LDAP Implementation HOWTO

Roel van Meer

Linvision BV

r.vanmeer@linvision.com

Giuseppe Lo Biondo

INFN MI

giuseppe.lobiondo@mi.infn.it

v0.5, 2001−03−30

Revision History

Revision 0.5	2001−03−30	Revised by: rvm
Cleanup, fixes, overview rewritten.		
Revision 0.4	2001−02−01	Revised by: rvm
Added dns section.		
Revision 0.3	2001−01−18	Revised by: rvm
Added MTA sections.		
Revision 0.2	2000−11−12	Revised by: glb
Improved section on nss. Added sections about certificates and wrappers.		

This document describes the technical aspects of storing application data in an ldap server. It focuses on the configuration of various applications to make them ldap−aware. Some applications that assist in handling ldap data are also discussed.

Table of Contents

Table of Contents

1. Overview

1.1. Why this howto?

I started learning about ldap when my company felt the need for a centralized storage of user account information, and wanted to use ldap for this. I soon found that there were bits and pieces of documantation everywhere, but that there was no document that put it all together. This has been the reason to start it.

Furthermore, Ldap is becoming more widely used every day. I think it is useful that when people are considering to use Ldap, they can get a full overview of which applications are Ldap aware. This might help them to choose their system setup carefully, without throwing everything about every time they want to change something or add functionality.

It started out as a project roadmap on how we wanted to implement Ldap for our own uses. But thanks to my employer, Linvision, who gave me the opportunity to do some research on things that weren't really useful to our own cause, it changed from a roadmap to a technical overview of applications that are ldap aware.

1.2. What is it about?

Most of the common services can be authenticated through PAM, Pluggable Authentication Modules. With the pam_ldap and nss_ldap modules, all pamified programs can get their information from LDAP. More information about PAM in general can be found on the Linux−PAM site. Information about pam_ldap and nss_ldap can be found on the padl software site.

For Samba, things are a little difficult at this moment. The current stable Samba versions do not have Ldap support. Ldap support can be found in the HEAD and TNG branch, and probably also in the combined tree. The problem is that samba has it's own usernames and passwords. It does have usage for PAM, in fact, but that is not sufficient to do all the authentication and retrieval of user information. Because the implementation of LDAP in samba is not fully finished yet, there are a few limitations to the use of ldap with samba. From my experiences, the HEAD is at this time (early June 2000) not stable enough, and the performance is unsatisfying. However, when the ldap support is fully functional in the new releases, samba too can be configured to get all of it's user information from ldap.

Another thing that can be stored into an ldap database is DNS. When the amount of machines connected to your network increases, it is no longer feasable to edit the DNS files by hand. When machine accounts are stored into ldap, two simple DNS entries (one for the lookup, and one for the reverse lookup) can easily be added at the same time. This too provides a simplification of system management. Although the storage of DNS entries in an ldap database may not be neccesary for most systems, it may prove useful to some people.

Since sendmail version 8.9 (see sendmail.net for more details), sendmail has Ldap support. Postfix and QMail are ldap−aware too. When setting up an email system which has multiple mailhosts and or fallback hosts, it is convenient to store all the information in one place. Normally, every system needs to be configured separately, with the same information. When using ldap, this can be avoided.

Roaming access can also be used with LDAP. Netscape versions 4.5 and up have the possibility to store user data like bookmarks and such via an HTML or LDAP server. This gives users their good old preferences, wherever they log in and use Netscape.

Microsoft's office programs can import address books. Thay can also use an Active Directory service to automagically match emailaddresses to user names or nicknames. With Ldap this can be done on a Linux system, without the need for Microsoft Exchange Server or something the like.

1.3. What is it NOT about?

First thing: I will try not to talk too much about the actual setup and administration of Ldap itself. There is an excellent Ldap HOWTO available at the Linux Documentation Project that discusses this.

Secondly, I will not discuss things regarding the applications itself, when they have nothing to do with Ldap.

Lastly, in most cases, I cannot tell you if it is wise to use Ldap. I don't have that kind of experience. I can tell you how to do it, if you want, but i cannot tell you if you should. There is plenty documentaion available that discusses the useability of Ldap in general.

1.4. Acknowledgements

At first, I would like to thank my employer, Linvision, for giving me the opportunity to work on this document in their time.

Furthermore, I would like to thank the following people, who have contributed to this document in some way (in no particular order): Giuseppe Lo Biondo.

1.5. Disclaimer

This document is provided as is and should be considered as a work in progress. Several sections are as yet unfinished, and probably a lot of things that should be in here, aren't. I would greatly appreciate any comments on this document, of whatever nature they may be.

In any case, think before you go messing around with your system and don't come to me if it breaks.

1.6. Copyright and license

2. LDAP authentication using pam_ldap and nss_ldap

This section focuses on how to use LDAP as a NIS substitute for user accounts management. Having a lot of user accounts on several hosts often causes misalignments in the accounts configuration. LDAP can be used to build a centralized authentication system thus avoiding data replication and increasing data consistency.

At the moment the most used method to distribute users account data and other information through a network is the Network Information Service (NIS). Like LDAP, NIS is a distributed service that allows to have a central server where configuration files such as passwd, shadow, groups, services, hosts etc. are kept. The NIS server is queried by NIS clients to retrieve this information.

LDAP can offer the same functionality of NIS, moreover there are several advantages on using LDAP:

- Information on the LDAP server can be easily used for several purposes. As outlined in this HOWTO, the same users entries on the LDAP database can be used for other applications like phone directories, mail routing, staff databases etc., thus avoiding data replication and inconsistency.
- LDAP allows complex access control lists to be applied on the database. This allows for a fine grain tuning of permissions on the database entries.
- A secure transmission channel between the LDAP server and the clients can be implemented through the Secure Socket Layer (SSL).
- A fault tolerant service can be implemented using slapd replication [1] and DNS round robin queries (this is not covered in this document).
- Having a single instance of users on the network helps to maintain users on many hosts from a single management point (i.e. you can create and delete accounts in the LDAP server and this changes are available immediately to LDAP clients).

Herein I'll focus on how an LDAP server can be used for authentication and authorization on systems providing the Pluggable Authentication Module (PAM) and the Name Service Switch (NSS) technologies, in particular I'll refer to the Linux operating system even if this instructions can be applied to other operating systems.

The environment proposed consists of an LDAP server where users account data is stored in a convenient format and a set of Un*x clients using this information to authenticate and authorize users on resources in a standard Un*x fashion.

A secure channel is also required in client/server communications since critical information such as user account data, should not be sent in clear over the network, this channel will be provided by the Secure Socket Layer.

On the client side a caching mechanism, needed for performance issues, can be provided by the Name Service Caching Daemon.

All (almost) the software used to build this system is Open Source.

2.1. The components of the framework

This section outlines the various components that are used to build the authentication system. For each component is given a brief description.

2.1.1. Authentication: PAM and pam_ldap.so

The Pluggable Authentication Module allows integration of various authentication technologies such as standard UNIX, RSA, DCE, LDAP etc. into system services such as login, passwd, rlogin, su, ftp, ssh etc. without changing any of these services.

First implemented by Sun Solaris, PAM is now the standard authentication framework of many Linux distributions, including RedHat and Debian. It provides an API through which authentication requests are mapped into technology specific actions (implemented in the so called pam modules). This mapping is done by PAM configuration files, in which, for each service are basically given the authentication mechanisms to use.

In our case, the pam_ldap module, implemented in the shared library pam_ldap.so, allows user and group authentication using an LDAP service.

Each service that needs an authentication facility, can be configured through the PAM configuration files to use different authentication methods. This means that it is possible, using the PAM configuration files, to write a custom list of requirements that an user must satisfy to obtain access to a resource.

2.1.2. The Name Service Switch and nss_ldap.so

Once an user is authenticated, many applications still need access to user information. This information is traditionally contained in text files (`/etc/passwd`, `/etc/shadow`, and `/etc/group`) but can also be provided by other name services.

As a new name service (such as LDAP) is introduced it can be implemented either in the C library (as it was for NIS and DNS) or in the application that wants to use the new nameservice.

Anyway, this can be avoided using a common, general purpose, name service API and by demanding to a set of libraries the task of retrieving this information performing technology based operations.

This solution was adopted in the GNU C Library that implements the *Name Service Switch*, a method originated from the Sun C library that permits to obtain information from various name services through a common API.

NSS uses a common API and a configuration file (`/etc/nsswitch.conf`) in which the name service providers for every supported database are specified.

The databases currently supported by NSS [2] are:

- aliases: Mail aliases.
- ethers: Ethernet numbers.

