Part I: Using OpenLDAP on Debian Woody to serve Linux and Samba Users

Note on Debian Sarge:
Please check out Part II: OpenLDAP on Debian Pre-Sarge below for some information on LDAP with a pre-release version of Debian 3.1, aka "Debian Sarge".

Table of Contents

For Debian 3.0 aka Debian Woody:

For Debian 3.1 aka Debian Sarge pre-release version:

User Comments:

Introduction

LDAP is one hell of a tool: it can be used to store any kind of information, starting with your network’s users (which is what we’ll do) and not even ending with your favorite cooking recipes.
As LDAP is one hell of a tool, it is all a pain in the you-know-what-I-mean to get to know it and to get it up and running. I spent *lots* of time with basics just to understand it. One problem for me was, that I didn’t find any good documentation on this topic for a long time.
Anyway, here is first of all a small list of IMHO good documentation on this topic as well as my stuff to get it working: OpenLDAP (the software written to host the database and do some other stuff) implements one part of the whole LDAP-specification, AFAICT. We’ll use it to do the major work: host the database. This "LDAP-server" (ldap.subnet.at) will serve to Linux and Windows workstations hosting the local users and corresponding information. Later on, it shall also serve the upcoming new Linux-based mailserver.

As Debian GNU/Linux is our distribution of choice, I’ll focus on the description for Debian Woody. Nevertheless, lot’s of stuff is generic material and you should be able to use it on other distributions too.

I’d like to thank all people I know and those I don’t know which made this LDAP solution possible - just to mention a few groups: Debian, OpenLDAP, Samba, #ldap (irc.debian.org), the authors of all these Howto’s and other documentations, package maintainers, etc. etc. etc.
Thanks!

This document was created during my work as network admin at subnet - platform for media art and experimental technologies.
This document’s home is http://homex.subnet.at/~max/ldap/.

PDF version of this document

As Postscript- or PDF-versions of this document have been requested several times: I created this file’s HTML/PHP code directly using vim -- which makes it a bit harder to create a proper PDF document that’s up to date.
(If only I knew that this document became this large -- I’d really spent the time to learn DocBook first, or had used LyX -- or whatever.)

Still, Andreas Heinzen pointed out to me, how to easily create a PDF version of this document (the latest version is as of June 11, 2005). Many thanks again to Andreas for his work and feedback on this!
(Just to let you know - in case you want to do this yourself: Use html2ps and ps2pdf to create the document. Beforehand, the feedback-form and the counter should be removed from the source code.)

Document History

- *05-09-18:*
  Added comment for correct handling of command-line parameter in tcsh shell.
- *05-06-11:*
  Added notes on work with Sarge-Pre-Release packages in Part II of this document.
  Updated the PDF version accordingly.
- *05-05-08:*
  Uploaded Andreas Heinzen’s PDF version of this document.
- *04-12-22:*
  Changed the style-sheet to reflect a more common design.
  Mentioned that Samba packages version 2.2.3a-13 are vulnerable.
- *04-11-05:*
  Added the section "NSCD and /etc/libnss-ldap.conf" describing some possible security-improvements.
- *04-10-22:*
  Added a comment about the format of `ldap.secret`.
  Use `dpkg-buildpackage` to compile the Deb-packages (instead of `debian/rules` directly).
- *04-09-19:*
  Added a link to Gerald Carter’s LDAP book to the resources section as well as some links in the miscellaneous section.
- *04-09-02:*
  Eventually, re-compiled and added security-fixed Samba-packages based on version 2.2.3a-13.
- *04-07-15:*

Added and updated some links to LDAP client programs and user-interfaces.

- *04-03-25:*
  Mention Samba security update (see DSA-463).
- *04-02-18:*
  Dual-licensed the document under GPL and GFDL.
- *04-02-09:*
  I’m considering to dual-license the document, adding the GPL as another choice (as Debian considers the GFDL to be a non-free license).
  Mind: According to the "draft" of the Debian Documentation Policy (http://www.debian.org/doc/manuals/ddp-policy/ch-common.en.html#s2.2) the GFDL as is applied here should be considered "free" even for Debian.
- *03-10-10:*
  Added section "What Is LDAP?".
- *03-08-29:*
  Added two links to the resources section.
- *03-08-18:*
  Releasing a security advisory.
  According changes to the available packages and their descriptions.
  Minor changes to clear things up.
- *03-08-13:*
  Add information on how to have Samba automatically create a machine account.
  (Change script "create-machine-account.sh" for this too.)
- *03-08-12:*
  Minor changes to clarify things up.
  Release the document under the GNU Free Documentation License.
  Add information on the MigrationTools.
  Add chapter "Miscellaneous".
  Further minor changes.
- *03-08-11:*
  "Version 1.0"!
  Finally, the first official release with only minor ToDo’s left.

Security Advisory

I wouldn’t have thought it to be necessary with a HOWTO, but it is: This section is for security issues coming up.

03-08-18

Overview:
The self-compiled Samba packages (DSA-280-1 samba -- buffer overflow) as well as the self-compiled LDAP packages (DSA-227-1 openldap2 -- buffer overflows and other bugs) used in previous versions of this HOWTO unfortunately are based on vulnerable versions of those packages. If you’ve simply downloaded and used the packages from this site, you are strongly encouraged to either recompile them yourself or use the new upgraded packages provided here.

Description:
For some reason I did not include `"deb-src http://security.debian.org woody/updates main contrib non-free"` in my `/etc/apt/sources.list` file when initially downloading and compiling the source packages, this means I used Woody's original packages which meanwhile turned out to be vulnerable here and there.)

Mind:
As to my knowledge, packages can now be considered "secure" currently. Nevertheless, Today's security advisory does not mean I necessarily put possibly needed packages up here in the future as well. Don't rely on this howto, keep track of security issues yourself!

04-03-25

There is a local root exploit in Samba 2.2.3a-12 that is fixed in Woody's 2.2.3a-13 packages (check out DSA-463).
04-09-02: Recompiled packages based on the fixed version 2.2.3a-13 are provided below now.

04-12-22

Samba 2.2.3a-13 is vulnerable, see DSA-600.
I removed the compiled packages, please follow the instructions below to build them yourself.

Licensing

This document is published under the licenses GPL and GFDL (see notes below for details). You may choose, which license to apply.

Copyright

GNU Free Documentation License (GFDL)

GNU General Public License (GPL)

Disclaimer

This document comes without any warranty and does not claim to be complete, nor does it necessarily hold correct information. The author(s) can not be held reliable for any loss of data or corrupted hardware or any other miscomfort due to information of this document. Use this at your own risk!

Original Author

- Markus Amersdorfer (markus.amersdorfer <at> subnet.at)

Hosted by

- subnet - platform for media art and experimental technologies

External Resources

If you want to get to know what LDAP is and how its data is organized, please check out the docs in the following subsection "Learning about LDAP".

- Learning about LDAP:
    - A really great and short introduction to LDAP by A. Frisch.
      It also links to the author’s more complete 3-part series: Exploring LDAP -- Part I (Basics), Exploring LDAP -- Part II (Directories) and LDAP -- Part III (Advanced Topics).
    - Some good articles are over at www.ldapman.org, including how to design an LDAP-Tree.
    - Norbert Klasen wrote his master thesis "Directory Services for Linux in comparison with Novell NDS and Microsoft Active Directory" about LDAP and similar setups. Lots of basic as well as detailed information there.
    - While I haven’t read it: Gerald Carter’s LDAP System Administration (ISBN: 1-56592-491-6). The sample chapter is about "Email and LDAP" and covers both configuration of MUAs and MTAs (Sendmail, Postfix, Exim)!
- Getting LDAP up and running:
    - Torsten Landschoff’s Using LDAP for name resolution is a compressed article that describes LDAP on Debian for NIS-like users (using Debian’s default LDAP packages without recompilations).
    - Here’s a great step-by-step guidance on how to get LDAP up and running on Debian Woody.
    - Samba & LDAP ...on Debian made simple! by "mawi" does exactly what it says it does :). Great! (Thanks to "mawi", comment from Aug 26, 2003.)
    - A step-by-step guidance for Mandrake 9.x is Using OpenLDAP for Authentication. Lot’s of good ideas and fine-tuning.
      Further articles ’round Mandrake and LDAP: Implementing a Samba LDAP Primary Domain Controller Setup on Mandrake 9.x and Implementing Disconnected Authentication and

PDC/BDC Relationships Using Samba and OpenLDAP.
"mandrakesecure.net" seems to be down currently (03-08-11) as the server is moved (AFAIK), but you should be able find these document in Google's cache. Buchan Milne, author of the third Mandrake-LDAP-HOWTO mentioned here, pointed me to a different server for his document. (According to him, it might be slightly out-of-date.)
    - Here's a SAMBA (v 2.2) PDC LDAP v.3 howto. Samba 3.x is dealt with in the according Samba (v.3) PDC LDAP howto, which is work in progress currently.
- Tools and stuff:
    - The LDAP Schema Viewer (in case this one's down, you may also try this link).
    - Also see LDAP-Client-Interfaces and Miscellaneous below.
- Miscellaneous:
    - The IRC channel `"#ldap"` on irc.debian.org.
    - RFC 2254 describes "The String Representation of LDAP Search Filters" and concludes with a few examples on how to use the filters.
    - The homepage of Christof Meerwald with some notes on LDAP and the patched Debian package `libpam-ldap`.
    - www.ldapguru.org.
    - pgina.xpasystems.com/ - a GINA for Windows 2000/XP. (Thanks to "unkown", comment from Aug 14, 2003.)
    - Building a LAMP Server w/ LDAP Authentication.

What Is LDAP?

I will not go into details, what LDAP really is and how to best design an LDAP tree - at least not for now. There *are* resources out there which can and do explain this, see section External Resources.

Nevertheless, I'd like to cite from the article Building an LDAP Server on Linux, Part 1 by Carla Schroder as she points out some IMHO crucial thing concerning the LDAP-world:
"Let's get like all pedantic for a moment (please put on your geek beard and pocket protector for this). LDAP--Lightweight Directory Access Protocol--is a protocol, not a database. It accesses a special kind of database that is optimized for fast reads. Use it for relatively static information, such as company directories, user data, customer data, passwords, and security keys. OpenLDAP uses the Sleepycat Berkeley DB. Having said all that, I'm not the pedant police; I'm OK with calling the whole works a database and being done with it."

Install OpenLDAP

First thing to do is to install the OpenLDAP-server.
I'd like to thank the author of the LDAP HOWTO over at howto.aphroland.de a lot. This doc helped me *a lot* by describing the installation from a Debian user's point of view. Lots of my description (above all to get this "LDAP-thing" do something close to what I wanted) is based on this doc and thus my howto cuts off some details which can be found at aphroland.de.
I'd like to notice one difference in advance: while aphroland's description uses a base structure like `"o=domain,c=country"` for the LDAP tree, I'll use the more common `"dc=domain,dc=country"`. Nevertheless, this actually depends on your taste (among other things)

and is just kind of a naming-convention.

