the DB2 back-end.

	Active Directory	eDirectory	Netscape Directory Server
Manufacturer	Microsoft	Novell	Netscape
Licensing	per user, per server	per user	per entry
Version	Windows 2000 SP1	8.5	4.12
Supported Platforms			
Linux	no	yes	yes
Solaris	no	yes	yes
HP-UX	no	no	yes
AIX	no	no	yes
Tru64	no	yes	yes
Netware	no	yes	no
Windows NT	no	yes	yes
Windows 2000	yes	yes	no
Supported Protocols			
LDAPv3	yes	yes	yes
X.500(93)	no	no	no
Supported Security Layers			
LDAPS	yes	yes	yes
StartTLS	no	no	no
SASL	yes	no	no
Supported Authentication Mechanisms			
Simple Bind	yes	yes	yes
CRAM-MD5	no	no	no
DIGEST-MD5	no	no	no
GSSAPI	yes	no	no
EXTERNAL	no	yes	yes
Password Policy	yes	yes	yes
Supported Controls			
Server-side sorting	yes	yes	yes
Paged results	yes	no	no
Virtual list view	no	yes	yes
Language Codes	no	no	yes
Extensible Matching	yes	no	yes
Dynamic Schema	yes	yes	yes
Dynamic ACLs	yes	yes	yes
Replication Architectures			
Single-master	no	yes	yes
Multi-master	yes	yes	no
LDIF im/export	yes	yes	yes
Management Tools	Windows	Java, Web	Java, Web
SNMP Monitoring	yes	no	yes
SDKs			
C	yes	yes	yes
Java	no	yes	yes

Figure 10.1: Directory Servers, part 1

	M-Vault	OpenLDAP	SecureWay Directory
Manufacturer	Messaging Direct	OpenLDAP Foundation	IBM
Licensing	per server	Artistic License	provided at no cost
Version	IC-R6.1v4	2.0.11	3.2a
Supported Platforms			
Linux	yes	yes	Technology Preview
Solaris	yes	yes	yes
HP-UX	no	yes	no
AIX	no	yes	yes
Tru64	no	yes	no
Netware	no	no	no
Windows NT	yes	experimental	yes
Windows 2000	no	experimental	no
Supported Protocols			
LDAPv3	yes	yes	yes
X.500(93)	yes	no	no
Supported Security Layers			
LDAPS	yes	yes	not in TP
StartTLS	yes	yes	no
SASL	no	yes	no
Supported Authentication Mechanisms			
Simple Bind	yes	yes	yes
CRAM-MD5	no	yes	yes
DIGEST-MD5	no	yes	no
GSSAPI	no	yes	not in TP
EXTERNAL	no	no	not in TP
Password Policy	no	no	no
Supported Controls			
Server-side sorting	no	no	no
Paged results	no	no	no
Virtual list view	no	no	no
Language Codes	no	yes	no
Extensible Matching	no	no	no
Dynamic Schema	no	no	yes
Dynamic ACLs	yes	no	yes
Replication Architectures			
Single-master	yes	yes	yes
Multi-master	no	experimental	no
LDIF im/export	yes	yes	yes
Management Tools	TCL	no	Java, Web
SNMP Monitoring	yes	no	no
SDKs			
C	yes	yes	yes
Java	no	no	yes

Figure 10.2: Directory Servers, part 2

Additionally, the following directory server specific problems were encountered:

- Active Directory

In Active Directory many of the standard attributes were redefined to be single-valued instead of multi-valued. This included “commonName”, “givenName”, “telephoneNumber” and “mail”, common attributes that often do have multiple values. In addition, if a multi-valued attribute is used to name an entry, the value used in the RDN has to appear before any other value of the same attribute.

Active Directory does not include the “inetOrgPerson” object class by default. Instead, the “contact” object class is intended to be used. However, a “contact” entry was not allowed below an “organization” or “organizationalUnit” entry. The containment rules for these object classes were therefore changed. Furthermore, “locality” was added to the possible superiors of “organization”.

As Active Directory only supports auxiliary object classes by binding them to a structural object class. The “labeledUri” attribute was therefore added to the “locality”, “o”, “ou” and “contact” attributes. “businessCategory”, which is an attribute of “inetOrgPerson” was also added to “contact”.

- eDirectory

To allow “inetOrgPerson” entries to be added below a “locality” entry, the schema had to be modified. Novell offers a downloadable file, which contains schema extensions and mapping changes to implement the “inetOrgPerson” object class separate from the NDS “user” object class.<sup>8</sup> This file also makes the required changes to the containment rules.

## 10.4 SASL GSSAPI

A special interest during this thesis has been taken in the interoperability of the SASL GSSAPI mechanism for Kerberos (GSS-KRB5). The following products with support for this feature were identified:

- Windows 2000. GSS-KRB5 is supported in Active Directory and the client library.
- Windows 2000 Kerberos interoperability tools for Unix [66]. A GSS-KRB5 library is provided in source code. It requires the source code release of the Netscape LDAP C SDK (Mozilla) and MIT Kerberos 1.1 to build.

---

<sup>8</sup><http://www.novell.com/products/nds/schema/index.html>

- OpenLDAP. GSS-KRB5 is supported in the directory server and in client library with the help of the Cyrus-SASL library and a Kerberos 5 implementation (MIT or Heimdal).
- SecureWay Directory. GSS-KRB5 is supported in the directory server and in client library (Windows platform only).
- JNDI. The GSS-KRB5 mechanism<sup>9</sup> is implemented using the JCSI cryptographic toolkit<sup>10</sup>.

Based on the GSS-KRB5 mechanism for JNDI, a SASL mechanism that works with Netscape Directory SDK for Java was implemented in the frame of this thesis.

Client	Server		
	Active Directory	OpenLDAP	SecureWay
Microsoft C API	ok	(4) (5)	(4) (7)
Microsoft Unix Tools	ok	not tested	not tested
OpenLDAP	(1) (2) (3) (8)	ok	(5) (7)
SecureWay	(7)	(5) (7)	ok
JNDI	(2) (6)	(3) (6)	(5) (7)
Netscape JDK	(2) (6)	(3) (6)	(5) (7)

Figure 10.3: LDAP GSSAPI Interoperability Matrix

Figure 10.3 shows the results from the interoperability tests conducted during this thesis. The following problems were identified:<sup>11</sup>

- (1) Active Directory returned an empty `serverSaslCreds`<sup>12</sup> string instead of omitting this optional element. This was unusual but was not a direct violation of the protocol. A patch for OpenLDAP was developed in the frame of this thesis and submitted for inclusion.<sup>13</sup>
- (2) Active Directory disregarded the client's maximum buffer size. This would become a problem, if security layers were negotiated and large result sets were returned. A workaround would to increase the client's maximum buffer size.

<sup>9</sup><http://security.dstc.edu.au/projects/java/java-sec/msg00290.html>

<sup>10</sup><http://security.dstc.edu.au/projects/java/jcsi.html>

<sup>11</sup>In LDAP, GSSAPI and Kerberos elements of protocol are specified in ASN.1. The SASL GSSAPI mechanism uses a plain binary encoding, and GSS-KRB5 uses mixture of plain and ASN.1 elements. This complicated the analysis of network traces, as recursive ASN.1 parsers could not be employed.