- group: Groups of users.
- hosts: Host names and numbers.
- netgroup: Network wide list of host and users.
- network: Network names and numbers.
- protocols: Network protocols.
- passwd: User passwords.
- rpc: Remote procedure call names and numbers.
- services: Network services.
- shadow: Shadow user passwords.

Using the nss_ldap shared library it is possible to implement the maps above using LDAP, anyway here I'll focus only on the LDAP implementation of shadow, passwd and group database tough all the maps above can be implemented. For most of the other maps it is even unadvisable to store them in ldap, as they tend not to change too often, so it is not a problem to have them locally as files, and storing them in ldap would cause some minor performance loss.

2.1.3. The Lightweight Directory Access Protocol

For our application LDAP is used to provide clients with information about user accounts and user groups. The standard objectclasses that are used to represent users and groups are: top, posixAccount, shadowAccount and posixGroup.

Users entries on the database must belong at least[3] to the top, posixAccount and shadowAccount objectclasses. Group entries must belong to the top and posixGroup objectclasses.

The implementation of pam_ldap and nss_ldap that we use refers to this objectclasses, that are described in RFC 2307.

Note: Actually LDAP NSS recognize other objectclasses

2.1.4. The Name Service Caching Daemon

The Name Service Caching Daemon (NSCD) is used to cache name service lookups and can improve performance with the services provided by the NSS.

It must be tuned with a large cache for passwd entries in order to have acceptable performance on the client side.

It has some disadvantages however, like the introduction of cache inconsistencies, so you would want to be sure you need this before you use it. We have succesfully running some systems without it, and personally i think that it isn't really neccesary on relatively small systems.

2.1.5. The Secure Socket Layer

For details on SSL refer to Section 10.

SSL is needed in the communication between the LDAP server and the clients libraries (pam_ldap.so and nss_ldap.so), since sensible data, such as password entries, needs to be encrypted between the client and the server. SLL also permits the client to uniquely identify the server, thus avoiding to obtain authentication informations from an untrusted source.

Client authentication (the server identifies the client) is not supported in the current implementation of pam_ldap and nss_ldap modules tough it may be useful.

2.2. Building the authentication system

This section describes the steps needed to build the authentication system using the components described in the previous section.

Figure 1. PAM Layout

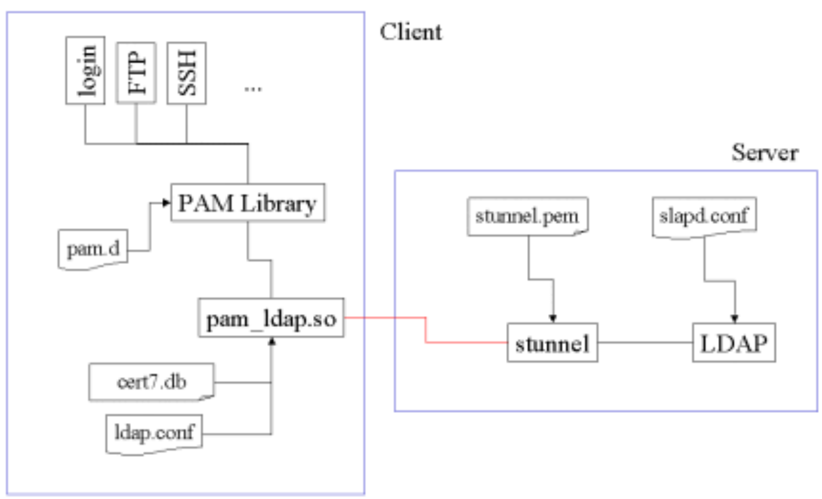
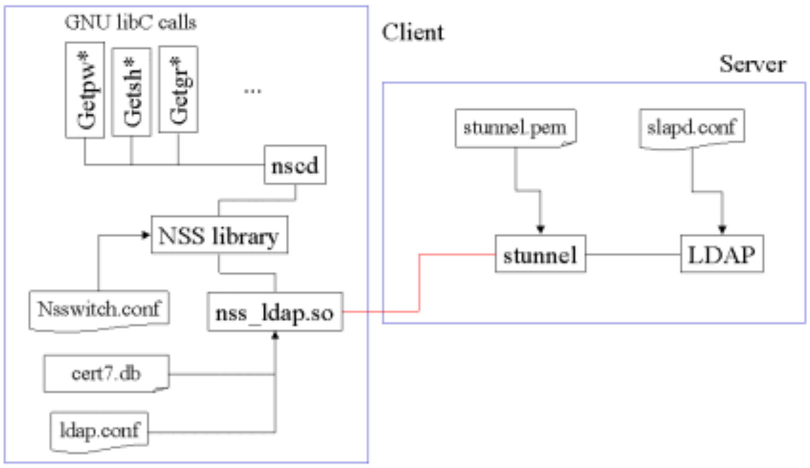


Figure 2. NSS Layout



Though this layout may seem quite complex to implement, most of the components are already in place in a Linux system.

2.2.1. Server side

On the server side an LDAP server must be installed and configured. The LDAP server used is OpenLDAP, an open source LDAP toolkit including an LDAP server (slapd), library and utilities.

At the moment OpenLDAP comes with two implementation of LDAP: a V2 implementation (OpenLDAP 1.2.x) ad a V3 (OpenLDAP 2.0.x) implementation

The V3 implementation provides native SSL, the V2 doesn't. Anyway it is possible to use an SSL wrapper to add SSL capabilities to the server (see Section 10).

2.2.1.1. Installing and configuring OpenLDAP

You can refer to the LDAP−HOWTO for instruction on installation and configuration of LDAP

Once slapd is properly configured we need to insert some data for the initial creation of the database. Therefore an LDIF (LDAP Data interchange format) file must be created. This is a text file that can be imported in the LDAP database with the command:

```
#ldif2ldbm −i your_file.ldif
```

> **Note:** ldif2ldbm is provided with the OpenLDAP 1.2.x package, if you use OpenLDAP 2.0.x, you should use the `ldapadd` command (after the server is started).

If you use OpenLDAP 2.0.x (LDAPv3) you can find the standard nis schema in the file `etc/openldap/schema/nis.schema`, include it in your `slapd.conf` with the include directive, to have schema enforcement.

Here is an example of a minimal LDIF file. Each entry is separated by a blank line.

```
dn:dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit

dn:ou=groups, dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups

dn:ou=people, dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit
ou: people

dn: cn=Giuseppe LoBiondo, ou=people, dc=yourorg, dc=com
cn: Giuseppe Lo Biondo
sn: Lo Biondo
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
uid:giuseppe
userpassword:{crypt}$1$ss2ii(0$gbs*do&@=)eksd
uidnumber:104
gidnumber:100
gecos:Giuseppe Lo Biondo
loginShell:/bin/zsh
homeDirectory: /home/giuseppe
shadowLastChange:10877
shadowMin: 0
shadowMax: 999999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
shadowFlag: 0

dn: cn=mygroup, ou=groups, dc=yourorg, dc=com
objectclass: top
objectclass: posixGroup
cn: mygroup
gidnumber: 100
memberuid: giuseppe
memberuid: anotheruser
```

> **Note:** Note that lines that are too long are continued on the following line started by a tab or a space, this in true too for LDIF format files

Here we defined the base DN for the orgazation *dc=yourorg, dc=com* under which are contained two sub organizational units: people and groups. Then is described a user that belongs to the people organizational unit and a group (which the users belongs to) under the groups organizational unit.

> **Note:** Useful tools to convert existing databases into ldif format are provided by PADL and can be found at the address ftp://ftp.padl.com/pub/MigrationTools.tar.gz.

The LDIF file must be imported in the server while it is not running since the ldif2ldbm command builds the database directly, bypassing the LDAP server. Once the LDIF file is imported into the database, the server can be started.

2.2.2. Client side

On the client side pam_ldap.so and nss_ldap.so are required and they must be compiled using the Netscape LDAP Library (Mozilla) since it provides the required LDAPS (LDAP over SSL) API. The library is distributed in a binary package under Netscape One license and is not open source (it is public domain anyway).

The package can be extracted, for example, in the directory `/usr/local/ldapsdk`.

Client libraries must also have access to a certificate database containing the LDAP (stunnel) server certificate and the CA certificate of the CA that signed the server certificate (marked as trusted).

The certificate database must be in Netscape format since the Mozilla LDAP API used to compile pam_ldap and nss_ldap uses certificate databases in Netscape format.

To deal with such certificate databases it is convenient to use the certutil utility found in the PKCS#11 package provided by Netscape [4].

The main configuration file for LDAP clients is `/etc/ldap.conf`.

Note that if you use nss_ldap, you don't strictly need to use pam_ldap.

You can use the pam_unix_auth module instead, since nss_ldap maps all getpw* and getsh* calls into LDAP lookups and pam_unix_auth uses this calls to authenticate users.