We want our server (as well as the clients later on) to support *SSL*, so we’ll have to recompile and install our own Debian packages:

```
Get the source:
  cd ~
  mkdir slapd_woody-source
  cd slapd_woody-source

  apt-get source slapd
  apt-get build-dep slapd
  apt-get install libssl-dev

Activate SSL:
  cd openldap2-2.0.23
  vi debian/rules
    --> and replace --without-tls with --with-tls
  [ vi debian/changelog ]

Compile the packages:
  dpkg-buildpackage -b -us -uc
```

FYI: "slapd" is the part of OpenLDAP which "is the server". With LDAP it is possible to reproduce the available database to other servers on the network too, which you’d use "slurpd" for (which we won’t do).
Mind:
The packages provided on this page have an edited `debian/changelog` as well to hold information about what was changed: Additionally to mentioning the addition of SSL support here, the package names are changed to contain the suffix "subnet". (BTW: Run `date -R` to get the correct date string for the changelog.)
Note:
In previous versions of this document I stated to run `./debian/rules binary` to compile the Deb packages. While this works, the somewhat more official way seems to me to be using `dpkg-buildpackage` instead.
(If for some reason the file `debian/rules` should not be executable, run `chmod +x debian/rules`.)

Invoking `dpkg-buildpackage -b -us -uc` creates the .deb-packages in `~/slapd_woody-source/`, which you should install blindly accepting the default-values presented by Debconf. (As described at aphroland.de, we’ll wipe out the default-stuff and start from scratch on ourselves.)

```
  cd ~/slapd_woody-source/
  dpkg -i slapd_2.0.23-6_i386.deb
          libldap2_2.0.23-6_i386.deb
          libldap2-dev_2.0.23-6_i386.deb
          ldap-utils_2.0.23-6_i386.deb
  [ Get subnet’s self-compiled slapd packages ]
  /etc/init.d/slapd stop
```

In order to prevent the packages to be replaced by the ones from the Debian-repository, set them to HOLD. (Use `dselect` or a command like `"echo "slapd hold" | dpkg --set-selections"` for this.)
But be aware to keep track of possible security-updates for these packages on your own from now on! (Upgrading to the possibly new packages then should be easily possible by running "dpkg -i ..." again. Make sure to have backups of your configuration before as well as to set the packages to HOLD afterwards again.)

Configure OpenLDAP

Wiping out Debian's default configuration and setting up our own one works as follows:

```
  adduser slapd

  chown -R slapd.slapd /etc/ldap
  chmod 770 /etc/ldap
  find /etc/ldap -type f -exec chmod 440 {} \;
  find /etc/ldap -type d -exec chmod 770 {} \;
  chown -R slapd.slapd /var/lib/ldap
  chmod 750 /var/lib/ldap
  rm /var/lib/ldap/*
  chown -R slapd.slapd /var/spool/slurpd
  rm /var/spool/slurpd/*

  cd /etc/ldap/
  mv slapd.conf slapd.conf_DEB-orig
```

This way, the user "slapd" (which we'll use to run the LDAP-server later-on) is the only one who can read the LDAP configuration as well as the database.

Here's the first basic version of our main configuration file, `/etc/ldap/slapd.conf`:

```
######################## /etc/ldap/slapd.conf ########################
# http://homex.subnet.at/~max/ldap/
#
# Basic slapd.conf

include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include         /etc/ldap/schema/misc.schema

schemacheck     on
pidfile         /home_local/slapd/slapd.pid
argsfile        /home_local/slapd/slapd.args
password-hash   {CRYPT}
replogfile      /var/lib/ldap/replog
loglevel        256
database        ldbm

suffix          "dc=subnet,dc=at"
# use "/usr/sbin/slappasswd -h {CRYPT}" to create a rootpw-string below
# (note: if you use the tcsh shell, you will have to use single quotes
#        to surround the {CRYPT}, i.e.: /usr/sbin/slappasswd -h '{CRYPT}')
rootpw          {CRYPT}xxxxxxxxxx
directory       "/var/lib/ldap"
index objectClass eq
lastmod         on

access to attribute=userPassword
        by dn="cn=manager,dc=subnet,dc=at" write
        by anonymous auth
        by * none

access to *
        by dn="cn=manager,dc=subnet,dc=at" write
        by dn="cn=nss,dc=subnet,dc=at" read
        by * auth
#######################################################################
```

Differences to aphroland's description include using {CRYPT}-hashes instead of {MD5}-ones as well as starting the server as root and have it drop his privileges in order to become the user "slapd" as soon as it has bound to the ports 389 and 636. (See below for details.)

Next, besides editing the rootpw-line in your `slapd.conf`, run some more file-system stuff:

```
  # chown slapd.slapd slapd.conf
  # chmod 440 slapd.conf

  # ll
  total 12
  drwxrwx---    2 slapd    slapd        4096 Jun  3 14:38 schema
  -r--r-----    1 slapd    slapd         864 Jun  3 14:41 slapd.conf
  -r--r-----    1 slapd    slapd        1928 Jun  3 14:38 slapd.conf_DEB-orig
```

Database Population

As our database is currently less than empty, we need to populate it.
To be able to test the setup, use a file like the following which holds the basic data to be added. As you've already checked out some general documents on LDAP (haven't you?), you should already know that this file is in "LDIF"-format:

```
dn: dc=subnet,dc=at
objectClass: organization
o: subnet

dn: cn=manager, dc=subnet,dc=at
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: admin
description: LDAP administrator
userPassword: {CRYPT}xxxxxxxxxx

dn: cn=nss, dc=subnet,dc=at
objectClass: organizationalRole
objectClass: simpleSecurityObject
```

```
cn: nss
description: LDAP NSS user for user-lookups
userPassword: {CRYPT}xxxxxxxxxx

dn: ou=People, dc=subnet,dc=at
objectClass: organizationalUnit
ou: People

dn: ou=Group, dc=subnet,dc=at
objectclass: top
objectclass: organizationalUnit
ou: Group

dn: uid=maxldap, ou=People,dc=subnet,dc=at
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
objectClass: organizationalPerson
objectClass: inetLocalMailRecipient
uid: maxldap
cn: Markus LDAP Test User Amersdorfer
sn: Amersdorfer
givenname: Markus LDAP Test User
title: Admin
departmentNumber: IT
mobile: 012-345-6789
postalAddress: AddressLine1$AddressLine2$AddressLine3
telephoneNumber: 1234-567890
facsimileTelephoneNumber: 012-345-6789
userpassword: {CRYPT}xxxxxxxxxx
labeleduri: http://homex.subnet.at/~max/
mail: my.email.address@example.com
mail: my.alternate.email.address@example.com
mailRoutingAddress: my.email.account@mail.server.example.com
loginShell: /bin/bash
uidNumber: 12345
gidNumber: 12345
homeDirectory: /home_local/maxldap/
gecos: maxldap_gecos-field
description: Not Available
localityName: Bellevue

dn: cn=maxldap,ou=Group,dc=subnet,dc=at
objectClass: posixGroup
objectClass: top
cn: maxldap
gidNumber: 12345
```

Don’t forget to run the `"/usr/sbin/slappasswd -h {CRYPT}"`-command to create password-hashes for the users with {CRYPT}-entries listed in the .ldif-file.
(Again, if you use the tcsh shell, this might produce an error stating something like "Password generation failed for scheme CRYPT: scheme not recognized". To work around this, surround the parameter with single quotes, i.e. run the following command instead: `"/usr/sbin/slappasswd -h '{CRYPT}'"`. Also see this OpenLDAP mailing-list article on this issue.

Many thanks to Martin B. Smith for pointing this out!)

The normal user `uid=maxldap`:
Mind the naming pattern used here for the normal user "maxldap": its distinguished name `"dn:"` (which is unique within the global LDAP namespace) is constructed by using the user’s `"uid=maxldap"` attribute (which equals to the Linux user’s login name, the corresponding Linux user’s UID can be found as LDAP’s attribute `"uidNumber"`) prefixing the tree `"ou=People,dc=subnet,dc=at"`.
This places the user in the organizational unit `"ou=People"` of "subnet.at" (`"dc=subnet,dc=at"`).
Some sites use the users’ common names (`"cn:"`) instead of the uid’s to differentiate between single LDAP entries (users). While it basically boils down to a matter of taste on the one hand (whether you prefer `"uid=maxldap,ou=People,..."` over `"cn=Markus Amersdorfer,ou=People,..."` or the other way round), on the other hand it’s *definitely* better to use `"uid="` here. The simple reason is that both the MigrationTools (see section Migrate Your Linux Users below) and Samba (see section Samba 2.2.x and LDAP below) use this pattern. You’ll save yourself a lot of time if you stick with `"uid=,ou=,dc=,dc="`.

The special user `cn=nss`:
With our current ACLs, nobody except cn=manager and cn=nss can perform standard *read* functionality on our LDAP tree. Nevertheless, to be able to become a user (e.g. using "su user") or to get information about the user ("finger user"), the tree must be readable, at least to the Name Switch Service (NSS) (see section NSS: Name Service Switch below).
It depends on your situation to either set read-rights for everyone to ou=People, or to use this cn=nss user so that NSS can lookup the users. I’ll describe the latter scenario.

You can and should add the data above to the (currently not running) OpenLDAP database by executing:

```
# su - slapd
$ /usr/sbin/slapadd -l /etc/ldap/basics-subnet.ldif
```

Using `"slapcat"` you get the database’s current contents without having to perform `"ldapsearch"` or similar. This can be useful for debugging processes.

Start OpenLDAP

You can now - being root - start the OpenLDAP server (without having it disappear into daemon-mode):

```
# /usr/sbin/slapd -u slapd -h ldap://0.0.0.0/ -d 255
```

This starts the OpenLDAP server "slapd" initially as root, binds to the corresponding port (TCP/389) on all local interfaces, drops root privileges by becoming the user "slapd" and presents you with ’live’ debugging information on your screen.
(Hint: Browse through this stuff to get a feeling for OpenLDAPs debugging information and error messages.)

Having the OpenLDAP server up and running, we can deal with the client-side now...

NSS: Name Service Switch

NSS: Introduction

On Linux (and some other UNIX-flavours), accessing the users-database is not just looking up the passwd/shadow/a.s.o. files. Nowadays, most applications use library calls to get user information or accomplish user authentication.
While the PAM system (Pluggable Authentication Module, see below) is used to accomplish a user’s authentication (i.e. checking if provided login and password are correct, accomplish some other (stackable and thus highly configurable) tasks and finally decide for example whether the user may login or not), the Name Service Switch is a service which provides you with a user/group/a.s.o. listing. To get your local machine’s or network’s listing, just run `"getent passwd"`.

The first task now is to set up the NSS correctly to query the OpenLDAP server additionally to the local passwd-files (and/or the already used NIS). This is done by installing the package "libnss-ldap" and configuring the nss-processes to use it.