<sup>12</sup>SASL specific information that the server wishes to send to the client is carried in the `serverSaslCreds` field of an LDAP response.

<sup>13</sup><http://www.OpenLDAP.org/its/index.cgi/Software%20Bugs?id=884>

- (3) The Cyrus-SASL library employed by OpenLDAP had the maximum buffer size values hard coded to 0x0FFFFFF. A patch has been proposed to the Cyrus-SASL mailing list.<sup>14</sup>
- (4) By default, the Windows 2000 client library for LDAP uses the GSS-SPNEGO [2] SASL mechanism to negotiate the GSSAPI mechanism for use. However, it generated an incorrect encoding of the mechanism OID for GSS-KRB5. This problem can be circumvented by using `ldap_set_option` to set the preferred SASL mechanism to GSSAPI.
- (5) [93] specified that the SASL service name for LDAP is `ldap`, i.e. in lower-case. The respective Kerberos principal should therefore be `ldap/host.domain@REALM`. While Active Directory ignores case, the Windows 2000 client library as well as the SecureWay server and the client library create and accept only the upper-case form `LDAP/host.domain@REALM`.

By patching `LDAP_PLUGIN_IBM_GSSKRB.DLL` the SecureWay client could be modified to use the lower-case form. At an offset of 0x3F55 `LDAP@` had to be changed to `ldap@`.

- (6) Due to a bug in the Java environment for Windows<sup>15</sup>, no security layers were supported.
- (7) The GSSAPI security layer was not supported in SecureWay Directory. Connections were not encrypted.
- (8) Additionally to an external security layer provided by SSL or TLS, OpenLDAP tried to establish a SASL security layer. Active Directory claims to support this but actually does not. The connection therefore stalled after the security layer was negotiated. A workaround would have been to set the maximum SASL security strength factor to zero. (Use `-O maxssf=0` for the OpenLDAP command-line tools.)

---

<sup>14</sup><http://www.openldap.org/lists/openldap-devel/200101/msg00041.html>

<sup>15</sup><http://security.dstc.edu.au/projects/java/java-sec/msg00290.html>

# Chapter 11

## Performance

In this chapter the performance of the different directory servers presented in chapters 6, 7 and 8 will be evaluated and compared. For reliable operation of a directory service, it must be ensured, that the server can meet the expected requirements. As the needs of the various applications can be very different, they should be analysed and tested, before a general judgement of the suitability of a product for various applications can be given.

The DirectoryMark<sup>1</sup> benchmark by Mindcraft has been designed to evaluate the performance of directory servers. This benchmark can be tuned to test a specific combination of LDAP operations. It will therefore allow an exact simulation of the prevailing usage patterns in the planned service.

Mindcraft has identified three common scenarios for directory services: loading, messaging and address look-up. [70] An additional scenario was identified during the work on this thesis, i.e. the use of an LDAP server as an authentication back-end.

### 11.1 Test Environment

All tests were performed on an AMD Athlon 750 MHz with 256 MB of RAM. SuSE Linux 7.0 (Kernel 2.2.16) was used as Linux distribution. The Active Directory test was done under Windows 2000 Server (Service Pack 1, Build 5.00.2195). The client machine, on which the DirectoryMark 1.2 software was run, was an Intel PIII 500 with 128 MB of RAM running Windows 2000.

The DirectoryMark suite was used to actually perform the tests. It consisted of three components:

---

<sup>1</sup><http://www.mindcraft.com/directorymark/>

- `dbgen.pl` generated the sample entries on which the tests were subsequently run. It created “inetOrgPerson” entries which were arranged below one of four “organizationalUnit” entries. The data generated was output into an LDIF file by `dbgen.pl` which was then imported into the different directory servers (see Section 11.2.1). LDIF (*LDAP Data Interchange Format*, [15]) is a textual representation of a DIT. It is the standard format for exchanging directory data and is supported by all software packages in this test.
- `scriptgen.pl` created scripts that instructed the simulator what to do. The behaviour of this script could be modified by a number of parameters. Among other things it was possible to specify the number of threads and operations per thread to create, as well as the weighing factor of the different LDAP operations.
- `DirectoryMark.exe` simulated LDAP clients. It processed the scripts generated by `scriptgen.pl` and gave a statistic result as output.

For use in this thesis, the `dbgen.pl` utility was customized:

- The length of the generated passwords was reduced from 10 to 8 as Unix systems will only take the first 8 characters into account.
- Seven attributes, which were not included in the test data for Active Directory but are available in Active Directory’s schema, were now also generated for Active Directory.
- The “unicodePwd” attribute was used instead of “userPassword” in the test data for Active Directory. This allowed for setting a user’s password in Active Directory via LDAP.
- As distributed, the script chose a random name for the “manager” and “secretary” fields. This behaviour was altered so that these fields were now filled with DNs of previously generated entries.

The main reason for the last modification was that NDS does a referential integrity check on attributes that contain DNs, i.e. it will only allow such values that point to existing entries.<sup>2</sup>

Finally, support for the authentication scenario was added to `scriptgen.pl`.

## 11.2 Tests

For the tests performed in this analysis, 20,000 sample entries were generated. This number was chosen to reflect the user population at Tübingen University Computing Centre. (see Section 11.4)

---

<sup>2</sup>As a side effect, this allows for an organizational chart to be drawn from the test data. [101]

The messaging, address look-up, and authentication tests modeled a maximum load scenario. For a period of five minutes the test client sent request upon request to the server. In the address look-up and authentication scenario the client spawned four parallel threads to simulate concurrent access. The messaging test used only one thread. Each thread processed a script generated by `scriptgen.pl` with 10,000 operations. If a thread completed the execution of a script during the five minutes period, it restarted again.

Each test was consecutively run four times. The first run was intended to allow the server to populate its caches. Its results were therefore discarded. The numbers in the tables show the average of the remaining three runs.

### 11.2.1 Loading - Initial directory population

Before a test on a directory server could be performed, the test data had to be loaded first. Where available, vendor specific tools were been used to import the LDIF file generated by `dbgen.pl`. These “bulkload” tools assumed that the syntax of the data was correct and therefore would bypass any consistency checks. Some of the utilities could also write directly to the underlying database, in which case the LDAP server had to be stopped. Others used framing technology (e.g. eDirectory) to add multiple records at a time or allowed delayed writing to disk (e.g. Active Directory).

Directory Server	Tool	Time [m:ss]
Active Directory	ldapmodify (1)	33:43
eDirectory	ice (2)	255:53
M-Vault	dbulk	1:02
Netscape Directory Server	ldif2db	2:23
OpenLDAP	slapadd (3)	4:22
SecureWay Directory	ldapmodify (4)	66:55

Figure 11.1: Bulk-load times for 20,000 accounts

- (1) Microsoft’s bulk-load tool `ldifde` could not be used to import passwords. Therefore OpenLDAP’s `ldapmodify` with an SSL connection and GSSAPI authentication was used.
- (2) On import eDirectory generated a public/private key pair for each user, which was very CPU intensive. The `ice` utility had an option (“-1”) to turn this off. However, this lead to intermittent failures (“unknown error”), so this option could not be used.