2.2.2.1. PAM LDAP Installation and Configuration

To compile and install pam_ldap, do the following:

```
$ ./configure --with-ldap-lib=netscape4 \
              --with-ldap-dir=/usr/local/ldapsdk
$ make
# make install
```

The configure switch ––with–ldap–lib tells which LDAP library you are going to use.

The switch ––with–ldap–dir tells where you have installed your Netscape ldapsdk toolkit.

This will install `/lib/security/pam_ldap.so.1` and the various symlinks.

PAM has to be properly configured in order to access the new authentication system. PAM configuration files are located in the directory `/etc/pam.d` and are named after the service for which authentication is provided.

For example this is the PAM configuration file for the login service (in a file named `login`).

```
#%PAM-1.0
```

```
auth     required   /lib/security/pam_securetty.so
auth     required   /lib/security/pam_nologin.so
auth     sufficient /lib/security/pam_ldap.so
auth     required   /lib/security/pam_unix_auth.so use_first_pass
account  sufficient /lib/security/pam_ldap.so
account  required   /lib/security/pam_unix_acct.so
password required   /lib/security/pam_cracklib.so
password sufficient /lib/security/pam_ldap.so
password required   /lib/security/pam_unix_passwd.so use_first_pass md5 shadow
session  required   /lib/security/pam_unix_session.so
```

Standard PAM configuration files for use with PAM can be found in the pam_ldap source distribution, in the directory `pam_ldap-version/pam.d`.

This files can be copied in the `/etc/pam.d` directory. Caution must be given when performing this operation, since if something goes wrong you probably will not be able to login again. It is suggested to make a backup copy of `/etc/pam.d` before installing new files there and to leave an open privileged shell.

> **Note:** In the example `pam.d` directory, a `sshd` file is not present, so unless you create one, you will be unable to login via ssh, if it uses pam (OpenSSH does use PAM).

2.2.2.2. NSS LDAP installation and configuration

After you've unpacked the sources, check the makefile. For most configurations, it doesn't need to be edited. Anyway, if you want to use SSL you must link against an SSL aware LDAP library, such as the Netscape one.

Assuming that the ldap sdk is in `/usr/local/ldapsdk` you have to modify the Makefile to enable SSL. Look for NSFLAGS in `Makefile.linux.mozilla` and uncomment −DSSL.

Also check the LIBS definition to see if the ldapssl library specified in the file is the same that you have installed (ldap_nss.so compiles with both libldapssl40 and libldapssl30).

Then you can install the library:

```
$ make −f Makefile.linux.mozilla
# make −f Makefile.linux.mozilla install
#ldconfig
```

this installs `/lib/libnss_ldap.so`, which is the nss_ldap library, and a set of example configuration files, `/etc/nsswitch.ldap` and `/etc/ldap.conf`, in case they do not exist already.

Once you have installed it you must edit the NSS configuration file `/etc/nsswitch.conf`. Tough LDAP can be used for all the services we use it only for passwd, group and shadow therefore we should have something like:

```
passwd: files ldap
group:  files ldap
shadow: files ldap
```

in the first lines of the configuration file. With this configuration, entries are first looked in the system files and, if no value is returned, the LDAP server is queried.

Note: Beware when using ldap as backup for your dns lookups. If dns cannot resolve the hostname, we're in infinite recursion, because libldap calls gethostbyname(). [from the nsswitch.ldap]

2.2.2.3. NSCD configuration

NSCD is already available in many Linux distributions, anyway it can be found within the GNU C library package.

The NSCD configuration file is `/etc/nscd.conf`. Each line specifies either an attribute and a value, or an attribute, cachename, and a value. Fields are separated either by SPACE or TAB characters. cachename can be hosts, passwd, or groups (in our case we won't cache hosts).

```
enable-cache           passwd  yes
positive-time-to-live  passwd  600
negative-time-to-live  passwd  20
suggested-size         passwd  211
keep-hot-count         passwd  20
check-files            passwd  yes
enable-cache           group  yes
positive-time-to-live  group  3600
negative-time-to-live  group  60
suggested-size         group  211
keep-hot-count         group  20
check-files            group  yes
```

Keep in mind that the nscd program caches passwd entries obtained from LDAP.

This means that when an user is modified on the ldap server, the nscd cache remains valid. This is avoided when using flat unix files by the check−files directive that invalidates the cache when the corresponding file is modified. Such a mechanism should be generalized, at the moment anyway does not apply to LDAP. A way to avoid possible misalignments between the LDAP server and the cache is to invalidate the cache manually when updating passwd entries with the command:

```
#nscd --invalidate=TABLE
```

Where TABLE can be passwd, groups or hosts.

To avoid confusion when testing, do not use nscd.

Moreover using nss and nscd will produce a lot of open filedescriptors, so is easy to run out of available filedescriptors on the system (this can hang your system).

You can increase the maximum number of filedescriptors in a Linux box (Kernel 2.2.x) with something like:

```
#echo 16384 > /proc/sys/fs/file-max
```

The maximum number of filedescriptors suggested for a system depends anyway from the configuration of your system.

2.2.2.4. LDAP client configuration file

The LDAP client configuration file `/etc/ldap.conf` is read by pam_ldap and nss_ldap as well as other LDAP clients. The following is an example of how it should look like in our environment.

```
#
# $Id: section-pamnss.sgml,v 1.2 2001/03/26 16:57:07 rolek Exp $
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
# PADL Software
# http://www.padl.com
#
# If the host and base aren't here, then the DNS RR
# _ldap._tcp.[defaultdomain]. will be resolved. [defaultdomain]
# will be mapped to a distinguished name and the target host
# will be used as the server.
#
# Your LDAP server. Must be resolvable without using LDAP.
host 192.111.111.111
#
# The distinguished name of the search base.
base dc=yourorg, dc=com
#
# The LDAP version to use (defaults to 2,
# use 3 if you are using OpenLDAP 2.0.x or Netscape Directory Server)
# ldap_version 3
#
# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
# binddn cn=manager,dc=padl,dc=com
#
# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret
#
# The port.
# Optional: default is 389. 636 is for ldaps
port 636
#
# The search scope.
#scope sub
#scope one
#scope base
#
# The following options are specific to nss_ldap.
#
# The hashing algorithm your libc uses.
# Optional: default is des
#crypt md5
#crypt sha
#crypt des
#
# The following options are specific to pam_ldap.
#
# Filter to AND with uid=%s
pam_filter objectclass=posixAccount
#
# The user ID attribute (defaults to uid)
pam_login_attribute uid
#
# Search the root DSE for the password policy (works
```

```
# with Netscape Directory Server)
#pam_lookup_policy yes
#
# Group to enforce membership of
#
#pam_groupdn cn=PAM,ou=Groups,dc=padl,dc=com
#
# Group member attribute
pam_member_attribute memberuid
# Template login attribute, default template user
# (can be overridden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody
#
# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
pam_crypt local
#
# SSL Configuration
ssl yes
sslpath /usr/local/ssl/certs
#
```

> **Note:** To avoid problems with the various applications that may read this file it is suggested not to use tabs between parameters and values, only a single space.

The pam_groupdn directive is useful when an LDAP server provides authentication information to a pool of clients, but the user should be authorized only on a set of clients. This directive can provide the same functionality of NIS netgroups.

The SSL configuration directives are not documented in the package, but they tell to enable SSL and where the file containing the LDAP server certificate and the CA certificate is stored.

A Netscape certificate database named cert7.db is searched in sslpath. This file must contain the server certificate and the CA certificate (unless the server certificate is self signed). There are two ways to generate this file: using the Netscape PKCS#11 tools or using the Netscape browser.

With the Netscape browser, after you have started slapd and stunnel on the server you can use Netscape Navigator to connect to the URL https://your.ldap.server:636/, you will be prompted to insert the server certificate in your database. Also the CA certificate (provided by your CA) must be loaded in the database (unless you are using a self signed certificate). At this point you can copy the `$HOME/.netscape/cert7.db` in `sslpath`. It is preferred that you use a scratch account with a default `cert7.db` file since other server certificates, that may be present in your personal certificate database, will be considered by your LDAP client as trusted authentication servers. Once the browser has imported the server certificate it can be used to debug SSL since it will behave like the pam and nss libraries.

2.3. Starting up

On the server side you have to start slapd (the LDAP daemon process) with a command like:

```
# slapd
```

If you use stunnel, it has to be started on the LDAPS port 636:

```
# /usr/local/sbin/stunnel  -r ldap  -d 636 \
 -p /usr/local/ssl/certs/stunnel.pem
```

If you use OpenLDAP 2.0.x, compiled with TLS (OpenSSL), you can start the server using the command

```
# slapd -h "ldap:/// ldaps:///"
```

On the client nscd can be started with the a startup script, usually found in many Linux distributions:

```
# /etc/rc.d/init.d/nscd start
```

If PAM and NSS are correctly configured this should be enough.