This description goes for both your network-clients as well as the LDAP-server itself (as we want the server’s Linux-system too to know the users and other information stored using OpenLDAP)!

NSS: Installation (with SSL capable packages)

In order to have any traffic between the clients and server be encrypted, we again need to compile the packages ourselves to support SSL. (The actual configuration of encrypted communication can be found later in the document.)

```
cd ~
mkdir libnss-ldap_woody-source
cd libnss-ldap_woody-source

apt-get source libnss-ldap
cd libnss-ldap-186
vi debian/rules
  --> and replace --disable-ssl with --enable-ssl
[ vi debian/changelog ]

dpkg-buildpackage -b -us -uc

dpkg -i libnss-ldap_186-1_i386.deb
[ Get subnet’s self-compiled libnss-ldap package ]
echo "libnss-ldap hold" | dpkg --set-selections

mv /etc/libnss-ldap.conf /etc/libnss-ldap.conf_DEB-orig
```

The final two commands install the new libnss-ldap-package and set it to HOLD status.
But be aware to keep track of possible security-updates for these packages on your own from now on!
(Upgrading to the possibly new packages then should be easily possible by running "dpkg -i ..." again.
Make sure to have backups of your configuration before as well as to set the packages to HOLD afterwards again.)

Mind: The manual page for `libnss-ldap.conf` does not specify all of the module’s options. In order to be able to browse through the capabilites later (and perhaps activate some of them), we made a backup of Debian’s original and (throughout the file itself) well-documented libnss-ldap.conf-file.

Once the package is installed, use the following `/etc/libnss-ldap.conf` file to configure the new functionality correctly:

```
######################### /etc/libnss-ldap.conf ########################
# http://homex.subnet.at/~max/ldap/

host ldap.subnet.at
base ou=People,dc=subnet,dc=at
uri ldap://ldap.subnet.at/
ldap_version 3

binddn cn=nss,dc=subnet,dc=at
bindpw the_one_you_set_above_in_the_ldif-file__as-plaintext

nss_base_passwd ou=People,dc=subnet,dc=at
nss_base_group  ou=Group,dc=subnet,dc=at
#######################################################################
```

The `bindpw`-entry is the password for the NSS-user (cn=nss,dc=subnet,dc=at) you created above when populating the LDAP database. The password has to be stated as plaintext here, do *not* use the {CRYPT}-hash.

Now, include the LDAP NSS module in the system lookups by editing `/etc/nsswitch.conf`:

```
  passwd:         ldap compat
  group:          ldap compat
  shadow:         ldap compat
```

This way, lookups for passwd, group and shadow try LDAP first (`"ldap"`) and NIS and the local files next (`"compat"`).
(If a user is listed both locally *and* in LDAP, it will also show up *twice* in the output. This feature is used in the setup described here to have the user "root" both be served from LDAP and - as a fallback in case of LDAP wasn’t reachable - have it stored locally. See section PAM: The user "root" and other system UIDs below for details.)

It should be possible to lookup the LDAP-user "maxldap" using `finger` or `getent` now:

```
  # finger maxldap
  Login: maxldap                          Name: maxldap_gecos-field
  Directory: /home_local/maxldap/       Shell: /bin/bash
  Last login Mon Jun  2 16:53 (CEST) on pts/1 from some-client.subnet.at
  No mail.
  No Plan.
```

NSCD and /etc/libnss-ldap.conf

NSCD is "a daemon which handles passwd, group and host lookups for running programs and caches the results for the next query". While this makes NSS-lookups faster, it also might lead to the situation where it might take some time for an update of user-data to reach all clients. (Or is there some "pushing"- or any other mechanism that solves this?)

Anyway, installing "nscd" might definitely be a good idea from the security point of view: The above mentioned `/etc/libnss-ldap.conf` file holds some clear-text information necessary to be able to perform NSS-lookups. In order to prevent the users from not knowing who they are - resulting in funny situations such as the prompt saying "i have no name!" instead of the actual user's login-name - this file has to be world-readable. But then, this means that everybody knows about the credentials of the "cn=nss" user and can do everything this special user can (which depends on the access-lists of the LDAP server).

Though I haven't tried it yet, NSCD can help you solve this issue: Just install it and set the file-access-rights for `/etc/libnss-ldap.conf` to "600" (owned by root).
In this setup, the corresponding library-request executed with the user's rights should be handled by NSCD, which in turn runs with root-privileges (I guess, at least), and thus can read the credentials from the config-file and perform the corresponding DB-lookup.

PAM: Pluggable Authentication Module

PAM: Introduction

As mentioned above, user lookups are seperated from user authentication on Linux systems. While the first is covered by NSS, the second is usually dealt with using PAM nowadays.

Basically, it's the same process here for the package "libpam-ldap" as it was with libnss-ldap above: recompilation with SSL enabled and installation of the new package.
Nevertheless, the Debian Woody package has a special patch applied to be able to use *filters* when checking whether a user is allowed to login or not. We'll use this feature to be able to allow users to login to some specific workstations and block access on the network's other workstations (see below for details).
Unfortunately, this filter patch has a bug, which means we'll need to install a patched version of libpam-ldap, compiled ourselves to support SSL. This patched version is available from Christof Meerwalds Debian section.

PAM: Clients vs. Server Configuration

As with NSS, we want the LDAP-server itself too to be able to use the LDAP-based users for authentication. Thus, basically, the same configuration applies to the server as it does to the client machines (= Linux/Unix stations not running the LDAP server but just querying it for user lookups and authentication.)

Nevertheless, you have several options here now, depending on your needs and wishes. It’s all about the user "root" and about changing user passwords.

1. You can configure a machine so that there is no almighty root anymore concerning the users. If you do so, root can not add users - root can’t even change the users’ passwords. This can be *bad* (additional "overhead", necessary change of habits) or *good* (the system administrator "root" is not responsible/able to change the users’ passwords, this can/must be done by someone else) - depending on your needs. You can keep the system-administrator (responsible for a machine’s uptime) seperated from the users administration. (This is not totally correct: root *can* change a user’s password, but it has to *know* the user’s old password to able to set a new one. This is the same behaviour as if the users theirselves would change their passwords.)
2. You can configure a machine to behave as if users were "installed" locally (in passwd|shadow) so that root can change them and their passwords.

I’ll describe a setup here where root *can* change any user’s password, but only on the machine running the OpenLDAP server. Reasons are that I want root to be able to change the passwords by simply running "passwd $user" (without having to know the user’s old password). Nevertheless, this ability includes the need for the password for "cn=manager,dc=,dc=" to be stored on such a machine locally in a file, additionally it has to be *in plaintext*. This doesn’t seem to be easy to administer (especially in the case where the manager’s password changes) on the one hand, and it doesn’t seem to be very secure either for obvious reasons on the other hand.

The only difference lies in the file `"/etc/pam_ldap.conf"` using the option `"rootbinddn"` on the server and `"binddn"` on all other machines. Furthermore, the server needs the file `"/etc/ldap.secret"` which holds the manager-user’s password in plaintext (with access rights "600", owned by "root").

Most of the steps following are the same for all machines, no matter which configuration you chose. If not stated otherwise, it’s the same for both possible setups.

PAM: Installation (with SSL capable packages)

```
Add to /etc/apt/sources.list:
  # Patched libpam-ldap for Woody (http://cmeerw.org/debian/)
  deb-src http://cmeerw.org/files/debian woody libpam-ldap

Run:
  cd ~
  mkdir libpam-ldap_cmeerw-source
  cd libpam-ldap_cmeerw-source

  apt-get update
  apt-get source libpam-ldap

  vi debian/rules
    --> and replace --disable-ssl with --enable-ssl
  [ vi debian/changelog ]

  dpkg-buildpackage -b -us -uc

  cd ..
  dpkg -i libpam-ldap_140-1cmeerw_i386.deb
  [ Get subnet’s self-compiled libpam-ldap package ]
  echo "libpam-ldap hold" | dpkg --set-selections

  mv /etc/pam_ldap.conf /etc/pam_ldap.conf_DEB-orig
```

Again after installing the package, we set its status to HOLD.
But be aware to keep track of possible security-updates for these packages on your own from now on! (Upgrading to the possibly new packages then should be easily possible by running "dpkg -i ..." again. Make sure to have backups of your configuration before as well as to set the packages to HOLD afterwards again.)

Though we’ll use a setup without SSL and without host-specific access controls for the moment, using the patched package and recompiling it with our modifications we’re ready for these things to come later on.
Mind: The manual page for `pam_ldap.conf` does not specify all of the module’s options. In order to be able to browse through the capabilites later (and perhaps activate some of them), we made a backup of Debian’s original and (throughout the file itself) well-documented pam_ldap.conf-file.

Next, on all the clients (where root is not able to change the users’ passwords), configure the new PAM module using this `/etc/pam_ldap.conf`:

```
########################## /etc/pam_ldap.conf #########################
# http://homex.subnet.at/~max/ldap/
#
# pam_ldap.conf for all client machines

host ldap.subnet.at
base dc=subnet,dc=at
uri ldap://ldap.subnet.at/
ldap_version 3

binddn cn=nss,dc=subnet,dc=at
bindpw the_one_you_set_above_in_the_ldif-file__as-plaintext

pam_password crypt
######################################################################
```

On the server (or on all machines where you want root to be able to change the users’ passwords), **first**, use this `/etc/pam_ldap.conf`:

```
########################## /etc/pam_ldap.conf #########################
# http://homex.subnet.at/~max/ldap/
#
# pam_ldap.conf for the server (where root can change user passwords)

host ldap.subnet.at
base dc=subnet,dc=at
uri ldap://ldap.subnet.at/
ldap_version 3

rootbinddn cn=manager,dc=subnet,dc=at
```

```
# don’t forget /etc/ldap.secret

pam_password crypt
#######################################################################
```

and **second** don’t forget to create the file `/etc/ldap.secret` with access rights "600" and owned by "root" (i.e. only root can read the file) which holds the plaintext-password for LDAP’s "cn=manager,...". (While, if I’m correct, I didn’t have such in my setup originally: according to Setting up LDAP for use with Samba, there has to be a blank second line in this file.)