- (3) With OpenLDAP, the best performance was achieved in a two step process. Entries were first added through `slapadd` without any indices defined (0:44). Index settings were then enabled in the config file and the indices generated with `slapindex` (3:38). Data was imported with `schema check` and `dbsync` disabled since the data was correct and the database did not need to be flushed after each operation.
- (4) Use of the provided tools (`bulkload`, `ldif2db`) only resulted in an error: “Error code -1 from odbc string: 'SQLAllocEnv'“. OpenLDAP’s `ldapmodify` was used instead.

### 11.2.2 Messaging

Directory Server	Operations/s	Average Time (ms)	Max Time (ms)
Active Directory	637	1	24
eDirectory	204	4	76
M-Vault	89	38	2967
Netscape Directory Server	606	1	269
OpenLDAP	270	3	208
SecureWay Directory	48	20	148

Figure 11.2: Results for the messaging scenario

The messaging scenario simulated a mail transfer agent (MTA) that queried a directory for mail routing information. After initialising itself the MTA opened a connection, which stayed open while the MTA was running. For each incoming email, it then issued a query for the local part of the email address. The result then indicated to which host the mail should be forwarded and if the “to” field should be changed.

### 11.2.3 Address Look-up

In the address setup the directory servers acted as an email look-up service for an organization. Users would use the directory to search for email addresses. The search pattern was a combination of substring and exact filters on given name, surname, common name and userid. After every fifth search, the connection was closed and reopened.

Directory Server	Operations/s	Average Time (ms)	Max Time (ms)
Active Directory	582	4	254
eDirectory	142	27	2826
M-Vault	99	9	151
Netscape Directory Server	356	8	2883
OpenLDAP	128	25	10203
SecureWay Directory	26	150	29695

Figure 11.3: Results for the address look-up scenario

### 11.2.4 Authentication

This test was setup to simulate an LDAP authentication back-end. A client first opened a connection to the server and searched for the username, i.e. an exact subtree search on the “uid” attribute. If the user existed in the directory, it would then try to bind as the user and finally retrieve the information contained in the user’s entry for further use, i.e. perform a base search on the user’s DN.

Due to a limitation in `DirectoryMark.exe`, the operations had to be reordered. The bind operation was now executed first. As `DirectoryMark.exe` only counted the number of search requests, the figures reported had to be halved, in order to get a correct estimate for the maximum capacity of the authentication back-end.

Directory Server	Operations/s	Average Time (ms)	Max Time (ms)
Active Directory	123	2	136
eDirectory	20	90	449
M-Vault	45	32	372
Netscape Directory Server	162	5	258
OpenLDAP	151	4	497
SecureWay Directory	(20) <sup>3</sup>	(61)	(1107)

Figure 11.4: Results for the authentication scenario

---

<sup>3</sup>The SecureWay Directory Server repeatedly died without any error messages during this test. The numbers included here are from one fully completed run. This fault could not be reproduced, if the number of concurrent clients was reduced to one.

## 11.3 Results

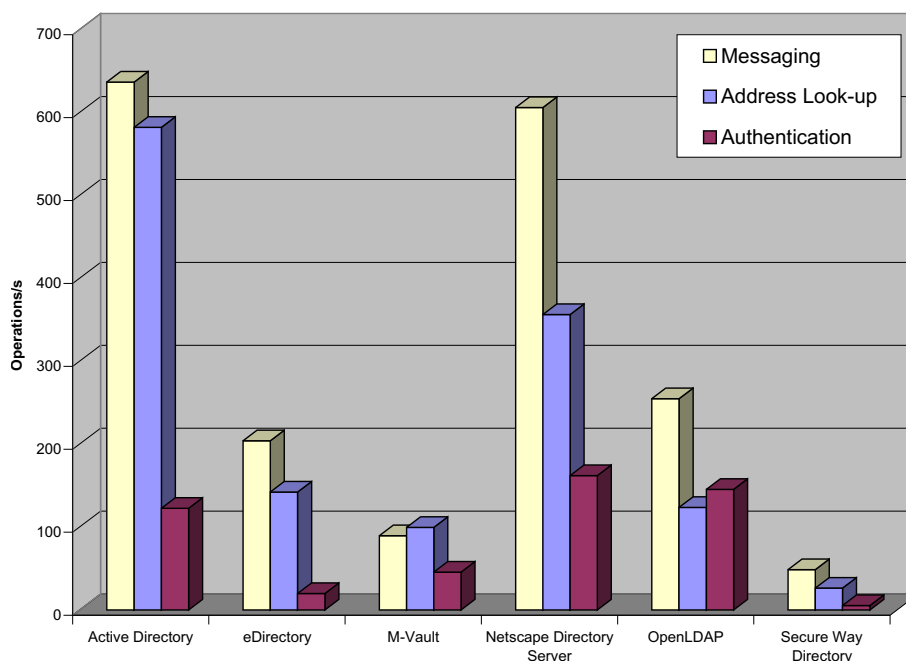


Figure 11.5: Results overview

Figure 11.5 shows the results from the query tests. Overall, Active Directory and the Netscape Directory Server stand out with regard to performance. If it was not for the bad result in the authentication test, eDirectory would be in joint third place along with OpenLDAP. Although M-Vault offers better authentication performance than eDirectory, it falls into fifth place overall.

Additional to its last place in the performance comparison, the Linux “Technology Preview” version of SecureWay Directory suffers from stability problems. The directory server silently died in the authentication tests with multiple concurrent clients. These errors will hopefully be fixed in the production release.

## 11.4 Comparison with real-life requirements

The Computing Centre at Tübingen University handles email and computer accounts for students and academic staff and has a user population of about 19,000.

As displayed in Figure 11.6, the daily number of emails handled by the MTA goes up to 70,000. Historically the peak value has been 220,000 emails on one day. This was