2.4. Accounts maintenance

At this point account creation and maintenance should be done using LDAP client tools.

Unfortunately these general purpose tools are not intended for Un*x accounts maintenance. The one that seems to be enough versatile is the LDAP Browser/Editor (http://www−unix.mcs.anl.gov/~gawor/ldap) that allows to set passwords in various formats and can use SSL to connect to the server.

2.5. Known limits

As it is for NIS with a single master server (no slave servers), LDAP without a replication mechanism represents a single point of failure for the authentication system. For authentication purposes it is rather important to implement LDAP replication. The server that cames with OpenLDAP (slapd) provides replication capabilities.

2.6. File permissions

The following are the file permissions that should be applied to some of the files used by the authentication system.

```
-rw-r--r--  root.root /etc/ldap.conf
-rw-------  root.root /usr/local/etc/openldap/slapd.conf
-rwxr-xr-x  root.root /lib/security/pam_ldap.so.1
-rw-r--r--  root.root /lib/libnss_ldap-2.1.2.so
-rw-r--r--  root.root /usr/local/ssl/certs/cert7.db
-rw-------  root.root /usr/local/ssl/certs/stunnel.pem
```

3. Radius authentication using LDAP

A Radius Server, is a daemon for un*x operating systems which allows one to set up (guess what!) a radius protocol server, which is usually used for authentication and accounting of dial–up users. To use server, you also need a correctly setup client which will talk to it, usually a terminal server or a PC with appropriate which emulates it (PortSlave, radiusclient etc). [From the freeradius FAQ]

Radius has its own database of users, anyway, since this information is already contained in LDAP, it will be more convenient to use it!

There are several freeware Radius servers, the one that has good support for LDAP is the FreeRadius server (http://www.freeradius.org), it is still a development version, anyway the LDAP module works fine.

3.1. FreeRadius Radiusd configuration

Once you have installed the server you have to configure it using the configuration files, that are located under `/etc/raddb` (or `/usr/local/etc/raddb`)

In the `radiusd.conf` file edit :

```
[...omissis]
# Uncomment this if you want to use ldap (Auth-Type = LDAP)
# Also uncomment it in the authenticate{} block below
        ldap {
                server   = ldap.yourorg.com
                #login    = "cn=admin,o=My Org,c=US"
                #password = mypass
                basedn   = "ou=users,dc=yourorg,dc=com"
                filter   = "(posixAccount)(uid=%u))"
        }

[...omissis]

# Authentication types, Auth-Type = System and PAM for now.
authenticate {
        pam
        unix
#       sql
#       sql2
# Uncomment this if you want to use ldap (Auth-Type = LDAP)
        ldap
}
[...omissis]
```

Also edit the `dictionary` file:

```
[...omissis]
#
#       Non-Protocol Integer Translations
#

VALUE           Auth-Type               Local                   0
VALUE           Auth-Type               System                  1
VALUE           Auth-Type               SecurID                 2
```

```
VALUE           Auth-Type               Crypt-Local             3
VALUE           Auth-Type               Reject                  4
VALUE           Auth-Type               ActivCard               4
VALUE           Auth-Type               LDAP                    5
[...omissis]
```

And the `users` file to have a default authorization entry:

```
[...omissis]
DEFAULT         Auth-Type := LDAP
                Fall-Through = 1
[...omissis]
```

If you alreay set up an LDAP server for Un*x accounts management, this is enough.

On the LDAP server ensure also that the radius server can read the all the posixAccount attributes (expecially `uid` and `userpassword`).

3.2. Testing Radius Authentication

To test everything server start `radiusd` in debugging mode:

```
/usr/local/sbin/radiusd  -X -A
```

Then use the `radtest` program whith a syntax like

```
radtest username "password" radius.yourorg.com 1 testing123
```

If everything went fine you should receive an Acces−Accept packet from the Radius server.

You can also use stunnel in client mode to provide SSL in the connection between the Radius server and the LDAPS server. For details on SSL refer to Section 10.

3.3. Sample CISCO IOS Configuration

Just for completeness, here is a sample Cisco IOS configuration. Anyway, this is outside the purpose of the HOWTO so it may not suit your needs.

```
[...omissis]
aaa new-model
aaa authentication login default radius enable
aaa authentication ppp default radius
aaa authorization network radius
[...omissis]
radius-server host 192.168.10.1
radius-server timeout 10
radius-server key cisco
[...omissis]
```

> **Note:** Almost all NAS use port 1645 for radius, check it out and configure the server appropriately.

4. Samba

The current stable samba tree does not contain ldap support. The HEAD and TNG branches should, but are still under heavy development. When stable version are released I will document the implementation of samba in it here. Until then, you might want to take a look a document written by Ignacio Coupeau, where he describes the setup of Ldap for both of these branches.

Anyway, at this point, the `smbpasswd` file still has to be used. User account information is already retrieved from ldap, though. (As this is done by nsswitch, not samba.) When samba supports ldap, it should be possible to store the information that is now contained in the `smbpasswd` and optionally `smbusers` files in ldap. Whether it is possible for shares to be dynamically defined in ldap, I don't know, but I suppose it is not.

5. DNS

There are to ways of dns that can be configured via ldap, client side and server side. The first, client side, is using the name server switch to access the dns entries in the Ldap database. This means that only clients that modify their `/etc/nsswitch.conf` file will see the dns entries from ldap. The second way to do it is to use ldap as a backend for bind or tinydns. There are some projects going on about this subject and i will describe them below.

5.1. Using nss

When using nss to access (additional) host entries, please take not that only "friendly" machines (e.g. machines that you know of and whoes configuration you can control) can use this service. It might be useful for intranet host lookups that change often, but it cannot be used to distribute your webservers virtual hostnames to the world. Note that also the **nslookup** command bypasses both `/etc/hosts` and ldap, so it cannot be used to check if your setup is working. Use something like **host** or **ping** instead, which does a lookup with the internel gehostbyname() function.

5.1.1. Configuration

To have the name server switch use ldap for dns lookups it must be configured with nss_ldap. How to set up nss_ldap, you can find in Section 2. Here i will assume you have a working nss_ldap configuration. The dns lookups of nss are controlled with the *hosts* line in `/etc/nsswitch.conf`. It is very unlikely that you do not already have a hosts line. Most probably it will contain the *files* and *dns* entries. You should add *ldap* to it like this:

```
hosts:          files, dns, ldap
```

Think well about the order in which you specify these! It is advised always to put *files* as the first entry. Then, if you want ldap to override your local dns server, you have to make sure that the ip of the ldap server can be found in the `/etc/hosts` file. If not, you will have a nice recursive lookup going. –– You want to look up a host, it's not in files, so we try to contact the ldap server, whoes ip we don't know, so we try to look it up in files, where we cannot find it, so we try to contact the ldap server –– get the point? You could bypass this problem entirely by referring to your ldap server with an ip number instead of a hostname (in `/etc/ldap.conf`, that is.)

5.1.2. Schema

The schema used for this, and similar services, can be found in RFC 2307. Entries used for mapping names to ipnumbers are in an objectclass *ipHost*. The name part of the mapping is given in the attribute *cn*, while the ip part lives in *ipHostNumber*. A typical ldif entry would therefore look like this:

```
dn: cn=somehostname.mydomain.com,ou=Network,o=YourOrg,c=NL
objectclass: top
objectclass: ipHost
cn: somehostname.internal.example.com
ipHostNumber: 10.1.5.13
```

Of course, the usual restrictions and possibilities that come with dns apply.

5.2. Using bind

There are a few possibilities with bind or tinydns nowadays, but imho none of them is a "real" solution (yet). I must say, however, that i have no experience with any of them. They are listed below.

5.2.1. Bind patch

David Storey is working on a patch for Bind, which makes it get its data directly from ldap. This means that every time a request is performed on the bind daemon, it does a lookup in ldap. At this time, his future plans were: (Taken from the source) to have at least two modes of operation: cached and dynamic. Cached mode operates just like an rbtdb by loading the entire zone into memory and reloading whenever the server is HUP'ed. Dynamic mode is much like it is now: every request means an LDAP lookup. For up to date information you should check out the sources.

5.2.2. ldap2dns

Taken entirely from their website:

ldap2dns is a program to create DNS records directly from a LDAP directory. It can and should be be used to replace the secondary name−server by a second primary one. ldap2dns helps to reduce all kind of administration overhead. No more flat file editing, no more zone file editing. After having installed ldap2dns, the administrator only has to access the LDAP directory. If he desires he can add access control for each zone, create a webbased GUI and add all other kind of zone and resource record information without interfering with the DNS server. ldap2dns is designed to write binary `data.cdb` files used by tinydns, but also may be used to write `.db`−files used by named.