Now that we have the PAM module configured, we need to include it into the PAM process: We’ll check out the modifications to be able to log in using `ssh` and `su`.
Most PAM-aware applications have their own PAM-stack they use. Debian stores these configuration-files in `/etc/pam.d/`.
Here is `/etc/pam.d/ssh`:

```
########################### /etc/pam.d/ssh ############################
# http://homex.subnet.at/~max/ldap/

auth       required     pam_env.so
# Woody’s SSHD checks for /etc/nologin automatically,
# so there’s no need for pam_nologin in /etc/pam.d/ssh.
#auth       required     pam_nologin.so
auth       sufficient   pam_ldap.so
auth       required     pam_unix.so

account    sufficient   pam_ldap.so
account    required     pam_unix.so

session    sufficient   pam_ldap.so
session    required     pam_unix.so
session    optional     pam_lastlog.so # [1]
session    optional     pam_motd.so # [1]
session    optional     pam_mail.so standard noenv # [1]
session    required     pam_limits.so

password   sufficient   pam_ldap.so
password   required     pam_unix.so
#######################################################################
```

The changes to the original file are:

- The addition of the "pam_ldap.so"-lines.
- Re-ordering the `"auth"`-lines. This is necessary as once pam_ldap authenticates successfully, no other `"auth"`-lines are used due to pam_ldap’s `"sufficient"` attribute. In order to have pam_env and similar be used, one has to place them *before* pam_ldap.
  ("sufficient" can not be replaced with "required" here, as we want to be able to fall back to the (local) pam_unix in case pam_ldap does not authenticate successfully. This might be the case if for example the LDAP server could not be contacted for some reason.)
  (The "session" section might have to be re-ordered too, of course.)

Here is `/etc/pam.d/su` as another example:

```
########################### /etc/pam.d/su #############################
# http://homex.subnet.at/~max/ldap/

auth       sufficient pam_rootok.so
auth       sufficient pam_ldap.so
auth       required   pam_unix.so use_first_pass

account    sufficient pam_ldap.so
account    required   pam_unix.so

session    sufficient pam_ldap.so
session    required   pam_unix.so
#######################################################################
```

The changes to the original file are:

- Again, the "pam_ldap.so"-lines were added.
- The option "use_first_pass" is passed to pam_unix.so. This way the pam_unix-module uses the password which was provided to "auth [...] pam_ldap". Otherwise, one would have to enter the password twice for users not in LDAP. (For some reason, "use_first_pass" is not needed in pam.d/ssh above...)

Things to **take care of**:

- Do not forget to edit the files `/etc/pam.d/other` and `/etc/pam.d/login` similarly. The first is used if some application without a specific service-file in /etc/pam.d/ uses PAM-calls, the latter is used if somebody logs in locally (not via SSH or something like that).
- I haven’t figure out yet when it is necessary to restart e.g. SSH after changing its PAM-stack file. Best thing is to restart it everytime to be sure changes are activated.
- Should you re-order your PAM stack, do not use "use_first_pass" on the first password-querying auth-module of your service-configuration, or this module can’t authenticate the users anymore.
- Even if all your users (including "root") are stored in the LDAP database, it would not be a good idea to remove the pam_unix.so module. In case your LDAP server wouldn’t be reachable for whatever reason, you would not be able to login to any of your machines - not even as root (and that’s exactly what you might *need* to do in this case to debug or administer your network).

Logging in via SSH or su’ing to the LDAP-user "maxldap" should work now:

```
  $ ssh maxldap@ldap.subnet.at
  maxldap@ldap.subnet.at’s password:
  Last login: Tue Jun  3 15:11:30 2003 from some-client
  maxldap@ldap:~$
```

PAM: Passwords: Facts

CRYPT vs. MD5:
The good old shadow file typically stores passwords as hashes using the "crypt" algorithm. More up to date systems often use "md5" somewhere in the process of hashing the password. (These

"md5"-passwords can be distinguished from the "crypt"-only ones by starting with "$1$".)
The Mandrake-based document on mandrakesecure.net (see section External Resources) describes a way to use exactly the newer MD5-based approach with your LDAP database. Unfortunately, either Debian doesn't support it or I simply couldn't get this thing to work. (It seems I managed to have passwords being created using MD5 and have them stored in the LDAP database this way, but unfortunately, I couldn't use these hashes. I never got a user to authenticate successfully.) Being well known for my paranoia ;), I really would have liked the MD5-thing as it creates longer password hashes and uses passwords with more than just 8 characters. Anyway, good old CRYPT will (have to) suffice...

BTW: One reason for *even me* thinking that "crypt" really is secure enough is that the password is *never* sent in any way in plaintext over the network, as every traffic between clients and server is secured using SSL (see below). Even a local "root" user can't see the password-hash by executing `"getent shadow"`. Only the "LDAP Manager" is allowed to see the hash.
The only (really bad) thing you should remember is: "Crypt"-passwords have a maximum length of 8 characters!

PAM: Passwords: How To Change Them

Here is my `/etc/pam.d/passwd`. Using it, the users can change their passwords on their own from the command-line by simply executing `passwd`.

```
########################### /etc/pam.d/passwd ###########################
# http://homex.subnet.at/~max/ldap/

password        sufficient      pam_ldap.so
password        required        pam_unix.so nullok obscure min=4 max=8
##########################################################################
```

I tried several versions here. Unfortunately, this is the only one I found where both root and the users themselves (both LDAP-based users and local-only ones in passwd|shadow) can change their passwords. (Especially, I could not get the stuff working using `pam_cracklib.so` or using `use_first_pass` with `pam_unix.so`. This would have been nice!)

Oh, and of course, don't forget the setup of `/etc/pam_ldap.conf` concerning root's non/ability to change users' passwords explained in section PAM Clients vs. Server Configuration above, but you've already read it anyway, haven't you?

PAM: The user "root" and other system UIDs

(You actually don't have to worry about that just yet, just bare in mind you will have to decide later (when actually doing all the migration stuff in section Migrate Your Linux Users below) what to do with all the system accounts and with "root". Nevertheless, as it's about user accounts, I think the general knowledge about it belongs to PAM and thus here.)

I definitly recommend to keep your system UIDs (Debian currently treats the UIDs 1-999 as such) only locally on every machine, do not migrate them into LDAP!
It's up to you what you do with "root". If you prefer to have one root account for all machines (as I do), migrate "root" to LDAP. Otherwise, just treat it as a system account and do not migrate it.

The system UIDs

If you need *reasons for not migrating them to LDAP*:
Different services on different servers/machines will need different system users to be present (e.g. the user "mysql" should only be available on a machine hosting a MySQL server).
One day, you will upgrade your machines from Woody to Sarge. It can (and probably will) happen that according to a new policy some system accounts changed their UID. Upgrading just only one server will not work in this context, as the user "root" should not be able to simply change those LDAP values. Even if it worked properly for one server, what about the second? (Well, ok, everything might turn out to work somehow, but IMHO you really don't want it this way :) ...)
And of course (above all), if you want to serve your users to client machines which do *not* run Debian Woody and thus use a different system accounts scheme, you'll have problems on your hands.

`UID 0`: "root"

If you want to migrate the user "root" into LDAP (like I did), simply migrate it as described below in section Migrate Your Linux Users.

But be sure to *also have "root" locally* in `/etc/(passwd|shadow|group)`. If for some reason your LDAP server was not reachable, you still can log in to your (LDAP)-client machines using the local "root" account.
With the setup explained here, logging in as "root" will use the LDAP-account by default. Only if the LDAP server is not accessible, you will automatically fall back to the local user/password from the flat files. Try this yourself to make sure your machines behave properly. (Mind the order `"ldap compat"` in `/etc/nsswitch.conf`, described in section NSS: Installation (with SSL capable packages).)

ToDo - verify this again:
Oh, and before I forget and because it differs from standard Linux behaviour: if "root" tries to change its LDAP-account password using `"passwd"`, it also needs to know its old password, just as any other user does!
(If you can't remember the password anymore, you'll need to change it directly in the LDAP database, but that's probably what you will do anyway, won't you? :) ...)
Additionally, this also depends on section PAM Clients vs. Server Configuration, actually...

Host Specific Access

Introduction

Up to now, if a user shall has access to one host, they actually have access to *all* hosts on the network.

As described in the Mandrake based LDAP-article already mentioned in the resources section, one can define one or more `"host"`-attributes (host is part of "objectClass: account") in the user's LDAP-entry, specifying that this user is allowed to login to the listed host(s).
Debian offers several ways to achieve our goal, from "simple" to "advanced":

Approach 1: pam_check_host_attr

For every host the user shall be able to login, add an attribute similar to `"host: allowed-host.mydomain.net"`.

While Mandrake seems to have merged "libnss-ldap.conf" and "pam_ldap.conf" to on single "ldap.conf" file, you already know that Debian uses the split-up approach.
The "pam_check_host_attr" option can be found in `"/etc/pam_ldap.conf"`, you can add the following there:

```
/etc/pam_ldap.conf
  [...]
  pam_check_host_attr yes
```

But be careful: As the comment in Debian's original pam_ldap.conf indicates, you'll need pam_ldap.so to be "configured for account management (authorization)". This means, that in the /etc/pam.d/<service> files, you'll have to replace `"account sufficient pam_ldap.so"` with `"account required pam_ldap.so"`. (Otherwise, despite the message "Access denied..." the user will be granted access to the host.)

Approach 2: Filters

The more powerful way is the following one:
Debian's package `libpam-ldap` has a "filtering" patch applied. This way it is possible to accompany the LDAP PAM module with some filtering-rules which must match in order for the module to return successfully and allow the authentication-process to proceed.
As already mentioned in section PAM: Pluggable Authentication Module, the standard Woody package's filter patch has a bug which needs to be fixed. As we've installed the corrected version from cmeerw.org, we already have a working version.

So here's how to do it:
For every host that the user shall be able to login, add an attribute similar to `"host: allowed-host.mydomain.net"`. If the user is allowed to login to *all* hosts, simply add `"host: *"` to the LDAP entry.
Next thing is to adapt the PAM stack: This can either be done by editing one (or more) specific service's config file in `/etc/pam.d/`, or by editing `/etc/pam_ldap.conf` (which has influence on all services at once):

```
For single services only (e.g. /etc/pam.d/ssh):
  #auth       sufficient   pam_ldap.so
  auth       sufficient   pam_ldap.so filter=|(host=this-host.mydomain.net)(host=\*)

For all services at once (/etc/pam_ldap.conf):
  pam_filter |(host=this-host.mydomain.net)(host=\*)
```

Only if the user's LDAP entry contains `"host: this-host.mydomain.net"` or `"host: *"`, they are allowed to login.

Things to **take care of**:

- Mind the backslash `"\"` in the filter `"host=\*"`. If you miss it (ending up with "host=*"), you’ll allow access to every user who has at least one "host"-attribute (with **any** content)!
- Be sure to add `"host: *"` for user "root" (if it is served via LDAP) if you want it to be able to login via SSH!

Add-On:
Of course this setup can be extended for example by using self-defined attributes representing groups of hosts (e.g. "my-hosts: workstations") and adapting the PAM module’s filters.
With a little imagination it could perhaps also be possible to assign a user different shells on different hosts!? I don’t know yet...
It’s all up to you. :)

SSL Encryption

Now that we have a running LDAP-server which provides user-information and also have clients authenticating using these LDAP-users, let’s move on and make everything safer: Up to now, any traffic between the server and the clients was unencrypted. Let’s activate the SSL-encryption our packages already are capable of (as we recompiled them ourselves).

You can find a description of how to create an SSL certificate here: HOWTO Create an SSL Certificate.