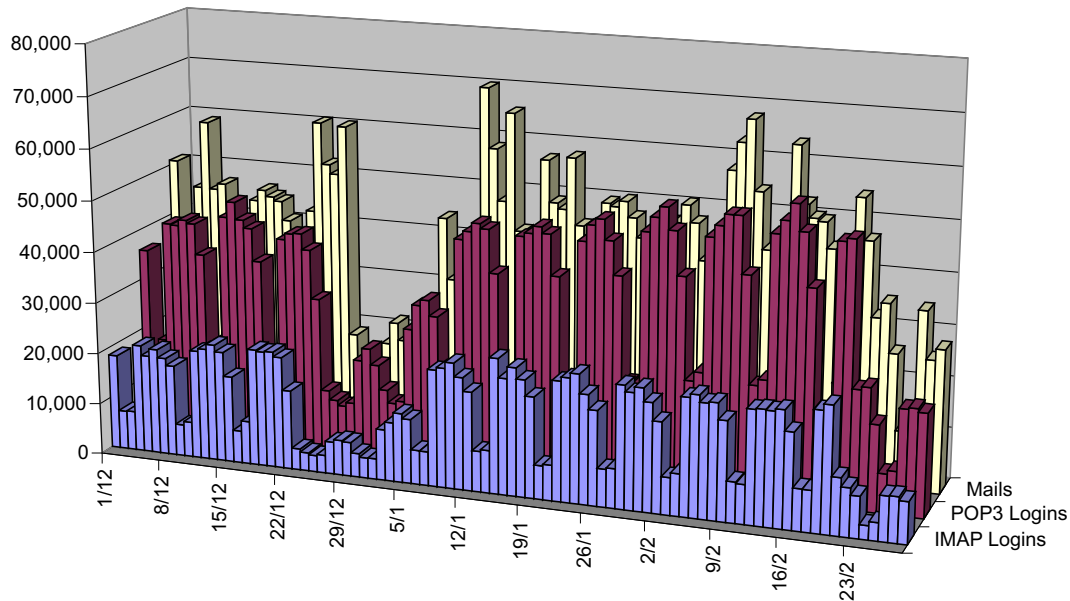


Figure 11.6: Daily amount of emails and login attempts at the Computing Centre

caused by a misconfigured client, which provoked an email loop. These figures represent the total numbers of emails handled by the MTA. However, a directory look-up will only be performed for emails addressed to local recipients. When compared with the performance numbers, it is obvious that leading products can easily handle this load. It takes Active Directory and the Netscape Directory Server under 10 minutes to process a full day's peak load. OpenLDAP and eDirectory would handle this load in about a quarter of an hour.

The number of client logins to retrieve emails is quite constant over the respective three month period. A clear distinction is visible between the load on work days and weekends. There are about 25,000 IMAP and 55,000 POP3 logins on an average day, yielding a total of 80,000 authentication attempts per day. As in the messaging scenario, 80,000 login attempts do not pose any substantial requirements on the host running the directory server.

Combining the numbers for email routing and authentication, a directory server would have to handle an average number of 150,000 search requests a day. On the other hand, there are only 30 to 40 changes on an average day. Since the modification rate is less than 0.001 of the total number of operations, no tests dealing with operations, which change the data in the directory, were performed.

## Chapter 12

# BIB<sub>T</sub>E<sub>X</sub> records in LDAP – Building a non-standard white-pages service.

Pierangelo Masarati had the idea of storing bibliographical references in a directory. His LDAP2BibTeX package<sup>1</sup> provides a schema and two utility programs, one for converting an existing collection of bibliographical references in BibTeX bib format to LDIF, which can be loaded into a directory server, and another for retrieving the information from the directory again.

### 12.1 Schema and DIT

The schema designed by Pierangelo Masarati uses one object class (“bibtexEntry”) to store a reference. Entries of this type must have a “bibtexEntryTag” attribute that holds the string used in the LaTeX `\cite{}` command and a “bibtexEntryType” attribute that describes the kind of publication, e.g. book, manual or article. All other attributes are optional.

The approach in this thesis was however to build a two-level object class hierarchy. The abstract “bibtexEntry” class is a collection of attributes common to all BibTeX resource types. It includes the mandatory “cn” (short for “common name”) attribute. Its value is used as parameter to the LaTeX `\cite{}` command. “cn” also is the naming attribute for “bibtexEntry”. Structural object classes exist for every BibTeX resource type. These classes are derived from “bibtexEntry”. Attributes that are required for a resource type, e.g. the publisher of a book, are declared as mandatory in these object classes.

---

<sup>1</sup><http://mbdyn.aero.polimi.it/~masarati/ldap2bibtex.html>

With the modified schema, application can take advantage of the information contained in the schema without having to be adapted specifically to BibTeX. For example, a dialog for creating a new reference could first enumerate the available resource types and would then only display fields for those attributes that make sense for the given type.

Another guideline in designing the schema was to store information in human-readable and standards-compliant form, and still keep the semantics of TeX. This makes information accessible with standard tools while avoiding any information loss. TeX-specials can generally be converted to respective Unicode characters. However, no Unicode representation exists for mathematical formulas, which sometimes appear in titles. The “author” attribute is also an area of concern. First, BibTeX uses curly brackets to get name prefixes right. Curly brackets are also used mark those words in titles, whose case must be preserved. Secondly all authors of a document appear on one line separated by and. When stored in the directory, the author field should be a multi-valued attribute. This allows for better search capabilities. However, LDAP does not guarantee the order of values in a multi-valued attribute. To cope with these problems, all values that have special TeX code in them are additionally stored in an attribute subtype that is identified by the `;lang-x-tex` tag<sup>2</sup>. The `;lang-x-tex` form should be used by BibTeX and also for editing purposes. If information is displayed by non-BibTeX-aware applications, the base form is used instead.

For example, the entry for [30] would look like this in LDIF notation:

```
dn: cn=Howes:1999:UDL, cn=Bibliography
cn: Howes:1999:UDL
objectclass: top
objectclass: bibtexEntry
objectclass: bibtexBook
bibtexAuthor: T. Howes
bibtexAuthor: M. Smith
bibtexAuthor: G. Good
bibtexAuthor;lang-x-tex: T. Howes and M. Smith and
    G. Good
bibtexTitle: Understanding and Deploying LDAP
    Directory Services
bibtexTitle;lang-x-tex: Understanding and Deploying
    {LDAP} Directory Services
bibtexYear: 1999
bibtexPublisher: Macmillan Technical Publishing
```

---

<sup>2</sup>Language subtypes in LDAP have been specified in [95]. This standard allows for private extensions in the form of `lang-x-*`.

## 12.2 Converting bibliographies to LDIF

The original package includes `bibtex2ldap`, a tool to convert `bib` files to LDIF. It is written as a `lex` and `yacc` parser. This program was extended to be more tolerant to the `bib` syntax. It will also migrate only those attributes that have been defined in the schema.

Separate tools have been developed for two special bibliographical collections:

- `rfc-parse2ldif.pl`<sup>3</sup> processes the index file<sup>4</sup> for IETF Request for Comments. RFCs are stored as “`bibtexTechReport`”. Where applicable, these entries are augmented by the “`rfcStatus`” auxiliary object class. This allows information about the status of an RFC (e.g. current category, earlier or later revisions) to be stored within its entry.
- `id-parse2ldif.pl` does the same for Internet Drafts. In addition to bibliographic information, the source<sup>5</sup> contains abstracts, which are also migrated.

## 12.3 Retrieving BibTeX records from LDAP

When compiling a `tex` source file, LaTeX writes meta-information like section names for the table of contents or referenced citations into an `aux` file. The provided `l2b.pl` utility scans this file, optionally recursing into included files, and builds a hash<sup>6</sup> of all references. A connection is then opened to an LDAP server and a search for each entry in the hash is performed. The results from these queries are converted to `bib` format and printed to `stdout`.

## 12.4 A web front-end with Java Servlets

A Java servlet was developed with the help of the Netscape Directory SDK for Java to allow efficient management of bibliographical references in the directory. It makes active use of schema information and stores its resource strings in the directory. To this end, an “`ldapServletSchemaItem`” auxiliary object class with two structural subclasses (“`ldapServletObjectClass`” and “`ldapServletAttributeType`”) were introduced. Both include a “`displayName`” attribute, which is used to store a friendly name for a schema

---

<sup>3</sup>Based on a script by the [RFC.net](http://RFC.net) web site.