The projects homepage is here.

5.2.3. ispman

ispman is a perl−based isp management package. It uses an ldap database backend for it's configuration. It can do lot's of things, so you might check out what you need exactly. It's at ispman.org.

6. Mail Transfer Agents

i will describe three different MTA's in this section, Sendmail, Postfix and Qmail. These are three MTA's that can be configured to use Ldap for the retrieval of information. From personal experience, I must say that Postfix is *much* easier to set up than sendmail, but this may change in the future, as the ldap support in sendmail develops to a more mature state. I have not used qmail myself.

6.1. Sendmail

6.1.1. Ldap support in sendmail

Sendmail has support for ldap since somewhere around version 8.8.x using the map type *ldapx*.From version 8.10 and up an ldap database type *ldap* is supported. Please note that the ldap map support is not enabled per default in the RedHat package. Debian versions 2.2 and later do have ldap support in their sendmail, I am told. If you have to compile it yourself, please read the file `sendmail/README` from the sendmail sources. It contains valuable information about how to compile in the ldap support.

Both the old and the new ldap map type have the ability to look up entries in an ldap database. There is one thing that must be noted however. When a search is being done, only one result should be returned. If more results are found, only the first is used. Additionally, is multiple return values for that result are found, only the first is returned. Let's take a look at the following example ldif file:

```
dn: cn=mailuser1,ou=mail,dc=company,dc=com
objectclass: top
objectclass: foo
cn: mailuser1
mail: mailuser1@company.com
mail: info@company.com
```

If a search is performed with a simple search filter like *cn=mailuser1*, and the attribute that is asked for is *mail*, only *mailuser1@company.com*, is returned. To get both results, they should we stored in a single−valued attribute with comma−separated values, like this:

```
mail: mailuser1@company.com, info@company.com
```

An email message containing information related to this subject can be found at the LIH home.

6.1.2. System layout.

When ldap maps are available, almost anything can be looked up in an ldap database. What we would like to do is to simplify the configuration of the following setup.

Let's say we have a medium−sized or large network where we recieve for many domains. We have two mailhosts and two or three fallback hosts. This setup will normally have four places where three types of information are stored.

- The mailhosts both need a file , or traditionally `sendmail.cw`, to store the domains for which they should recieve mail. The fallback hosts keep the same information in the `access` file, but they use it to list the domains for which they should relay any incoming mail.

* The mailhosts both have a `virtusers` file, which maps multiple adresses (or entire domains) to a single virtual or local user.
* The mailhosts both have an `aliases` file, which maps virtual users to one or more mail addresses or local users.

If this information is stored in a single database, each host will read its configuration from that database, which improves both the manageability and the scaleability of the network. One could even think of a single host which carries all data, mapped to with nfs. In such a case it makes no difference anymore to which host we are connecting. To a user, they appear to be all the same.

6.1.3. Sendmail configuration file

To understand how this information is read from an ldap database instead of the regular files a little background knowledge of the `sendmail.cf` file is neccesary. The information we're dealing with here is stored in two different ways. The `local-host-names` file is read into a class (class w, to be exactly, hence the old extension cw), while the virtusers file is used through a simple map. The aliases file is also a map, but it is defined in a different manner and used internally, instead of being referred to in rules.

When information is retrieved from a ldap database, it always ends up being in a map. This is somewhat problematic with the information stored in the `local-host-names` file, because this used to be a class. I have been unable to fill the class with the information from the map or something alike. That would be the easy way, but I think it's not possible. (If I'm incorrect about this, please let me know). Therefore, I had to define a new map, and insert rules in the sendmail configuration to make sure that (almost) every time when a value is looked up in the class, the new map is searched for the value too.

For the maps, the change of configuration is easy. A map is normally defined with a name and database type, and some database−specific options, (like the location of the file, for the normally used newdb database types). So for maps, it suffices to change the definition of the map and voila, we're done. Ldap maps have a few more options, some of which can be predefined. They are explained in the following list (largely taken from Booker Bense's document):

Ldap−specific map options in sendmail.cf

−h

Defines a space−separated list with hostnames of ldap servers. All machines will be queried in this order, until a result is found. This can be configured globally.

−b

Defines the ldap search base, e.g. the ldap directory you are going to search in. This can be configured globally.

−k

Defines the ldap search filter. It is a "sprintf" style string that defines how the map takes it's input value and constructs an ldap search. It has the form of an ordinary ldap search filter, with %s replaced by the value that is searched for. To learn more about ldap search filters please see RFC 2254. The search filter and the search base used above should define a search that returns at most one entry. The ldap map will only use the first entry it recieves.

−v

Defines the ldap attribute of which the value will be returned by the map lookup. Explained in detail later.

PLease note that all ldap options must be double−quoted and must immediately follow the sendmail option. Here is an example:

```
Kldapexamplemap ldap -h"localhost ldap.myorg.com" -b"ou=mail,dc=myorg,dc=com" -k"(mailstuff)(uid=%s))" -v"mailaddress"
```

6.1.4. Schema

For this particular setup I have defined a subtree *mail* in the ldap directory, under which all mail−related information will be stored. I t would have been possible to store some of the user−related mail−information in the *ou=Users* subtree, but I have specificalle chosen not to do this. When using a separate subtree, all information for sendmail is stored in a single place, and, when having many users, searches may be faster, because not the entire *ou=Users* subtree needs to be searched, but only the *ou=mail* subtree.

In this subtree two kinds of records will appear.

1. Entries that hold mappings from the `virtuser` file and `aliases` file, for a single virtual user. I have chosen to store both mappings in a single entry, because this clarifies the effect and configuration used. For this I have defined an objectclass *inetmailrecipient*, and three attributes, *mailid*, *mailacceptinggeneralid* and *maildrop*.

   inetmailrecipient

   This is a classification that tells us the entry is some form of mapping from one or more real mail addresses and/or mail domains to one or more real users.

   mailid

   This describes the mail addresses that this virtual user recieves mail for. Can be in the form of normail addresses, like *foo@myorg.com*, or complete domains like *@my2nd.org*. Multiple of these attributes may be present, but they should all contain only one value. For each of these id's mail is sent to *mailacceptinggeneralid*.

   Here you'll put what you used to put at the left side in the `virtusers` file.

   mailacceptinggeneralid

   Defines a virtual user. This is in fact the link between the `virtusers` and `aliases` file. In each entry, one attribute of this type must be present, but no more than one may be present, and the attribute may contain only one value. This can either be a local username, or a virtual user. In the second case, a maildrop attribute must be present, in the first case, it does not.

Here you'll put what you used to put in the `virtusers` file, and at the left side in the `aliases` file.

maildrop

Defines the addresses and/or users that mail received for should be delivered to. Only one attribute of this type may be present, but it may contain a comma−separated list of addresses and/or users. If the value of *mailacceptinggeneralid* is a virtual user, this attribute must be present. If the value is a real user, this entry may be omitted.

Here you'll put what you used to put at the right side in the `aliases` file.

In general, one could say that the *mailid* and *mailacceptinggeneralid* together provide the functionality of the `virtusers` file, and *mailacceptinggeneralid* and *maildrop* together provide the functionality of the `aliases` file.

2. One or more entries that hold the domain names which would normally apear in the `sendmail.cw` and `access` files. For this I have defined an objectclass *inetmaildomain*, and three attributes, *maildomain*, *sendmailislokalkey* and *sendmailaccesskey*.

inetmaildomain

This is a classification that tells us the entry lists mail domains which belong to our system, and either should be delivered locally, or be relayed to another host.

maildomain

Defines the mail domain. Multiple of these attributes may be presentin a single entry. The value should be the domain without the "`@`".

For each of the dmain entries in the *local−host−names* file, one of these attributes should be present.

sendmailislocalkey

This defines a simple key that is used in the sendmail rules to determine if a domain is local. It can be anything really, but is must be the exact string you will use in your sendmail rules. For now I have used *`<LDAPLOCAL>`*. One attribute of this type must be present, and no more than one may be present in each entry.

sendmailaccesskey

This defines the key that is used in the sendmail rules to determine what action needs t be taken for the specified domain. It can be one of *RELAY*, *OK*, *REJECT*, *DISCARD* or an error indicator. (For detailed information see the `cf/README` file from the sendmail sources.)

Note: PLease note that in this particular setup I will only be using entire domains in the *access* file. That means that in this case I can use the *maildomain* attribute for both the information used in the *access* file and the information used in the *local−host−names*. If you want a finer control over your access list, you should define a separate entry to hold that information, instead of using the *maildomain* attribute.