Once you have your signed certificate, you need to configure your OpenLDAP server to use it:

```
Add to your /etc/ldap/slapd.conf:
  [ ... loglevel xxx ]

  TLSCipherSuite HIGH:MEDIUM:+SSLv2
  TLSCertificateFile /etc/ldap/server.cert
  TLSCertificateKeyFile /etc/ldap/server.key
  TLSCACertificateFile /etc/ldap/ca.cert
  TLSVerifyClient 0

  [ ... database ldbm ]

Start the LDAP server with the following command:
  # /usr/sbin/slapd -u slapd -h ’ldap://0.0.0.0/ ldaps://0.0.0.0/’ -d 1
```

You can *test the server*’s SSL capabilities (of course the client you are executing the second command needs the "ldap-utils"-package with SSL-support compiled into it!):

```
  ldapsearch -b "ou=People,dc=subnet,dc=at" -LLL -D "cn=manager,dc=subnet,dc=at" \
    -H "ldap://ldap.subnet.at/" -W -x "(uid=maxldap)"
  ldapsearch -b "ou=People,dc=subnet,dc=at" -LLL -D "cn=manager,dc=subnet,dc=at" \
    -H "ldaps://ldap.subnet.at/" -W -x "(uid=maxldap)"
```

Both commands, with and without SSL encryption, should return the entry for the user "maxldap".

Activate SSL Encryption for the Clients’ Queries

Next step is to adapt the clients’ setup:
Having already installed our re-compiled library-packages, this is as easy as changing `"uri ldap://ldap.subnet.at/"` to `"uri ldaps://ldap.subnet.at/">` in both `/etc/libnss-ldap.conf` and `/etc/pam_ldap.conf`.

Add-On:
To test if your clients really *are* communicating with the server using an encrypted connection, make user-queries and logins with both settings ldap:// and ldaps:// and a concurrently running `"tcpdump -X host ldapserver"`. This tcpdump-command shows you in ASCII the transmitted data. While using ldap:// you should be able to find some cleartext in the data garbage, after switching to ldaps:// you should only see ... well, lot’s of stuff, but no plaintext information.

OpenLDAP Startup Script

Now that we have everything set up correctly concerning the connections (including SSL-support), we should look at the changes necessary to `/etc/init.d/slapd` to have the OpenLDAP server started correctly every time the machine boots - all we have to do is to change one line:

```
# start-stop-daemon --start --quiet --pidfile "$pf" --exec /usr/sbin/slapd
start-stop-daemon --start --quiet --pidfile "$pf" --exec /usr/sbin/slapd -- -u slapd \
  -h ’ldap://0.0.0.0/ ldaps://0.0.0.0/’
```

This way, we have the process’ user changed to "slapd" as well as the server listen for ldap-traffic on port 389 and ldaps-traffic on port 636.

"slapd" now logs to `/var/log/debug`.

So Far So Good, Part 1

So far, on the server side we have accomplished to set up and populate the OpenLDAP server.
Next, on the client side (this means every machine querying OpenLDAP, this most probably also includes the "server" where OpenLDAP is running on) we have configured the Linux clients to use it, they can look up the users and use those to log into machines. Communication between the clients and the server is secured using SSL encryption.
All you have to set up on each "client" is:

- Install our self-compiled libnss-ldap package.
- Install our self-compiled libpam-ldap package based on cmeerw.org.
- Edit `/etc/nsswitch.conf`, `/etc/libnss-ldap.conf` and `/etc/lib_pam.conf (clients)` respectively `/etc/lib_pam.conf (server)` and `/etc/ldap.secret (server)`.
- Edit the corresponding service files in `/etc/pam.d/`, for example /etc/pam.d/ssh.

Migrate Your Linux Users

Migrate Linux: Prerequisites

Of course you want to migrate your current Linux users from passwd/shadow to the new OpenLDAP server. Debian offers help here by providing the MigrationTools from padl.com as the Debian package `migrationtools`, so you could `apt-get install` this package.
Unfortunately, the package in Debian Woody (version 40-1) is rather buggy: `"migrate_base.pl"` forgot an "s" with "dc: subnet", `"migrate_group.pl"` didn't produce any output and `"migrate_passwd.pl"` produced junk values for the attributes "cn", "givenname" an "sn". Furthermore, the latter also uses different letter cases in a newer version than Woody's one does (e.g. "givenName" instead of "givenname").
The newer version of Debian Sarge (currently version 44-6) definitely works better, but also still produces junk values for the attributes "cn".

In order to get a working package you should download the original version of the MigrationTools from padl.com. I used version 44, which works fine for me.

In either case, you'll have to execute the scripts on the machine which currently holds your users already, in my case this was our NIS server (which was a different machine than the upcoming LDAP server), so install the migration-package there.

After exploding the tar-ball, edit `/usr/local/MigrationTools-44/migrate_common.ph` and adapt the following variables:

```
$DEFAULT_MAIL_DOMAIN = "subnet.at";
$DEFAULT_BASE = "dc=subnet,dc=at";
$DEFAULT_MAIL_HOST = "mail.subnet.at";
$EXTENDED_SCHEMA = 1;
```

Citing the initial Mandrake-LDAP document on this one (with "localisation"): *"This sets some defaults for the migrated data. Here we set the default mail domain, in this case "subnet.at" which will assign all users a default email address of "user@subnet.at". The default base is "dc=subnet,dc=at" which should be identical to the suffix defined in slapd.conf. The default mail host is the SMTP server used to send mail, in this case "mail.subnet.at". The extended schema is set to 1 to support more general object classes."*

03-08-12:
As Buchan Milne pointed out in a mail on the Samba mailing-list and his Howto Implementing disconnected authentication and PDC/BDC relationships using Samba and OpenLDAP, changing those values in migrate_common.ph is not necessary: Setting and exporting according environment variables (for example `export LDAP_DEFAULT_MAIL_DOMAIN="subnet.at"`) works too and would survive an eventual upgrade (though of course *migrating users* will probably be performed only once).

It's possible to migrate nearly all data to LDAP (including `/etc/(hosts|protocols|services)` etc.), nevertheless, the only things we'll use LDAP for are users and groups. ("hosts" would be good to migrate too, but we have a local DNS server running, so no need for this here.)
Furthermore, as already mentioned above in section PAM: The user "root" and other system UIDs, you'll have to exclude your system accounts (Debian Woody uses the UIDs 1-999 for this) and decide on what to

do with "root" (UID 0). Especially keep in mind when migrating both groups and users to delete all system accounts from the .ldif-file created by the migration scripts!
ToDo: It should be possible to do this using some /sed/awk/grep/etc/ scripts.

Migrate Linux: The Scripts

The migration scripts are located in `/usr/local/MigrationTools-44/` (or wherever you saved them to):
`migrate_base.pl` creates the LDAP tree's base structure.
The `migrate_all_*` migrate everything by simply calling the single perl scripts.
As I've already mentioned, the only things we'll migrate are users and groups, and of course we'll check out the base structure, so let's start with this one:

migrate_base.pl

```
  cd /usr/share/migrationtools/
  ./migrate_base.pl > base.ldif
```

The only entries which are or might become interesting for us are the following base.ldif:

```
dn: dc=subnet,dc=at
dc: subnet
objectClass: top
objectClass: domain
objectClass: domainRelatedObject
associatedDomain: subnet.at

dn: ou=People,dc=subnet,dc=at
ou: People
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: subnet.at

dn: ou=Group,dc=subnet,dc=at
ou: Group
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: subnet.at
```

(Here is the original base.ldif file with all the additional entries.)

Well, we basically do have this base structure already. The only difference is that we are missing "objectClass: domainRelatedObject" and its associated attribute.
So we won't do anything here at the moment.

migrate_group.pl

Here is the minimized output from `./migrate_group.pl /etc/group group.ldif`:

```
dn: cn=max,ou=Group,dc=subnet,dc=at
objectClass: posixGroup
objectClass: top
cn: max
userPassword: {crypt}x
gidNumber: 1000

dn: cn=users,ou=Group,dc=subnet,dc=at
objectClass: posixGroup
objectClass: top
cn: users
userPassword: {crypt}x
gidNumber: 100

dn: cn=nogroup,ou=Group,dc=subnet,dc=at
objectClass: posixGroup
objectClass: top
cn: nogroup
userPassword: {crypt}x
gidNumber: 65534
```

This information can now be added to the LDAP server by executing something like the following:

```
  ldapadd -H ldap://ldap.subnet.at/ -D "cn=manager,dc=subnet,dc=at" -x -W -f $FILE
```

BTW: The main purpose of a group is to hold several users :).
An entry in `/etc/group` like `"somegrp:x:12345:userone,usertwo"` can be accomplished by adding a `"memberUid"` attribute for each user to the group's LDAP entry: `"memberUid: userone"` and `"memberUid: usertwo"`.

migrate_passwd.pl

```
  ./migrate_passwd.pl /etc/passwd passwd.ldif

  # Restrict access as this file holds all passwords:
  chmod 600 passwd.ldif
```

If you don't have any user passwords in your ldif file, try the command with an explicit environment variable set: `ETC_SHADOW=/etc/shadow ./migrate_passwd.pl /etc/passwd passwd.ldif`. This should ensure that the script find the /etc/shadow file, which most probably holds your passwords.

Debian Woody's schema files *do not* provide the objectClass "mailRecipient". Instead, it is called "inetLocalMailRecipient". Additionally, we didn't include the Kerberos schema in our slapd.conf, so we have to remove the corresponding objectClass. You can simply adapt all the entries by executing "sed" like the following:

```
  sed s/mailRecipient/inetLocalMailRecipient/g passwd.ldif | \
    sed '/^objectClass: kerberosSecurityObject$/d' | sed '/^krbName: /d' \
    > passwd.ldif_corrected
  chmod 600 passwd.ldif_corrected
```

Here is the corresponding passwd.ldif file (again with one user only, the corrected objectClass and other values):

```
dn: uid=max,ou=People,dc=subnet,dc=at
uid: max
cn: Markus Amersdorfer
givenname: Markus
sn: Amersdorfer
mail: THEUSERNAME@subnet.at
mailRoutingAddress: THEUSERNAME@mail.subnet.at
mailHost: mail.subnet.at
objectClass: inetLocalMailRecipient
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}$1$_my-password-hash-from-/etc/shadow
shadowLastChange: 12174
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /data/home/max
gecos: Markus Amersdorfer,,,
```

This information can again be added to the LDAP server by executing something like the following:

```
  ldapadd -H ldap://ldap.subnet.at/ -D "cn=manager,dc=subnet,dc=at" -x -W -f passwd.ldif_corrected
  Enter LDAP Password:
  adding new entry "uid=max,ou=People,dc=subnet,dc=at"
```

Samba 2.2.x and LDAP

Samba: Introduction

Now that we have a working LDAP server to hold all our Linux-users, it would be useful to have the network's Windows machines use this database too, of course.
As no flavour of Windows (AFAIK) can access the LDAP database directly, the Windows machines (both servers and clients) will have to be part of a domain which is controlled by a Samba PDC. Thus, the Windows machines will allow access based on what the (Samba) PDC says which again makes its decisions on what the users in the LDAP database look like.
(This means, all Windows/Samba clients query the Samba-PDC. The Samba-PDC queries LDAP for entries which are "objectClass: sambaAccount".)
03-08-15: See user comment from Aug 14, 2003.