<sup>4</sup><ftp://ftp.isi.edu/in-notes/rfc-index.txt>

<sup>5</sup><ftp://ftp.isi.edu/internet-drafts/lid-abstracts.txt>

<sup>6</sup>A Perl hash is used as data-structure instead of an array because this way duplicate citations will be included only once.

item, for example “Book” for “bibtexBook”. Support for additional languages can thus be added easily by adding an attribute value with the appropriate `lang` subtype. To use this information in the servlet, two new classes (`BibtexAttributeSchema` and `BibtexObjectClassSchema`) were derived from the respective classes for schema items in the SDK. These classes are able to read the additional information from an entry in the directory and provide methods to return it to the servlet.

A pool of persistent connections to the directory server is used to maximise performance. To serve an http request, a connection from the pool is requested. This avoids the overhead of having to establish a new LDAP connection for each request. By using the integrated session management of the servlet container, authenticated LDAP connections are used for tasks that involve modifying entries in the directory.

## 12.5 Conclusions and future work

This example shows how a directory service tailored for a specific application can be designed. A more general approach for storing bibliographic information might look into using the standards proposed by the Dublin Core meta-data initiative<sup>7</sup> instead of a BibTeX specific schema. In addition, implementing support for the format proposed in [55] is worth being considered.

---

<sup>7</sup><http://dublincore.org/>

## Chapter 13

# Conclusions and Future Work

Directory services are designed and developed for managing different aspects of modern computer networks. In this thesis, the use of directory services as central authentication backend and as central repository for provisioning email applications has been presented and analysed with regard to its resource requirements. The benchmarks that have been conducted in the frame of this thesis show that current directory servers running on commonly available hardware can easily keep up with the performance requirements of a typical academic institution (see Chapter 11).

As the performance requirements are met by a number of products, other criteria become more important in the decision making process for a particular product. If costs are a primary concern, products that do not incur any license costs are likely to be chosen. This would include OpenLDAP and—when released—SecureWay Directory. Organisations, which own hardware from Sun Microsystems, could consider the Netscape Directory Server instead. The free binary license of Solaris 8 for the SPARC platform contains a 200,000-entry production license. If the intended use of the directory goes beyond user management, flexibility and full compliance to the standards becomes an issue. While Active Directory in its current version does a good job in fulfilling its primary task as NOS specific directory, it is not as capable and compliant as the standalone directory servers. As a result, a migration of the AMBIX directory, which offers a white-pages service for the German academic community, to Active Directory as provided in Windows 2000 is not feasible (see Chapter 10).

Many German universities provide heterogeneous Windows/Linux computer pools to their students. Networks that consist of client PCs running Windows are generally connected to either a Windows NT/2000 or a NetWare server. In such an environment, integrating the Linux client PCs with the native Windows—pending an upgrade to Windows 2000, if Windows NT is still used—or NetWare directory services seems feasible <sup>1</sup>. This allows for investments, which have been put into the infrastructure in

---

<sup>1</sup>Windows 2000 Professional systems can also be configured to use a non-Windows 2000 Server

the past, to be retained and reused for new applications.

If the Linux clients in a heterogeneous Windows/Linux network should not to access Active Directory directly, but rather authenticate to a Linux based directory server, passwords need to be synchronised to provide users with a unified login. A prototype software that accomplishes this task has been developed in the frame of this thesis. It is comprised of two modules—one for each direction of synchronisation (see Appendix A). A dynamic link library, which implements the Windows password change notify API, handles the Windows to Linux side. Password changes originating on Linux can be forwarded to an Active Directory by means of a modified PAM module. Additionally, the password verification mechanism in OpenLDAP has been extended to support the hash function used by a Windows server. This allows for bootstrapping a Linux directory from a user base held in a Windows server (see Appendix B).

Directories are not limited to user white-pages and authentication services though. The technology offers the required flexibility for being used as information store in other areas as well. As an example, a schema for storing bibliographical references has been developed. Based on the information model specified therein, applications for provisioning the BibTeX program from the LaTeX typesetting suite and for managing the references by means of an HTTP-gateway have been written (see Chapter 12).

Future work might concentrate on adapting `nss.ldap` to Active Directory. `nss.ldap` is a name service module, which allows the use of LDAP instead of a local `/etc/passwd` file. This would allow Linux clients to retrieve user information besides passwords directly from Active Directory and would eliminate the need to maintain a second directory server on Linux. The reverse approach, i.e. integrating a Unix Kerberos distribution with OpenLDAP, would also offer interesting possibilities.

Additionally, research could be carried out into the area of certificate based authentication mechanisms. As part of this work, the SASL EXTERNAL mechanism could be implemented in OpenLDAP. To make effective use of this mechanism, ways to associate a certificate with a directory entry need to be specified and implemented—including a check of the certificate's validity. One way of establishing this relationship would be to map the subject's name from the certificate to the distinguished name of an entry in the directory. Another possibility would be to store data that uniquely identifies a certificate in the associated entry. As part of this analysis, the role of the directory service as part of a Public Key Infrastructure could to be investigated.

---

KDC. However, this will only provide authentication services. Advanced management options as they can be implemented with GPOs in a native Windows 2000 network will not be available.

# Appendix A

## Password Synchronization

### A.1 Changes originating in Linux

In addition to the authentication and authorisation functions, PAM (see Section 6.3.2) offers an interface for changing passwords. However, its implementation in `pam_ldap` was not capable of changing passwords in Active Directory. To change passwords in Active Directory via LDAP, the following requirements have to be fulfilled:

1. The LDAP connection has to be made over SSL with an encryption algorithm that supports at least 128-bit session keys. [67]
2. a) The old and new password must be provided –or–  
2. b) The new password is provided and the operation is performed by someone who has the “Reset Password” right for the entry.
3. Passwords have to be provided in the syntax of the “unicodePwd” attribute. To construct a “unicodePwd” value from a Unix password, the password is surrounded by double quoting marks (") and converted to a two-byte Unicode (UCS-2) representation in network byte order (little endian). The conversion to Unicode is straightforward for ASCII characters, as the ordinal numbers for ISO-8859-1, of which ASCII is a subset, are equal to the ones for Unicode.

To implement this scheme in `pam_ldap` two patches have been written in the frame of this thesis. The first one allows LDAP over SSL (LDAPS) connections using the OpenLDAP 2.0 SDK<sup>1</sup>. The second patch implements the conversion routine for “unicodePwd” described above. Both patches have been integrated into `pam_ldap` in version 86.

---

<sup>1</sup>To make LDAPS connections work in the OpenLDAP SDK, an additional patch had been written to fix a bug in OpenLDAP. <http://www.OpenLDAP.org/its/index.cgi/Software%20Bugs?id=889>

## A.2 Changes originating in Windows 2000

In Windows 2000 *Password Filters* offer a way of implementing password policies and change notifications. [65] To synchronize password changes with Linux, a password filter DLL (`LDAPSYNC.DLL`) that implements a change notification for an arbitrary LDAP server was written using Microsoft's C LDAP SDK (`WLDAP32.LIB`).