6.1.5. More information.

Here are a few sources of information that might be useful:

- Booker Bense has written a document about the usage of ldap with sendmail 8.9.3. He says it's the wrong place to start when learning about using sendmail and ldap, but it has been of great help to me.
- A short howto about how to setup sendmail with ldap, written by Jason Christopher Radford
- New articles about ldap and sendmail are being published on sendmail.net, written by Michael Donnelly, originating from ldapmap.org, which has also a lot of general interesting ldap–related information.

6.2. Postfix

6.2.1. Support

Postfix has native ldap support. A lot of options in postfix can be configured using *maps* which can be of various types. One of those types is ldap. For each ldap map a couple of options can be configured. (See Section 6.2.2.)

The process of having postfix look up certain data in an ldap database is pretty straightforward. The most common use (so i think) is to have postfix look up virtual users in the ldap database. Together with the above explaind nss_ldap feature, this allows you to have all your email users in an ldap database. But other things can be configured too, like the domains postfix is allowed to relay mail for, or relay mail from, or for which is should act as a backup mail server.

6.2.2. Configuration

The description of the configuration options is completely taken from LDAP_README in the postfix docs from version 20001217.

server_host

The name of the host running the LDAP server, e.g.

```
ldapsource_server_host = ldap.your.com
```

It should be possible with all the libraries mentioned above to specify multiple servers separated by spaces, with the libraries trying them in order should the first one fail. It should also be possible to give each server in the list a different port, by naming them like "ldap.your.com:1444".

server_port (389)

The port the LDAP server listens on, e.g.

```
ldapsource_server_port = 778
```

search_base (No default; you must configure this.)

The base at which to conduct the search, e.g.

```
ldapsource_search_base = dc=your, dc=com
```

timeout (10 seconds)

The number of seconds a search can take before timing out, e.g.

```
ldapsource_timeout = 5
```

query_filter (mailacceptinggeneralid=%s)

The RFC2254 filter used to search the directory, where %s is a substitute for the address Postfix is trying to resolve, e.g.

```
ldapsource_query_filter = (%s)(paid_up=true))
```

domain (No default; you must configure this.)

This is a list of domain names, paths to files, or dictionaries. If specified, only lookups ending in a domain on this list will be searched. This can significantly reduce the query load on the LDAP server.

```
ldapsource_domain = postfix.org, hash:/etc/postfix/searchdomains
```

result_attribute (maildrop)

The attribute(s) Postfix will read from any directory entries returned by the lookup, to be resolved to an email address.

```
ldapsource_result_attribute = mailbox,maildrop
```

special_result_attribute (No default)

The attribute(s) of directory entries that can contain DNs or URLs. If found, a recursive subsequent search is done using their values.

```
ldapsource_special_result_attribute = member
```

scope (sub)

The LDAP search scope: sub, base, or one. These translate into LDAP_SCOPE_SUBTREE, LDAP_SCOPE_BASE, and LDAP_SCOPE_ONELEVEL.

bind (yes)

Whether or not to bind to the LDAP server. Newer LDAP implementations don't require clients to bind, which saves time. Example:

```
ldapsource_bind = no
```

If you do need to bind, you might consider configuring Postfix to connect to the local machine on a port that's an SSL tunnel to your LDAP server. If your LDAP server doesn't natively support SSL, put a tunnel (wrapper, proxy, whatever you want to call it) on that system too. This should prevent the password from traversing the network in the clear.

bind_dn ("")

If you do have to bind, do it with this distinguished name. Example:

```
ldapsource_bind_dn = uid=postfix, dc=your, dc=com
```

bind_pw ("")

The password for the distinguished name above. If you have to use this, you probably want to make main.cf readable only by the Postfix user. Example:

```
ldapsource_bind_pw = postfixpw
```

cache (no)

Whether to use a client−side cache for the LDAP connection. See ldap_enable_cache(3). It's off by default.

cache_expiry (30 seconds)

If the client−side cache is enabled, cached results will expire after this many seconds.

cache_size (32768 bytes)

If the client−side cache is enabled, this is its size in bytes.

dereference (0)

When to dereference LDAP aliases. (Note that this has nothing do with Postfix aliases.) The permitted values are those legal for the OpenLDAP/UM LDAP implementations:

0	never
1	when searching
2	when locating the base object for the search
3	always

6.2.3. Example setup

If you want a virtual domain (say foo.virtualdomain.com) and you want to store email addresses in Ldap for this domain, youwould need the following piece in your `main.cf`.

```
virtual_maps = ldap:ldapvirtual
ldapvirtual_search_base = ou=mail,o=YourOrg,c=nl
ldapvirtual_query_filter = (mailacceptinggeneralid=%s)
ldapvirtual_domain = foo.virtualdomain.com
ldapvirtual_result_attribute = maildrop
ldapvirtual_bind = no
ldapvirtual_scope = one
```

With this setting, if postfix receives mail for a user with a domain part "foo.virtualdomain.com", to do a search in the database for entries that have an attribute *mailacceptinggeneralid* that matches "user@foo.virtualdomain.com". If such an entry is found, the values of all available *maildrop* attributes are returned, and to these values the mail is delivered. If "user@foo.virtualdomain.com" is not found, another query is performed, that tries to match the catchall user, "@foo.virtualdomain.com". If this is again not found, the message will be bounced.

6.3. Qmail

Qmail itself does not have any ldap support. However, there is a patch by Andre Oppermann that provides ldap support. This package, including the documentation for it, can be found at his site.

7. Address books

A very useful feature of an Ldap database on a linuxserver is that when you have an internal network in your organization, you can have a single place to store all your external contacts. You could even divide it in groups, or departments. It is no longer neccesary to give each employee a seperate address book. Apart from using Ldap, this could also be done with Microsoft Exchange Server, Lotus Domino, and Netscape Active Directory.

To use Microsoft Address Book and programs that rely on it, such as Microsoft Outlook, Microsoft Outlook Express and Microsoft Outlook 2000 there is no need to change the basic ldap configuration. There are two things that need to be modified though.

At first, you have to create a directory tree to store your addresses and relevant data. In Section 11 will be shown which entries will be used in this tree.

Second, you have to make sure that all hosts on your local network have read access to this tree. This will be dealt with in the chapter 'security' which is not finished yet.

All Microsoft Email programs can use the Ldap Directory Services. If you want to search for people, you have to use the Address Book. When composing a new email message, a name can be automatically matched to an email address. To do this, the *cn*,*sn*,*givenname* and *mail* fields are searched. When you want to configure your Microsoft email program to use an Ldap server as your address book, or to look up email addresses, you need to do the following:

1. Start your favorite email program and open the address book. This can be done by selecting Tools, Addressbook from within the program, or via the start menu by selecting Start,Programs,Accessories,Address Book.
2. Click on Tools,Accounts to open the Internet Account window.
3. Click Add, now you get an Internet Connection Wizard window, type the ip address or hostname of your Ldap server, and click OK.
4. On the next window, answer Yes to confirm you want to check your adresses using this directory, or No if you don't want do not want that. Now click Next and click Finish.
5. Now you're back at the Internet Account window. Select your newly−added account and click Properties.
6. On the properties window, click the Advanced tab.
7. In the Search Base field, enter the base of the tree where your adresses will be stored. An example could be *ou=Addressbook,dc=yourorg,dc=com.*
8. Press OK to close the window and click Close to close the Internet Account window. You should have returned to the main Address Book window now.

Now, when you enter a name in the to: field, the email address is looked up in the Ldap Directory, and automagically filled in for you. If an entry is not found, a window is presented, and any typos can be corrected, or a new search can be done.

8. Netscape roaming access

ToDo.

A good article on this subject can be found here.

9. Publishing digital certificates with LDAP

This section focuses on how to publish digital certificates into an ldap server. You need to publish digital certificates if you run a Certificaton Authority. Publishing to LDAP is a simple way to make this information available in the network .Also many certificate aware software uses LDAP as a preferred repository for user certificates.

This allows to keep users certificates with the rest of the user information avoiding useless replication of data.

To deal with certificates you need a cryptographic toolkit, the one used here is OpenSSL.

9.1. LDAP Server configuration

The LDAP server used here is OpenLDAP 2.0.x.

Your LDAP server must support objectclasses that allows attributes to store certificates. In particular you need to store in the LDAP server the Certification Authority certificate, the Certificate Revocation List, the Authority Revocation List and end users certificates.

The `certificationAuthority` objectclass implements the `authorityRevocationList`, `certificateRevocationList` and `cACertificate` attributes.

The `inetOrgPerson` objectclass supports the usercertificate (binary) attribute.

You can also use the mix−in objectclass `strongAuthenticationUser` to add certificates to non `inetOrgPerson` entries.