Oh, and by the way, remember that this HOWTO-document uses Debian Woody as the underlying distribution, so "Samba" refers to "Samba 2.2.3a" currently.
(Infos on Samba 3.x are as welcome and will be included here as any other feedback!)

Another add-on: You only have to install this "special LDAP-Samba version" on the machine which performs as your network’s PDC. All other Samba machines should be configured to use this one Samba PDC as their "oracle" for user authentication. (You’ll use options like `"security = DOMAIN"` and `"password server = MYPDC"` to accomplish this.)

Samba: Installation and Setup

Samba itself (if it’s not configured to query some other server for user-authentication) can *either* use the `smbpasswd` file to hold its users *or* use an LDAP server to accomplish this. It’s *not* possible to do both at the same time, you have to decide at compile time.
The default behaviour is to use the flat smbpasswd file. Debian Woody’s Samba packages of course defaults to the default in this case. Conclusion: Recompilation is necessary.

```
cd ~
mkdir samba-source
cd samba-source
apt-get source samba
cd samba-2.2.3a
vi debian/rules
  --> add "--with-ldapsam \" just before "--with-msdfs"
[ vi debian/changelog ]

dpkg-buildpackage -b -us -uc
  [ Due to missing build-dependencies:
      apt-get install libreadline4-dev libcupsys2-dev
        [with extra-packages "libcupsys2 libjpeg62 libtiff3g"]
      dpkg-buildpackage -b -us -uc
  ]
cd ..
dpkg -i samba-common_2.2.3a-13subnet_i386.deb
        samba_2.2.3a-13subnet_i386.deb
        smbclient_2.2.3a-13subnet_i386.deb
        smbfs_2.2.3a-13subnet_i386.deb
        samba-doc_2.2.3a-13subnet_all.deb

[ Currently now packages available, please build them yourself ... ]
```

Mind:
In order for Samba to support ACLs, I also added `"--with-acl-support"` and changed in `debian/config.cache` the value `"ac_cv_header_sys_acl_h=${ac_cv_header_sys_acl_h=no}"` to `"ac_cv_header_sys_acl_h=${ac_cv_header_sys_acl_h=yes}"`.

Again, `"dpkg-buildpackage"` might not compile the packages at the first time you run it due to some missing build-dependencies. `apt-get install` the mentioned packages and run `"dpkg-buildpackage"` again in this case.
Again, set the packages to HOLD using `dselect` or something like `"echo "samba-common`

hold" | dpkg --set-selections".
But be aware to keep track of possible security-updates for these packages on your own from now on! (Upgrading to the possibly new packages then should be easily possible by running "dpkg -i ..." again. Make sure to have backups of your configuration before as well as to set the packages to HOLD afterwards again.)

Next, teach your LDAP server the possibilities of Samba by adding the Samba schema file and restrict the access to the users' Samba passwords using the Access Control Lists:

```
Run:
  cp /usr/share/doc/samba-doc/examples/examples/LDAP/samba.schema.gz /etc/ldap/schema/
  cd /etc/ldap/schema/
  gunzip samba.schema.gz
  chown slapd.slapd samba.schema
  chmod 440 samba.schema

Add to /etc/ldap/slapd.conf:
  include /etc/ldap/schema/samba.schema

Change the already exiting password ACL rule in /etc/ldap/slapd.conf:
  # access to dn=".*,dc=subnet,dc=at" attribute=userPassword
  access to dn=".*,dc=subnet,dc=at" attribute=userPassword,lmPassword,ntPassword
    [...]

Restart OpenLDAP:
  /etc/init.d/slapd restart
```

Now that you have an "LDAP capable" Samba installed and now that your OpenLDAP knows about the new attributes, of course the Samba server needs to be able to query the LDAP server for users. I tried to think of a setup which would not use "cn=manager,..." for this, but I just couldn't figure one out. The main reasons are that Samba (in some way) must be able to change the passwords (e.g. using `"smbpasswd"`) and above all edit or even add LDAP entries (e.g. machine accounts for new SMB-clients joining the domain).

Of course, Samba has to know about this user and it's password. This leads us to the next (and one of the last) steps to do: configure `smb.conf` (and thus Samba and its tools) correctly to use the LDAP server properly. Here are the additions to be added to the `[global]` section of your Samba PDC:

```
/etc/samba/smb.conf, add to [global]:

  # Without SSL:
  ldap admin dn = cn=manager,dc=subnet,dc=at
  ldap server = ldap.subnet.at
  ldap suffix = ou=People,dc=subnet,dc=at

  # Plus these options for SSL support:
  #ldap port = 636
  #ldap ssl = on

Restart Samba:
  /etc/init.d/samba restart
```

Here is a sample smb.conf file.

BTW: As Implementing a Samba LDAP Primary Domain Controller Setup on Mandrake 9.x states, we probably don’t need (and thus don’t want) the overhead of encryption on the very same system where both OpenLDAP and Samba are running on. (If the two services run on different machines, you can add the two options "ldap port" and "ldap ssl" accordingly.

Last but not least, we have to tell Samba the corresponding password:

```
  # read -s -p "Enter LDAP Root DN Password: " LDAP_BINDPW
  # smbpasswd -w $LDAP_BINDPW
    --> output:
      Setting stored password for "cn=manager,dc=subnet,dc=at" in secrets.tdb
```

This way, you set an environment variable to the corresponding password and use it in the next command to tell Samba about it. (That’s a very neat trick I saw at B. Milne’s "Implementing Disconnected Authentication and PDC/BDC Relationships Using Samba and OpenLDAP" to keep the password from your shell’s history file and similar. It will not show up anywhere :).)
`"smbpasswd -w $LDAP_BINDPW"` is the actual command of interest. Basically, it stores a password hash for the user of smb.conf’s option `"ldap admin dn"` in the file `/var/lib/samba/secrets.tdb`.

Samba: Test your Setup

Before messing around too much with your (already existing?) LDAP users, you might want to test if the new setup works properly.
If you don’t want to do this, just go on and proceed with the next section Samba: Add (Windows) Users.

As Samba needs a "Linux user" for/below every "Samba user", we simply add one to the local flat files (`passwd|shadow|group`). Not saving it in LDAP at this time yet helps keeping things seperated, simple and "a way which is already known".
Nevertheless, the corresponding Samba user to be added will - of course - be stored in the LDAP database, ’cause that’s what we want actually.

```
Add the "Linux user" (to local flat files):
  # adduser --no-create-home maxsmb
      [e.g. password: 12345]

Add the "Samba user" (to LDAP):
  # smbpasswd -a maxsmb
    --> output:
      New SMB password: abcde
      Retype new SMB password: abcde
      LDAP search "(&(uid=maxsmb)(objectclass=sambaAccount))" returned 0 entries.
      Added user maxsmb.

Check yourself:
  # getent passwd | grep maxsmb
  # ldapsearch -b "ou=People,dc=subnet,dc=at" -LLL -D "cn=manager,dc=subnet,dc=at" -W -x "(uid=maxsmb)"
```

One can now access the Samba server using this user, change its password using `"smbpasswd $user"`, etc. - business as usual. The "only" difference is that now *the LDAP server* is used to hold the information which is usually stored in `smbpasswd`:

```
client$ smbclient -L //ldap
Password:
Anonymous login successful
Domain=[SUBLDAP] OS=[Unix] Server=[Samba 2.2.3a-12 for Debian]
      Sharename      Type      Comment
      ---------      ----      -------
      tmp            Disk
      maxsmb         Disk
      IPC$           IPC       IPC Service (yellow server (Samba 2.2.3a-12, LDAP-Test))
      ADMIN$         Disk      IPC Service (yellow server (Samba 2.2.3a-12, LDAP-Test))

client$ smbclient //ldap/tmp
added interface ip=193.170.141.119 bcast=193.170.141.127 nmask=255.255.255.128
Password:
Anonymous login successful
Domain=[SUBLDAP] OS=[Unix] Server=[Samba 2.2.3a-12 for Debian]
smb: \>

client$ smbclient //ldap/maxsmb
added interface ip=193.170.141.119 bcast=193.170.141.127 nmask=255.255.255.128
Password:
Anonymous login successful
Domain=[SUBLDAP] OS=[Unix] Server=[Samba 2.2.3a-12 for Debian]
tree connect failed: NT_STATUS_WRONG_PASSWORD

client$ smbclient //ldap/maxsmb -U maxsmb
added interface ip=193.170.141.119 bcast=193.170.141.127 nmask=255.255.255.128
Password: abcde
Domain=[SUBLDAP] OS=[Unix] Server=[Samba 2.2.3a-12 for Debian]
smb: \>
```

Now that everything is proven to work properly, you can extend your LDAP users to become Samba capable.
But before doing that, don’t forget to remove this section’s test user "maxsmb":

```
# ldapdelete -D "cn=manager,dc=subnet,dc=at" -W -x "uid=maxsmb,ou=People,dc=subnet,dc=at"
# deluser --remove-home maxsmb
```

Samba: Add (Windows) Users

In order for Samba to allow access to shares for certain users (and don’t allow for others), it needs to know these users. If you just have a few of them, there’s nothing easier than to simply add them:

```
# smbpasswd -a $user
```

This command seems to perform an LDAP search for an already existing entry matching `"(&(uid=maxsmb)(objectclass=sambaAccount))"`.
If none is found, it either creates the entire user (as is the case in the example in section Samba: Test your Setup), or it uses an already existing normal Unix-user and adds the corresponding Samba attributes.
If it finds an entry matching its query (which means the user already exists as a Samba-user), it simply updates the password-hashes (no matter whether it’s invoked as `"smbpasswd $user"` or `"smbpasswd -a $user"`).

If you have *lots of users* already existing in the LDAP tree (e.g. due to migrating them as described above), or if you have lots of "Windows-users" to add, you'll need a script to do the work:

```
# Warning: This should work, but I didn't try it in large scale:

ldapsearch -b "ou=People,dc=subnet,dc=at" -LLL -D "cn=manager,dc=subnet,dc=at" \
  -W -x '(&(objectClass=posixAccount)(!(objectClass=sambaAccount)))' | grep "uid: " \
  | awk '{print $2}' > linux-and-not-samba-users.txt
for user in `cat linux-and-not-samba-users.txt`; do echo $user `makepasswd` \
  >> users-with-samba-passwords.txt; done
sed s/^/"smbpasswd -a "/ users-with-samba-passwords.txt > make-them-samba-users.sh

chmod 600 users-with-samba-passwords.txt
chmod 700 make-them-samba-users.sh
./make-them-samba-users.sh
```

This takes all Linux-users which are not Samba-users already, makes them Samba-users and assigns them a random password (creating the random passwords using `makepasswd` might take a while!). You can easily tell your users about their passwords as they are stored in users-with-samba-passwords.txt.
Mind that the filter is included in `'...'` and not `"..."`. (Somehow the BASH messes something up when using double-quotes, even when escaping & and `!` using `\`.)