To initialise the DLL, the `InitializeChangeNotify` function is called. This routine reads the configuration parameters from the registry and opens an SSL connection to the specified LDAP server. It also initialises error reporting with the Event Log service and builds a `printf` format string (`dnFormatString`) that will be used to construct the DN of the entry whose password is to be changed. Since only the new password is provided as parameter in the `PasswordChangeNotify` function, the DLL has to authenticate itself as someone who may reset a user's passwords. As the configuration parameters include the DN and password of an administrative user to the LDAP directory, the registry values must only be accessible to the Windows system and trusted users.

On a successful password change request, the Local Security Authority (LSA) calls the `PasswordChangeNotify` function. It first constructs the DN of the entry to be changed from the `UserName` parameter using the `dnFormatString`. Then an LDAP modify request is issued to update the password.

To build `LDAPSYNC.DLL`, the following components are required:

- `ldapsync.c`, the source code for the library.
- `messages.mc`, a resource file that specifies the DLL's error messages. This is required to make use of Windows' EventLog service.
- `ldapsync.rc`, this resource file specifies the version information and includes the binary resource files generated by the message compiler (`mc`) from the above file.
- `Makefile`, specifies how to link all parts to create the DLL.

## Appendix B

# LanManager Hashes in OpenLDAP

To allow synchronization of user accounts between an existing Active Directory and a newly set up OpenLDAP server, the data contained in the Windows 2000 server has to be migrated to the Unix host first. Most of the information stored in Microsoft's Active Directory can be retrieved via LDAP. There is, however, one notable exception: user passwords or cryptographic hashes thereof. This information is only available through the Security Accounts Manager (SAM), a protected subsystem of Windows 2000 (and NT). A migration would thus require that new passwords are issued to all users. Fortunately a program called "pwdump"<sup>1</sup> is available that can read the data stored in the SAM and export it to a text file. This file is in the format of SAMBA's `/etc/smbpasswd` file and contains a line for each account specifying account name, UID, LanManager and NT password hashes. To make use of these values in OpenLDAP a verification function for LanManager password hashes is needed.

OpenLDAP supports an extensible scheme to verify plain text passwords provided in an LDAP simple bind. The verification is implemented in the `passwd.c` module of `liblutil`. The server calls `lutil_passwd` to verify a password. This function takes three parameters:

- `passwd`, the stored password,
- `cred`, the credentials provided in simple bind,
- `schemes`, an array of allowed schemes.

The value with which the provided password is compared is stored in the directory using the "userPassword" attribute. While this attribute was originally intended to hold only clear text passwords<sup>2</sup>, its usage has been extended to store values derived

---

<sup>1</sup><http://www.webspan.net/~tas/pwdump2/>

<sup>2</sup>[92, Section 5.36]

from the actual passwords.<sup>3</sup>

The syntax of the userPassword attribute is:

$$\{scheme\}hashvalue$$

Where *scheme* is the name of the hash algorithm used, and *hashvalue* the result of the application of the *scheme* algorithm to the password. OpenLDAP 2.0 supports crypt, MD5 and SHA (including salted versions). But support is not limited to cryptographic hash functions that store their secrets in the directory. Password verifiers can also make use of external sources. The following external schemes exist:

- “unix” uses the systems `/etc/passwd` file,
- “sas1” calls the Cyrus SASL library to verify a password,
- “kerberos” tries to get an initial ticket from a KDC.

To actually check a password, `lutil_passwd` iterates over the `pw_schemes` array that holds the names of the available schemes and pointers to the corresponding verification functions.

To support LanManager password hashes, a “lanman” scheme has been developed in the frame of this thesis. The `hash_lanman` function computes the hash according to [109, Appendix A.2 - A.4] which is then compared to the stored value in `chk_lanman`. A migration script in Perl that reads the output of `pwdump` and generates an LDIF file for import into the directory server has also been provided. These contributions have been submitted to the OpenLDAP project and have been incorporated to the head development branch.<sup>4</sup>

---

<sup>3</sup>Since this is in conflict with the core LDAP RFCs, a proposal has been made to use an “authPassword” attribute instead. [107]

<sup>4</sup><http://www.openldap.org/its/index.cgi/Contrib?id=899>

# Appendix C

## Abbreviations

**ACL** Access Control List

**API** Application Programming Interface

**ASN.1** Abstract Syntax Notation One

**COM** Common Object Model

**DAP** Directory Access Protocol (X.500)

**DIB** Directory Information Base (X.500)

**DISP** Directory Information Shadowing Protocol (X.500)

**DIT** Directory Information Tree (X.500)

**DLL** Dynamic Link Library

**DNS** Domain Name Service

**DN** Distinguished Name (X.500)

**DSA** Directory System Agent (X.500)

**DSE** DSA Specific Entry

**DSML** Directory Services Markup Language

**DSP** Directory System Protocol (X.500)

**DUA** Directory User Agent (X.500)

**GSSAPI** Generic Security Service API

**IETF** Internet Engineering Task Force

**IMAP** Internet Message Access Protocol

**IP** Internet Protocol

- ISO** International Standards Organization (<http://www.iso.ch/>)
- ITU-T** International Telecommunication Union - Telecommunication Standardization Sector (<http://www.itu.int/ITU-T/>)
- KDC** Key Distribution Center (Kerberos)
- LDAP** Lightweight Directory Access Protocol
- LDIF** LDAP Data Interchange Format
- NDS** Novell Directory Services
- NIS** Network Information Service
- OSI** Open Systems Interconnection
- PDU** Protocol Data Unit
- PKCS** Public Key Cryptosystem
- PKI** Public Key Infrastructure
- POP3** Post Office Protocol, version 3
- RDN** Relative Distinguished Name (X.500)
- RFC** Request for Comment
- SASL** Simple Authentication and Security Layer
- SDK** Software Development Kit
- SMTP** Simple Message Transfer Protocol
- SNMP** Simple Network Management Protocol
- SQL** Standard Query Language
- SSL** Secure Socket Layer
- SSO** Single Sign-On
- TCP/IP** Transmission Control Protocol / Internet Protocol
- TGT** Ticket Granting Ticket (Kerberos)
- TLS** Transport Layer Security

# Bibliography

- [1] A. Anantha, D. Boreham, J. Sermersheim, and M. Armijo. [LDAP Extensions for Scrolling View Browsing of Search Results](#). Internet Draft, April 2000. 32
- [2] E. Baize and D. Pinkas. [The Simple and Protected GSS-API Negotiation Mechanism](#). RFC 2478, IETF, December 1998. 59
- [3] Micky Balladelli. [Building a Very Large Active Directory](#). White paper, Compaq, February 2000. 42
- [4] J. Banning. *LDAP unter Linux. Netwerkinformation in Verzeichnisdiensten verwalten*. Addison-Wesley, 2001. 8
- [5] A. D. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder. Grapevine: An exercise in distributed computing. *Communications of the ACM*, 25(4):260–274, April 1982. 9
- [6] D. Chadwick. *Understanding X.500 - The Directory*. International Thomson Computer Press, 1996. 10
- [7] P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider. [Architecture of the WHOIS++ service](#). RFC 1835, IETF, August 1995. 16
- [8] T. Dierks and C. Allen. [The TLS Protocol Version 1.0](#). RFC 2246, IETF, January 1999. 23
- [9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976. 18
- [10] EuroSInet. [EuroSInet x.500 \(93\) Interoperability Tests](#), March 1997. 53
- [11] EuroSInet. [EuroSInet LDAP v2 Interoperability Tests](#), March 1997. 53
- [12] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. [HTTP Authentication: Basic and Digest Access Authentication](#). RFC 2617, IETF, June 1999. 37