You can include required schemas to OpenLDAP including the following schemas into your `slapd.conf` file.

```
include         /usr/local/etc/openldap/schema/core.schema
include         /usr/local/etc/openldap/schema/cosine.schema
include         /usr/local/etc/openldap/schema/inetorgperson.schema
```

9.2. Certificate Publishing

Certificates are encoded using ASN.1 DER (Distingushed Encoding Rules). So it must be published into the LDAP server as a binary piece of data (using BER encoding).

You can convert a pem certificate into der format using openssl

```
openssl x509 −outform DER −in incert.pem  −out outcert.der
```

Then an LDIF file can be created using the `ldif` utility provided with OpenLDAP. The command:

```
ldif −b "usercertificate;binary" < outcert.der   > cert.ldif
```

creates an usercertificate attribute encoded in BASE64. You can add this certificate to an LDIF entry and then use `ldapmodify` to add the certificate to an entry.

```
ldapmodify -x -W -D "cn=Manager,dc=yourorg,dc=com" -f cert.ldif
```

Where `cert.ldif` contains something like:

```
dn: cn=user,ou=people,dc=yourorg,dc=com
changetype: modify
add: usercertificate
usercertificate;binary:: MIIC2TCCAkKgAwIBAgIBADANBgkqhkiG9w0BAQQFADBGMQswCQYD
 VQQGEwJJVDENMAsGA1UEChMESU5GTjESMBAGA1UECxMJQXV0aG9yaXR5MRQwEgYDVQQDEwtJTkZO
 IENBICgyKTAeFw05OTA2MjMxMTE2MDdaFw0wMzA4MDExMTE2MDdaMEYxCzAJBgNVBAYTAklUMQ0w
 CwYDVQQKEwRJTkZOMRIwEAYDVQQLEwlBdXRob3JpdHkxFDASBgNVBAMTC0lORk4gQ0EgKDIpMIGf
 MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCrHdRKJsobcjXz/OsGjyq8v73DbggG3JCGrQZ9f1Vm
 9RrIWJPwggczqgxwWL6JLPKglxbUjAtUxiZm3fw2kX7FGMUq5JaN/Pk2PT4ExA7bYLnbLGZ9jKJs
 Dh4bNOKrGRIxRO9Ff+YwmH8EQdoVpSRFbBpNnoDIkHLc4DtzB+B4wwIDAQABo4HWMIHTMAwGA1Ud
 EwQFMAMBAf8wHQYDVR0OBBYEFK3QjOXGc4j9LqYEYTn9WvSRAcusMG4GA1UdIwRnMGWAFK3QjOXG
 c4j9LqYEYTn9WvSRAcusoUqkSDBGMQswCQYDVQQGEwJJVDENMAsGA1UEChMESU5GTjESMBAGA1UE
 CxMJQXV0aG9yaXR5MRQwEgYDVQQDEwtJTkZOIENBICgyKYIBADALBgNVHQ8EBAMCAQYwEQYJYIZI
 AYb4QgEBBAQDAgAHMAkGA1UdEQQCMAAwCQYDVR0SBAIwADANBgkqhkiG9w0BAQQFAAOBgQCDs5b1
 jmbIYVq2epd5iDjQ109SJ/V7b6DFw2NIl8CWeDPOOjL1E5M8dnlmCDeTR2TlBxqUZaBBJZPqzFdv
 xpxqsHC0HfkCXAnUe5MaefFNAH9WbxoB/A2pkXtT6WGWed+QsL5wyKJaO4oD9UD5T+x12aGsHcsD
 Cy3EVEaGEOl+/A==
```

It is also possible to specify the certificate in the LDIF file as:

```
userCertificate;binary:< file:///path/to/cert.der
```

9.3. LDAP Aware Clients

Once you stored certificates in the server you may wonder to retrieve them.

Among other clients, Netscape has supprt to retrieve certificates automatically from an LDAP server. Using the Security Panel−−>User Certificates−−>Search Directory; you can search for certificates in the LDAP dierctory and have them automatically installed in your Netscape certificate database.

Another client that has good support for certificates is web2ldap www.web2ldap.de

10. SSL/TLS and SSL/TLS wrappers for LDAP

10.1. A Brief description of SSL

The Secure Socket Layer (SSL) is an application layer protocol that provides a secure transmission channel between parties. It stands between TCP/IP and application level protocols, such as HTTP, LDAP, SMTP etc... It is based on public key cryptography systems (various ciphers can be used) and on X.509 certificates.

SSL was initially a Netscape protocol, then it has gone trough a standardization process and now is called TLS (Transmission Layer Security). It is commonly referred as SSL/TLS.

The SSL/TLS protocol provides:

- Data encryption: Client/server session is encrypted
- Server authentication: Client can verify the server identity
- Message integrity: Data is not modified during transmission; this prevents "man in the middle" attacks.
- Client authentication: Server can verify the client identity

---

10.2. SSL/TLS availability for OpenLDAP

Since OpenLDAP 2.0.x, that is an LDAP V3 toolkit, SSL/TLS is provided by the server. OpenLDAP 2.0.x needs to be compiled using the OpenSSL library to add SSL/TLS. It also has Start−TLS support.

> **Note:** Start−TLS allows to enable TLS if the client requests it. This way it is possible to use only an LDAP port for both secure and insecure connections.

OpenLDAP 1.2.x, instead, is an LDAP V2 protocol implementation and does not provide SSL/TLS.

Valuable information on SSL/TLS on OpenLDAP 2.0.x can be found on the OpenLDAP web site, here we will focus how to use an SSL tunnel to secure LDAP parties that are not SSL/TLS aware

---

10.3. How to use stunnel to provide SSL/TLS to an LDAP V2 server

If you use OpenLDAP 1.2.x you need a general purpose SSL wrapper to add SSL capabilities to the server. Stunnel (www.stunnel.org) has been found to be stable and suitable for this application.

Installing it is quite simple, but first you have to install OpenSSL (www.OpenSSL.org) to have the required library and tools.

OpenSSL, is an open source implementation of the SSL protocol that provides the SSL library and a set of cryptography tools.

To install OpenSSL you have to type the following commands:

```
$ ./config
$ make
$ make test
# make install
```

usually, everything will be installed in `/usr/local/ssl`.

If OpenSSL is correctly installed the only command needed to compile and install stunnel are:

```
$ ./configure
$ make
# make install
```

Stunnel uses a server certificate for SSL, this can be a self signed certificate, or, better, a certificate signed by your own Certification Authority (the SSL client has to trust the CA too).

A commonly used place used to store such certificate is:

```
/usr/local/ssl/certs/stunnel.pem
```

If having a Certification Authority is not a concern, a self signed certificate can be produced using the tools provided by the OpenSSL suite.

In the stunnel directory (to use the configuration file `stunnel.cnf`) type the following commands:

```
$ openssl req -new -x509 -days 365 -nodes -config stunnel.cnf \
            -out stunnel.pem -keyout stunnel.pem
$ openssl gendh 512 >> stunnel.pem
```

This will produce a self signed certificate, valid for a year, in the file `stunnel.pem`.

Once stunnel is installed, you can start up first the LDAP server on port 389 (the default LDAP port):

```
#/usr/local/libexec/slapd
```

Then stunnel on port 636 (the port used by LDAPS client):

```
# /usr/local/sbin/stunnel  -r ldap  -d 636 \
 -p /usr/local/ssl/certs/stunnel.pem
```

For debugging you can start `stunnel` in foreground with the following syntax:

```
# /usr/local/sbin/stunnel  -r ldap  -d 636 \
 -D 7 -f -p /usr/local/ssl/certs/stunnel.pem
```

10.4. How to use stunnel to provide SSL to LDAP clients

Many LDAP client are not SSL aware, anyway, it is possible using stunnel in client mode, to provide SSL to these clients.

This is quite simple. You can start stunnel on the client host, using the LDAPS port, and forward requests to this port to the actual LDAP server:

```
# stunnel -c -d 636 -r  ldapserver.yourorg.com:636
```

Now LDAP clients must be configured using `localhost:636` as the LDAPS server to use.

10.5. How to use stunnel to provide SSL for slurpd replication

At the moment slurpd (slapd replication daemon) hasn't SSL capabilities, anyway you can use stunnel in client mode to have this job done.

Using stunnel in client mode on the master, you can forward a local port to a remote port:

```
# stunnel -c -d 9636 -r  ldapreplica.yourorg.com:636
```

and have on the master LDAP server in `slapd.conf`

```
replica host=localhost:9636
```

11. Ldap schema's

Warning: this section is terribly incomplete and outdated. I should be updating it, adding the various rfc's or other authoritive sources of schematic data.

This is a proposition of a schema that can be used to accommodate all the data needed for the previously listed functions. It should under no circumstances be regarded as authoritative. It is an example that should serve it's purpose, but it is likely you have to adapt it to match your specific needs.