Samba: Migration Summary

- If you need to migrate lot's of users from a Windows-PDC (such as Windows NT), you might check out smbldap-tools.
  Scott Phelps migrated a Windows-PDC to a Samba-PDC without the clients noticing a change: He used pwdump2 to keep the user passwords and rpcclient and smbpasswd to have the Samba-PDC use the old Windows-PDC's SID for the domain. (*ToDo:* Insert link to the Howto once available.)
  Also, Samba 3.0 will allow to migrate a complete NT-securitydatabase to a Samba-PDC by calling `"net rpc vampire"`.
- If you need to migrate lot's of users from an already existing Samba-PDC with the users being stored in a flat smbpasswd file, check out /usr/share/doc/samba-doc/examples/...
- If you need to create an LDAP user database with lot's of users which are to become Samba-users, create the posixAccount-users (e.g. by migrating them as described above) and afterwards run the commands mentioned in section Samba: Add (Windows) Users to "smbpasswd -a" all posixAccounts and hereby make them sambaAccounts too. (This seems to me to be easier than to create an .ldif-file holding posixAccounts *and* sambaAccounts in the first place and add this .ldif-file to the LDAP server. "smbpasswd" does all the work for us... perfect!)

Samba: Join Windows-Workstations to our Samba-PDC Domain

03-08-13:
In contrast to the initial release of this howto, I meanwhile figured out how to have a machine account automatically be added to the LDAP-tree.

Both possible ways, adding the account manually or have Samba add it automatically (if it doesn't exist already), use this script I wrote: create-machine-account.sh.
ToDo: Some parts of the script should be rewritten to clear things up and make the script simpler (e.g. by

using functions to print the status messages). It’s a little mess currently, but it works.

1. To add the machine manually, run `"./create-machine-account.sh NewMachineName$ I"`. Option "I" activates the interactive mode (printing status messages to stdout and possibly querying you for the rootbinddn-password).
2. To have Samba add the account automatically while the machine joins the domain, add the following option to your `smb.conf`’s `[global]` section:

   ```
   add user script = /usr/local/sbin/create-machine-account.sh %u
   ```

What `create-machine-account.sh` does basically boils down to:

- Get the necessary data to be able to connect to the LDAP-server. (The script uses settings in `pam_ldap.conf` and `ldap.secret` for this. This indicates it’s best run on the LDAP-server/PDC itself.)
- Next, it finds the highest uidNumber of any already existing machine-account. (In this setup, machine-accounts can be distinguished from others as they are posixAccounts with "gecos=MachineAccount".)
  In this setup, machines use a uidNumber-range which is seperated from the normal Linux users. Seperation is done by simply using uidNumbers above $DEFAULTUID, defaults to 20000.
- If necessary, it creates the group "machines" (gidNumber $GIDNUMBER, default 20000). This group will be the group of all machines.
- It checks if the machine-account already exists. If so, it exits.
- If everything went fine until here, it creates the Linux-account.
- Afterwards, it makes this new entry a full Samba-Machine-Account using `smbpasswd -a -m`.

Usage: `# ./create-machine-account.sh NewMachineName$ <I>` (*WITH* the machine account’s trailing "$"). In "non-interactive" mode (i.e. without option "I"), all the script’s status messages are logged using /usr/bin/logger.
Mind: At the beginning of the script are three options which can be changed, but are consistent with our setup here by default.

Joining the domain on the Windows machine works as usual. Just perform steps 1 and 3 of this mini-Howto describing exactly that: Howto join a Windows client to a domain. (*But be careful:* You’ll need the user "root" as mentioned in step 2. If it doesn’t already exist, adding at least the "objectClass: sambaAccount" part to LDAP "is left to the reader as an exercise". I now, you probably hate this phrase as much as I do... sorry!)
You can now log on to this machine using all valid LDAP sambaAccount entries.
According to B. Milne’s document it should be possible to have all "Domain Administrators" join a machine to the domain. I didn’t try this yet.

BTW: Using `"ou=People"` is different from the smbldap-tools (smbldap-tools work-log) which use `"ou=Computers,dc=,dc="` for machine accounts.
Nevertheless, searching for MachineAccounts is easy here too: just "ldapsearch" for "(gecos=MachineAccount)".

Add-on: Joining a Windows machine to the domain if the machine-account already exists works just fine too. (Thus it’s possible to re-use machine-accounts in case a machine left the domain without having to delete the machine-account first.)

Samba: Join Samba-Workstations to our Samba-PDC Domain

Joining a Samba-Workstation to our Samba-PDC controlled domain involves the same steps on the server-side as does joining a Windows-Workstations. That makes sense, of course, since both systems behave the same and talk the same protocol.
This results in in the same steps as described above in section Samba: Join Windows-Workstations to our Samba-PDC Domain:

1. On the Server: Create the Linux-User account "MachineName$"
2. On the Server: Make it a Samba-machine-account
3. On the Client: Join the domain

So, after creating the machine account (for both Linux and Samba) using our script `"create-machine-account.sh"`, join the client to the domain by running the following commands on the new workstation:

```
client:~# /etc/init.d/samba stop

client:~# smbpasswd -j SUBLDAP -r YELLOW
2003/08/08 20:24:31 : change_trust_account_password: Changed password for domain SUBLDAP.
Joined domain SUBLDAP.

client:~# /etc/init.d/samba start
```

Here is the according sample smb.conf of a joining workstation.
The most important options are `"workgroup = SUBLDAP"`, `"security = DOMAIN"`, `"password server = YELLOW"` and `"local master = No"`.
Oh, and just to make sure: I performed this stuff with `/etc/samba/` containing *only* `smb.conf`, and no other files such as `MACHINE.SID` or similar stuff.

Just for your information: Joining the domain creates the file `/etc/samba/MACHINE.SID` on the joining Samba machine.

Samba: Miscellaneous

- Here’s a document describing my experiences with the smbldap-tools: my smbldap-tools work-log.
- Mind the very useful `smb.conf`-option `"domain admin group = @staff @winadmins"`.

So Far So Good, Part 2

After successfully setting up the server and the (Linux-) clients (see So Far So Good, Part 1), we migrated the current Linux-users of the network (which a NIS-server might have provided to the clients) using the official MigrationTools package from padl.com: We created ldif files for the base structure, the groups and the users (including shadow-passwords) and edited them (e.g. to exclude system users) prior to adding this

information to the LDAP-Server.
Next, we set up LDAP-capable Samba-packages on the server only. In our network, this Samba-server will become the domain’s PDC with all Windows- and Samba-clients querying this PDC. We taught the OpenLDAP-server to use the new Samba-attributes. Using some commands, we added sambaAccount’s to our posixAccount’s users, assigning random passwords to the users. Furthermore, we discussed joining SMB-clients (both Windows and Samba) to the domain using a custom script to set up the corresponding machine accounts.

ToDo: LDAP-Client-Interfaces

*As the header indicates, this section is currently still marked "**ToDo**".*
As basically goes for the total HOWTO: I’ll add things as soon as they are ready, corrections as soon as bugs are found.

Besides the Gnome- and web-based tools mentioned here, there are also KDE tools, of course. Check out this list of Graphical LDAP tools over at the LDAP Linux HOWTO.

Directory Administrator

Directory Administrator is a GTK-based LDAP client.
Directory Administrator makes adding/editing/deleting users/groups *really* easy! If it wasn’t for all our wish to understand what’s *behind* all those GUI tools, one would probably use this one from the beginning already.

Debian packages for a current version backported to Woody can be found in Christof Meerwald’s Debian section.

GQ

GQ is a GTK-based LDAP client.
GQ is great to browse your overall LDAP tree and get a good feeling of what’s where. Easy access to all available attributes etc.

Debian packages for a current version backported to Woody can be found in Christof Meerwald’s Debian section.

ToDo: phpLDAPadmin

Citing the homepage:
phpLDAPadmin is a web-based LDAP application for managing all aspects of your LDAP server."

(phpLDAPadmin was formerly known as "DaveDAP". Citing from its description: *"DaveDAP is a web-based LDAP admin tool written in PHP. You can browse your LDAP tree, create, delete, edit, and copy objects, perform searches, and view your server’s schema. You can even copy objects between two LDAP servers and recursively delete or copy entire trees."*)

ToDo: Miscellaneous

- http://www.gonicus.de/
  (Found on http://www.pro-linux.de/news/2002/4491.html)
- http://ldapweb.sourceforge.net/
- http://www.iit.edu/~gawojar/ldap/
- cpu -- Console based LDAP user management tool.
- LDAP Explorer Tool.

Miscellaneous

- As B. Milne points out in a mail on the Samba mailing-list, it might be good to use Samba 2.2.8a (or later) if you want to get a Samba-BDC working fine.
- HOWTO: LDAP Programming in Python
- Only book Understanding LDAP featured by IBM.
- libapache-mod-ldap - Apache authentication via LDAP directory.
- Carla Schroder’s Building an LDAP Server on Linux, Part 1, Part 2, and Part 3.
- Some links about LDAP I found in a posting by Georg Wallner on the LUG-TS-Mailing-List:
  - Lots of links to and info ’bout LDAP.
  - Kerberos and LDAP slides.
  - Deutsches LDAP-HOWTO zu Debian Potato.
  - Verschiedenes zu LDAP - ebenfalls deutschsprachig.

Part II: Using OpenLDAP on Debian Sarge to serve Linux Users

Table of Contents (Part II: Sarge)

Part II - Sarge: Introduction

Part II - Sarge: On the Versions ...

As much as I would have liked to re-work this document for Debian 3.1 ("Debian Sarge"), I unfortunately neither have the time nor the resources to do so. (I moved to the UK, and I do not have access to e.g. the necessary hardware resources here anymore.)

Still, I would like to share my experiences with a *preliminary version* of Debian Sarge, dated back about half a year ago to January 2005.
Note though that my notes below are based on LDAP packages of versions 2.1.xx, while the final Debian Sarge ships LDAP packages of versions 2.2.xx.
Since, thus, the following work-log is currently *not* based on the final version of Sarge, please treat it as such: A work-log.
(This state of information is also reflected in the state of structuring: It's more a "bunch of" than a "list of". Sorry 'bout that ... should I get the time and resources again to work on this any further, I'll gladly do so.)

Part II - Sarge: The Differences to the Woody Documentation Above

With the Sarge version, I tried to stick more to the Debian default packages as they are. The default configuration is not removed anymore, for example. Recompilation of the packages is also not necessary ...

If you're running Debian Sarge, give the descriptions below a chance. For more information on some issues, please check the Woody documentation above. (The general LDAP principles are still the same, of course!)

Part II - Sarge: Feedback!?