- [13] A. Freier, P. Karlton, and P. Kocher. [The SSL Protocol Version 3.0](#), November 1996. 23
- [14] P. Gietz, R. Schneider, and K. Spanier. [X.500 für alle](#). *DFN Mitteilungen*, (42):23–24, November 1996. 54
- [15] G. Good. [The LDAP Data Interchange Format \(LDIF\) - Technical Specification](#). RFC 2849, IETF, June 2000. 27, 61
- [16] Bruce Greenblatt. *Internet Directories: How to Build and Manage Applications for LDAP, DNS, and Other Directories*. P T R Prentice-Hall, 2000. 9, 16
- [17] A. Gulbrandsen, P. Vixie, and L. Esibov. [A DNS RR for specifying the location of services \(DNS SRV\)](#). RFC 2782, IETF, February 2000. 31
- [18] K. Harrenstien, M.K. Stahl, and E.J. Feinler. [NICNAME/WHOIS](#). RFC 0954, IETF, October 1985. 16
- [19] K. Hockenbury. [The Ambitious Amateur vs. crypt\(3\)](#). <http://attila.stevens-tech.edu/~hockenb/crypt3.html>, 1997. 20
- [20] J. Hodges and R. Morgan. [Lightweight Directory Access Protocol \(v3\): Technical Specification](#). Internet Draft, January 2001. 15
- [21] J. Hodges, R. Morgan, and M. Wahl. [Lightweight Directory Access Protocol \(v3\): Extension for Transport Layer Security](#). RFC 2830, IETF, May 2000. 15, 23
- [22] D. House. Understanding and extending the Windows 2000 schema. Programming paper, IBM Global Services, September 1999. 44
- [23] L. Howard. [An Approach for Using LDAP as a Network Information Service](#). RFC 2307, IETF, March 1998. 35, 48, 51
- [24] T. Howes. [The String Representation of LDAP Search Filters](#). RFC 2254, IETF, December 1997. 15
- [25] T. Howes, S. Kille, W. Yeong, and C. Robbins. [The X.500 String Representation of Standard Attribute Syntaxes](#). RFC 1488, IETF, July 1993. 12
- [26] T. Howes, S. Kille, W. Yeong, and C. Robbins. [The String Representation of Standard Attribute Syntaxes](#). RFC 1778, IETF, March 1995. 12
- [27] T. Howes and M. Smith. [The LDAP Application Program Interface](#). RFC 1823, IETF, August 1995. 25, 28

- [28] T. Howes and M. Smith. [The LDAP URL Format](#). RFC 2255, IETF, December 1997. 15
- [29] T. Howes, M. Smith, and B. Beecher. [DIXIE Protocol Specification](#). RFC 1249, IETF, August 1991. 12
- [30] T. Howes, M. Smith, and G. Good. [Understanding and Deploying LDAP Directory Services](#). Macmillan Technical Publishing, 1999. 3, 6, 11, 12, 13, 68
- [31] T. Howes, M. Smith, M. Wahl, A. Herron, and A. Anantha. [The C LDAP Application Program Interface](#). Internet Draft (work in progress), November 2000. 25
- [32] T. Howes, M. Smith, R. Weltman, C. Tomlinson, J. Sermersheim, M. Kekic, and S. Sonntag. [The Java LDAP Application Program Interface](#). Internet Draft, February 2001. 27
- [33] T. Howes, M. Wahl, and A. Anantha. [LDAP Control Extension for Server Side Sorting of Search Results](#). RFC 2891, IETF, August 2000. 32
- [34] iPlanet. [Netscape Directory Server Plug-In Programmer's Guide](#), September 1998. 32
- [35] iPlanet. [The History behind LDAP](#), 2001. 32
- [36] ITU-T. [Open Systems Interconnection – Specification of Abstract Syntax Notation One \(ASN.1\)](#). Recommendation X.208, International Telecommunications Union, Geneva, 1988. 11
- [37] ITU-T. [Message Handling System and Service Overview](#). Recommendation X.400, International Telecommunications Union, Geneva, 1993. 10
- [38] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and service](#). Recommendation X.500, International Telecommunications Union, Geneva, 1993. 1, 4, 10
- [39] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Models](#). Recommendation X.501, International Telecommunications Union, Geneva, 1993. 10
- [40] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Authentication framework](#). Recommendation X.509, International Telecommunications Union, Geneva, 1993. 10, 22

- [41] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Abstract service definition](#). Recommendation X.511, International Telecommunications Union, Geneva, 1993. 10
- [42] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation](#). Recommendation X.518, International Telecommunications Union, Geneva, 1993. 10
- [43] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Protocol specifications](#). Recommendation X.519, International Telecommunications Union, Geneva, 1993. 10
- [44] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Selected attribute types](#). Recommendation X.520, International Telecommunications Union, Geneva, 1993. 10, 54
- [45] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Selected object classes](#). Recommendation X.521, International Telecommunications Union, Geneva, 1993. 10
- [46] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Replication](#). Recommendation X.525, International Telecommunications Union, Geneva, 1993. 10
- [47] ITU-T. [Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory](#). Recommendation X.530, International Telecommunications Union, Geneva, 1997. 10
- [48] ITU-T. [Security architecture for Open Systems Interconnection for CCITT](#). Recommendation X.800, International Telecommunications Union, Geneva, 1991. 17
- [49] S. Kille. [A String Representation of Distinguished Names](#). RFC 1779, IETF, March 1995. 12
- [50] S. Kille, M. Wahl, A. Grimstad, R. Huber, and S. Sataluri. [Using Domains in LDAP/X.500 Distinguished Names](#). RFC 2247, IETF, January 1998. 5, 42
- [51] J. Klensin, R. Catoe, and P. Krumviede. [IMAP/POP AUTHorize Extension for Simple Challenge/Response](#). RFC 2195, IETF, September 1997. 21
- [52] J. Kohl and C. Neuman. [The Kerberos Network Authentication Service \(V5\)](#). RFC 1510, IETF, September 1993. 22