Because it has been a lot of work (for me, maybe it's out there but I don't know where?) to find out the specific meaning of each entry, and what information it should contain, I'll try to do this as well. It should be noted, however, that it doesn't fit together seamlessly. The Microsoft Addressbook does not seem to use some of the fields it is presenting. I suspect that for the "Title", "Nickname", "Home City", "Home State/Province", "Home ZIP Code", "Home Country/Region" and "Home Web Page" entries no information is requested. For the "Personal", "Netmeeting" and "Digital IDs" I didn't yet bother to figure out how it should be put in the Ldap database. Any information is welcome. The netscape address book has a similar problem. When a record is copied from an LDap directory to a local address book, some of the fields are lost. As the nature of an company−wide addressbook should discourage users to copy addresses locally, this is not a big problem though. But netscape address book has another little oddity though. In a normal address record, the Ldap attribute associated with "Nickname" is *xmozillanickname*. When searching for addresses however, the associated attribute is simple *nickname*. That is the reason why the nickname entry shows up twice in the schema.

This schema is known to work with Microsoft Outlook 2000, and Netscape 4.73. If you find I'm wrong about a description, function, or neccessity of an entry, please do let me know!

The schema file that represent this schema can be found in Section 12.1.

Table 1. Ldap attributes and objectclasses − quick description

Function	Objectclass	Attributes	Description	(Default) value
User accounts	top		default	
		ou	Organizational Unit	Users
	person		Owner is a person	
		uid	unix login name	foo
		cn	Common Name	Foo Bar
		sn	Surname	Bar
	account		Owner has an account	
	posixaccount		Owner has a Unix account	
		uidNumber	uid	513

		gidNumber	gid	100
		homedirectory	Home directory	/home/users/foo
		userpassword	unix password	S3cr3t
	sambaaccount		Owner has a samba account	
		ntuid	Unknown	uid
		rid	Unknown	uidnumber
		lmpassword	Lanman password hash	Unused
		ntpasswd	NT password hash	Unused
		loginshell	Users shell	/bin/pleurop
Machine accounts	top		default	
		ou	Organizational Unit	Machines
	posixaccount		Owner has a unix account	
		uid	login name	speed$
		uidnumber	unix uid	514
		gidnumber	gid	100
		homedirectory	Home directory	Unused
Microsoft Address Book	top		default	
		ou	Organizational Unit	Addressbook
	microsoftaddressbook		Owner has Microsofts Addressbook properties	
		cn	Name	
		c	Business country	
		department	Business department	
		facsimiletelephonenumber	Business fax number	
		givenname	First name	
		homephone	Home phone number	
		homepostaladdress	Home postal address	

		info	Notes	
		initials	Initials	
		l	Business city	
		mail	Email address	
		mobile	Home cellphone number	
		organizationname	Company name	
		otherfacsimiletelephonenumber	Home fax number	
		otherpager	Business pager number	can be "pager" too?
		physicaldeliveryofficename	Location of office at work	
		postaladdress	Business postal address	
		postalcode	Business postal code	
		sn	Last Name	
		st	Business state/province	
		telephonenumber	Business phone number	
		title	Job title	
		url	Business web page	
Netscape Address Book	top		default	
		ou	Organizational Unit	Addressbook
	netscapeaddressbook		Owner has Netscape's properties	
		cn	Name	
		cellphone	Cellphone number	
		countryname	Country	
		description	Description	
		facsimiletelephonenumber	Fax number	
		givenname	First Name	
		homephone		

			Home phone number	
		homeurl	Personal web page	
		locality	Home city	
		mail	Email address	
		nickname	Nickname	
		o	Company	
		ou	Department	
		pagerphone	Pager number	
		postalcode	Home postal code	
		sn	Last name	
		st	State	
		streetaddress	Home postal address	
		telephonenumber	Business phone number	
		title	Title	
		xmozillaanyphone	Business phone number	
		xmozillanickname	Nickname	Same as nickname
		xmozillausehtmlmail	Client uses html mail	TRUE
Netscape roaming access	top		default	
		ou	Organizational Unit	Roaming

> **Note:** Netscape and Microsoft use the addressbook entries in a slightly different way. Netscape stores a postal address in the streetaddress entry in a base64 encoded string, while Microsoft uses the postaladdress entry. However, when a streetaddress entry is present, Microsoft uses this instead of the postaladdress entry, but it's value is stored plaintext, not base64 encoded. So you cannot use them at the same time.

More information about Ldap schema's in general can be found on Linux Center. I found a document describing Microsoft Addressbook's properties on the Microsoft Developers Network.

Beware, the description given on the Microsoft page doesn't match the fields where the content shows up in address book. Also, not all fields in address book contain information, but if the listed keys don't work I wouldn't know which keys *do* work.

12. Example files

Here are the example files that can be used to setup an installation as I described here.

12.1. The schema file

```
# Unix related and default classes (Modified)

attribute       userpassword                            ces
attribute       telephonenumber                         tel
attribute       facsimiletelephonenumber        fax     tel
attribute       pagertelephonenumberpager               tel
attribute       homephone                               tel
attribute       mobiletelephonenumber           mobile  tel
attribute       member                                  dn
attribute       owner                                   dn
attribute       dn                                      dn

objectclass top
        requires
                objectClass

objectclass organization
        requires
                objectClass,
                o
        allows
                description

objectclass organizationalUnit
        requires
                objectClass,
                ou
        allows
                description

objectclass person
        requires
                objectClass,
                cn
        allows
                description

objectclass account
        requires
                objectClass,
                uid
        allows
                description,
                host,
                o,
                ou

# Samba related classes (Original)

objectclass sambaaccount
        requires
                objectclass,
```

```
                uid,
                uidnumber,
                ntuid,
                rid
        allows
                gidnumber,
                grouprid,
                nickname,
                userpassword,
                ou,
                description,
                lmpassword,
                ntpassword,
                pwdlastset,
                smbhome,
                homedrive,
                script,
                profile,
                workstations,
                acctflags,
                pwdcanchange,
                pwdmustchange

objectclass sambagroup
        requires
                cn,
                rid
        allows
                ntuid,
                member,
                description

objectclass sambaconfig
        requires
                id
        allows
                nextrid

objectclass sambabuiltin
        requires
                cn,
                sid
        allows
                ntuid,
                rid,
                member,
                description

# Sendmail related class (new / modified)

objectclass inetmailrecipient
        requires
                objectclass
        allows
                mailid,
                mailacceptinggeneralid,
                maildrop

objectclass inetmaildomain
        requires
                objectclass,
                sendmailislocalkey
```

```
        allows
                maildomain,
                sendmailaccesskey

# Addressbook related classes

objectclass netscapeaddressbook
        requires
                objectclass,
                cn
        allows
                cellphone,
                countryname,
                description,
                facsimiletelephonenumber,
                givenname,
                homephone,
                homeurl,
                locality,
                mail,
                nickname,
                o,
                ou,
                pagerphone,
                postalcode,
                sn,
                st,
                streetaddress,
                telephonenumber,
                title,
                xmozillanickname,
                xmozillausehtmlmail,
                xmozillaanyphone

objectclass microsoftaddressbook
        requires
                objectclass,
                cn
        allows
                c,
                department,
                facsimiletelephonenumber,
                givenname,
                homephone,
                homepostaladdress,
                info,
                initials,
                l,
                mail,
                mobile,
                organizationname,
                otherfacsimiletelephonenumber,
                otherpager,
                physicaldeliveryofficename,
                postaladdress,
                postalcode,
                sn,
                st,
                telephonenumber,
                title,
                url
```

12.2. Example base ldif

```
dn: dc=yourorg,dc=com
objectClass: top
objectClass: organization
o: YourOrg
description: This is our organizations base dn. Everything is stored beneath this

dn: ou=Users,dc=yourorg,dc=com
objectClass: top
objectClass: organizationalunit
ou: Users
description: This is the tree were user accounts are stored

dn: ou=Machines,dc=yourorg,dc=com
objectClass: top
objectClass: organizationalunit
ou: Machines
description: This is the tree were machine accounts are stored

dn: ou=Roaming,dc=yourorg,dc=com
objectClass: top
objectClass: organizationalunit
ou: Roaming
description: This is the tree were netscape roaming profiles are stored

dn: ou=Addressbook,dc=yourorg,dc=com
objectClass: top
objectClass: organizationalunit
ou: Addressbook
description: This is the tree were addressbook entries are stored
```

Notes

[1]
A mechanism that permits LDAP database replication between servers.

[2]
It is not a case that these are the maps provided by NIS.

[3]
An entry can belong to several objectclasses.

[4]
In a tricky way, it is also possible to use the Netscape Communicator certificate database.