- [53] H. Lachman and G. Shapiro. [LDAP Schema for Intranet Mail Routing](#). Internet Draft, January 2001. 36, 49
- [54] D. Larisch. [Verzeichnisdienste im Netzwerk – NDS, Active Directory und andere](#). Hanser, April 2000. 38
- [55] R. Lasher and D. Cohen. [A Format for Bibliographic Records](#). RFC 1807, IETF, June 1995. 70
- [56] P. Leach and C. Newman. [Using Digest Authentication as a SASL Mechanism](#). RFC 2831, IETF, May 2000. 21, 24
- [57] R. Lee and S. Seligman. [JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications](#). The Java Series. Addison-Wesley, May 2000. 27
- [58] H. Levanen, B. Freund, and H. Mansi. [Using LDAP for Directory Integration – A Look at IBM SecureWay Directory, Active Directory and Domino](#). Number SG24-6163-00 in IBM Redbooks. IBM Corporation, International Technical Support Organization, December 2000. 22
- [59] J. Linn. [The Kerberos Version 5 GSS-API Mechanism](#). RFC 1964, IETF, June 1996. 22
- [60] J. Linn. [Generic Security Service Application Program Interface Version 2, Update 1](#). RFC 2743, IETF, January 2000. 22
- [61] A. G. Lowe-Norris. [Windows 2000 Active Directory](#). O’Reilly, January 2000. 42, 43, 44
- [62] Microsoft. [How To Enable Secure Socket Layer \(SSL\) Communication Over LDAP For Windows 2000 Domain Controllers](#). Knowledge Base Q247078, December 1999. 44
- [63] Microsoft. [Windows 2000 Kerberos Authentication](#). Windows 2000 Technical Library, 1999. 22
- [64] Microsoft. [Active Directory Service Interfaces](#). Platform SDK, 2000. 28
- [65] Microsoft. [Password Filters](#). Platform SDK, 2000. 74
- [66] Microsoft. [Interoperability with Microsoft Windows 2000 Active Directory and Kerberos Services](#). MSDN Library, February 2000. 57
- [67] Microsoft. [Description of Password-Change Protocols in Windows 2000](#). Knowledge Base Q264480, October 2000. 73

- [68] Microsoft. [Understanding the Role of Directory Services versus Relational Databases](#). Windows 2000 Technical Resources, February 2001. 1
- [69] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. Project Athena Technical Plan, Section E.2.1, MIT Laboratory for Computer Science, Cambridge, MA, December 1987. 21
- [70] Mindcraft. [DirectoryMark 1.2 Run Rules](#), 2000. 60
- [71] J. Myers. [Simple Authentication and Security Layer \(SASL\)](#). RFC 2222, IETF, October 1997. 23, 24
- [72] National Institute of Standards and Technology (NIST). [Secure Hash Standard](#). Federal Information Processing Standards Publication (FIPS PUB 180-1), April 1995. 18, 20
- [73] National Institute of Standards and Technology (NIST). [Data Encryption Standard \(DES\)](#). Federal Information Processing Standards Publication (FIPS PUB 46-3), October 1999. 20
- [74] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978. 21
- [75] Netscape. [Netscape LDAP SDK for Java 4.0 Programmer's Guide](#), 1999. 27
- [76] C. Newman. [Anonymous SASL Mechanism](#). RFC 2245, IETF, November 1997. 24
- [77] C. Newman. [Using TLS with IMAP, POP3 and ACAP](#). RFC 2595, IETF, June 1999. 24
- [78] Novell. [NDS Technical Overview](#), November 1999. 38
- [79] Novell. [DENIM Overview for ISV's](#), 2000. 39
- [80] Novell. [DirXML Administration Guide](#), November 2000. 40
- [81] R. Rivest. [The MD5 Message-Digest Algorithm](#). RFC 1321, IETF, April 1992. 20
- [82] M.T. Rose. [Directory Assistance Service](#). RFC 1202, IETF, February 1991. 12
- [83] V. Samar and R. Schemers. [Unified Login with Pluggable Authentication Modules \(PAM\)](#). RFC 86.0, Open Software Foundation, October 1995. 36

- [84] B. Schneier. *Applied cryptography. Protocols, Algorithms, and Source Code in C*. Wiley, 1994. 20
- [85] O. Schönherr. *Konzeption und Implementierung eines Java-Backends für einen LDAP-Server*. Master's thesis, Fachhochschule für Technik und Wirtschaft Berlin, July 1999. 31
- [86] M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. RFC 2798, IETF, April 2000. 48, 54
- [87] Sun Microsystems. *Netscape LDAP SDK for C 4.1 Programmer's Guide*, November 2000. 25
- [88] Sun Microsystems. *JNDI Service Providers*, 2000. 28
- [89] The Open Group. *LDAP 2000 Test Suite - VSLDAP*, November 2000. 53
- [90] The Open Group. *Basic LDAPv3 Interoperability Test Suite (BLITS)*, February 2001. Issue 2.5, Draft 1. 53
- [91] P. Vixie, Ed., S. Thomson, Y. Rekhter, and J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC 2136, IETF, April 1997. 45
- [92] M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. RFC 2256, IETF, December 1997. 15, 54, 75
- [93] M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan. *Authentication Methods for LDAP*. RFC 2829, IETF, May 2000. 15, 22, 24, 59
- [94] M. Wahl, A. Coulbeck, T. Howes, and S. Kille. *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*. RFC 2252, IETF, December 1997. 15
- [95] M. Wahl and T. Howes. *Use of Language Codes in LDAP*. RFC 2596, IETF, May 1999. 68
- [96] M. Wahl, T. Howes, and S. Kille. *Lightweight Directory Access Protocol (v3)*. RFC 2251, IETF, December 1997. 15
- [97] M. Wahl, S. Kille, and T. Howes. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. RFC 2253, IETF, December 1997. 15
- [98] A. Waugh. *X.500 and the 1993 Standard*. Technical Report TR-SA-94-03, Division of Information Technology, CSIRO Australia, March 1994. 11

- [99] C. Weider, A. Herron, A. Anantha, and T. Howes. [LDAP Control Extension for Simple Paged Results Manipulation](#). RFC 2696, IETF, September 1999. 32
- [100] T. Weitzel, S. Son, F. v. Westarp, P. Buxmann, and W. König. [Wirtschaftlichkeitsanalyse von Kommunikationsstandards – eine Fallstudie am Beispiel von X.500 Directory Services mit der Siemens AG](#). Arbeitsbericht 00-05, SFB 403, Johann Wolfgang Goethe-Universität Frankfurt, 2000. 2
- [101] Rob Weltman and Tony Dahbura. [LDAP Programming with Java](#). Addison-Wesley, 2000. 27, 61
- [102] M. Wilcox. [Implementing LDAP](#). Wrox, 1999. 25
- [103] S. Williamson, M. Kusters, D. Blacka, J. Singh, and K. Zeilstra. [Referral Whois \(RWhois\) Protocol V1.5](#). RFC 2167, IETF, June 1997. 16
- [104] W. Yeong, T. Howes, and S. Kille. [X.500 Lightweight Directory Access Protocol](#). RFC 1487, IETF, July 1993. 12
- [105] W. Yeong, T. Howes, and S. Kille. [Lightweight Directory Access Protocol](#). RFC 1777, IETF, March 1995. 12
- [106] K. Zeilenga. [LDAP Password Modify Extended Operation](#). RFC 3062, IETF, February 2001. 36
- [107] K. Zeilenga. [LDAP Authentication Password Schema](#). RFC 3112, IETF, May 2001. 76
- [108] D. Zimmerman. [The Finger User Information Protocol](#). RFC 1288, IETF, December 1991. 16
- [109] G. Zorn and S. Cobb. [Microsoft PPP CHAP Extensions](#). RFC 2433, IETF, October 1998. 76