Comments and corrections are welcome indeed, as always!

Part II - Sarge: The Work-Log

Part II - Sarge: Basic Installation and Configuration

Install the packages `"slapd"` (2.1.30-3) and `"ldap-utils"` (2.1.30-3) as well as the dependencies and recommendations:

```
# dselect
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  db4.2-util ldap-utils libiodbc2 libltdl3 libsasl2-modules slapd
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1559kB of archives.
After unpacking 4026kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Concerning the configuration of these packages:

- enter your DNS domain: `example.com`
- Name or your organization: `example.com`
- Admin password: `adminpassword`
- Allow (old) LDAPv2? `No.`
  ("allow bind_v2" would have been added to slapd.conf in case we allowed v2.)

The only thing to change in the `/etc/ldap/slapd.conf` file is to add (near the beginning of the file) the "misc.schema" to be used:

```
######################### /etc/ldap/slapd.conf #########################
# Note: We need this to be able to have entries with attributes such as
#       "mailRoutingAddress" (which we need as we’ll use the LDAP-server
#       to host the mail-users and -domains for the Postfix SMTP server).
#
# [...]
include /etc/ldap/schema/misc.schema
# [...]
```

Don’t forget to restart the LDAP server afterwards:

```
# /etc/init.d/slapd restart
```

Part II - Sarge: LDAP Clients

In order to connect to the server from an Ubuntu Linux client (which is what I use on my client machine), simply install the `"ldap-utils"` package and run something like:

```
$ ldapserach -x -b "dc=example,dc=com"
```

If you want to connect as the LDAP-admin from anywhere on the network, use something like this

```
$ ldapsearch -b "dc=example,dc=com" -LLL -D "cn=admin,dc=example,dc=com" \
  -H "ldap://yourserver.example.com" -W -x
```

If you want a graphical client, install e.g. the package `"gq"`. Ubuntu 4.10 (aka "Warty") comes with version 1.0beta1-1. For some reason, I could not add a new server to its config using the graphical menu. The solution is to fire up your editor and add the following section to your (already existing as you should start once and stop GQ first) `~/.gq` file:

```
  <ldapserver>
      <name>peach</name>
      <ldaphost>yourserver.example.com</ldaphost>
      <ldapport>389</ldapport>
      <basedn>dc=example,dc=com</basedn>
      <binddn>cn=admin,dc=example,dc=com</binddn>
      <pw-encoding>Base64</pw-encoding>
      <search-attribute>cn</search-attribute>
  </ldapserver>
```

You can now modify the server settings, but with the above config, you should already be able to connect to the server as "admin". Note though that all your data (including the password) is transferred in cleartext!

Part II - Sarge: Database Population

Next, add some test-data to your LDAP database, using the following file "`basic-test-user.ldif`":

```
dn: ou=People,dc=example,dc=com
objectClass: organizationalUnit
ou: People

dn: ou=Group,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Group

dn: uid=maxldap,ou=People,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
objectClass: organizationalPerson
objectClass: inetLocalMailRecipient
uid: maxldap
cn: Markus Peach LDAP User Amersdorfer
sn: Amersdorfer
givenname: Markus Peach LDAP User
title: Dipl.-Ing.
departmentNumber: IT
mobile: 012-345-6789
postalAddress: AddressLine1$AddressLine2$AddressLine3
telephoneNumber: 1234-567890
facsimileTelephoneNumber: 012-345-6789
userpassword: {CRYPT}SOME-CHARACTERS-OF-YOUR-PASSWORD-HERE
labeleduri: http://homex.subnet.at/~max/
mail: my.email.address@example.com
mail: my.alternate.email.address@example.com
mailRoutingAddress: my.email.account@mail.server.example.com
loginShell: /bin/bash
uidNumber: 12345
gidNumber: 12345
homeDirectory: /home_local/maxldap/
gecos: maxldap_gecos-field
description: Not Available
localityName: I dont know

dn: cn=maxldap,ou=Group,dc=example,dc=com
objectClass: posixGroup
objectClass: top
cn: maxldap
gidNumber: 12345
```

To add this to the running slapd-LDAP-server’s database, run the following:

```
  $ ldapadd -f basic-test-user.ldif -D "cn=admin,dc=example,dc=com" -W -x
  Enter LDAP Password:
  adding new entry "ou=People,dc=example,dc=at"

  adding new entry "ou=Group,dc=example,dc=com"

  adding new entry "uid=maxldap,ou=People,dc=example,dc=com"

  adding new entry "cn=maxldap,ou=Group,dc=example,dc=com"
```

Running `"slapcat"` as root on your LDAP-server, you’ll notice that some additional attributes such as "creatorsName" and "modifyTimestamp" were added automatically.
Note: AFAIK, you should use the `"slapcat"` and `"slapadd"` commands only when the slapd-process is stopped!

If you wanted to remove these entries again, use the following file `"basic-remove.ldif"`:

```
dn: uid=maxldap,ou=People,dc=example,dc=com
changetype: delete

dn: cn=maxldap,ou=Group,dc=example,dc=com
changetype: delete

dn: ou=People,dc=example,dc=com
changetype: delete

dn: ou=Group,dc=example,dc=com
changetype: delete
```

and run the following command:

```
  $ ldapmodify -f basic-remove.ldif -D "cn=admin,dc=subnet,dc=at" -x -W
  Enter LDAP Password:
  deleting entry "uid=maxldap,ou=People,dc=subnet,dc=at"

  deleting entry "cn=maxldap,ou=Group,dc=subnet,dc=at"

  deleting entry "ou=People,dc=subnet,dc=at"

  deleting entry "ou=Group,dc=subnet,dc=at"
```

Part II - Sarge: Configuring NSS

```
  # apt-get install libnss-ldap
```

You are asked questions such as "is a login needed to retrieve data from the LDAP db?" and "should the libnss-ldap configuration file be readable and writable only by the file owner?", which is all more than interesting and could be necessary if you use a non-Debian-default setup. (This might be used to harden the installation later, but at the moment, we use the defaults...)
(From one of those Debconf-dialogs: "Note: As a sanity check, libnss-ldap will check if you have nscd installed and will only set the mode to 0600 if nscd is present." Thus, it might really be a good idea to use

"nscd"! See http://homex.subnet.at/~max/ldap/index.php#nss-install, section "NSCD and /etc/libnss-ldap.conf" for more details!)
The example-file `/usr/share/doc/libnss-ldap/examples/nsswitch.ldap` holds really good information for decent configuration of our `/etc/nsswitch.conf`.

Nevertheless, we'll stick with the basics at the moment and just change the existing `/etc/nsswitch.conf` a little bit - after making a backup of it:

```
  # cp /etc/nsswitch.conf /etc/nsswitch.conf_05-01-05
  # $EDIT /etc/nsswitch.conf
    [...]
    passwd:  ldap compat
    group:   ldap compat
    shadow:  ldap compat
    [...]
```

You should now be able to see the user via the NSS library calls:

```
  # finger maxldap
  Login: maxldap                          Name: maxldap_gecos-field
  Directory: /home_local/maxldap/         Shell: /bin/bash
  Never logged in.
  No mail.
  No Plan.

  # getent passwd|grep maxldap
  maxldap:x:12345:12345:maxldap_gecos-field:/home_local/maxldap/:/bin/bash
```

Great.

Part II - Sarge: Configuring PAM

In order to be able to *authenticate* against the LDAP server using the user's password, we need to adapt the PAM service:

```
  # apt-get install libpam-ldap
```

I basically used the same choices are as mentioned in http://people.debian.org/~torsten/ldapnss.html, as described as follows.
(Note: You could choose to use "MD5" password hashes here!? Again, this sounds like it would be a good idea, but we'll stick with "crypt" for the moment though...)

I adapted `common-account`, `common-auth` and `common-password` -- slightly different changes each:

```
# /etc/pam.d/common-account - authorization settings common to all services
# markus -- 05-01-05:
# To activate LDAP support, comment the default and add the LDAP config
#account        required        pam_unix.so
account         sufficient      pam_ldap.so
account         required        pam_unix.so try_first_pass
```

```
# /etc/pam.d/common-auth - authentication settings common to all services
# markus -- 05-01-05
# To activate LDAP support, comment the default and add the LDAP config
#auth   required        pam_unix.so nullok_secure
auth    sufficient      pam_ldap.so
auth    required        pam_unix.so nullok_secure use_first_pass

# /etc/pam.d/common-password - password-related modules common to all services
# markus -- 05-01-05
# To activate LDAP support, comment the default and add the LDAP config
#password   required   pam_unix.so nullok obscure min=4 max=8 md5
password   sufficient pam_ldap.so
password   required   pam_unix.so nullok obscure min=4 max=8 md5 use_first_pass
```

(Note: As mentioned in Torsten’s Howto, `common-session` goes unchanged... At the moment I do not know why? Shouldn’t we make the ldap-changes there too?)

It is now possible to log in on the command-line using NIS-users (correct password: accept; wrong password; reject) as well as the maxldap-LDAP-user!! :)
`"su"` and `"ssh"` work too!
(Note: For "ssh" to work properly, you have to restart the ssh-service first!)

Part II - Sarge: Have the Server Processes Run as Non-root

In order to get the "slapd" process to run as a different user and group than root, check out `"/etc/default/slapd"` first of all.

According to `/usr/share/doc/debian-policy/policy.txt.gz` (package "debian-policy"), section "9.2.2. UID and GID classes", the system-UIDs 100-999 can be assigned for system-purposes on a dynamical basis. Only *0-99 must not be used on a per-machine-basis*!

Thus, create the group and user as follows:

```
  # addgroup --system ldap
  # adduser --system --no-create-home --group ldap
```

Check `/etc/passwd` and `/etc/group`. There should be a group called "ldap", as well as a "ldap"-user with the GID of the just mentioned group. Both IDs should be between 100-999.

After adding "ldap" to SLAPD_USER and SLAPD_GROUP in `/etc/default/slapd`, trying to restart the slapd-process will result in error. (Check `/var/log/syslog` for more information.)
You need to enable "ldap" to read the config-file:

```
  # chown ldap /etc/ldap/slapd.conf
```

Next, change the ownership of the files under /var/:

```
  [<0> root@peach ldap]# ll /var/lib/ldap/ -d
  drwxr-xr-x  2 root root 4096 Jan  3 15:04 /var/lib/ldap/
  [<0> root@peach ldap]# ll /var/lib/ldap/ -a
  total 540
  drwxr-xr-x   2 root root   4096 Jan  3 15:04 .
```

```
drwxr-xr-x  16 root root   4096 Jan  3 15:04 ..
-rw-------   1 root root   8192 Jan  3 15:04 __db.001
-rw-------   1 root root 270336 Jan  3 15:04 __db.002
-rw-------   1 root root  98304 Jan  3 15:04 __db.003
-rw-------   1 root root 368640 Jan  3 15:04 __db.004
-rw-------   1 root root  16384 Jan  3 15:04 __db.005
-rw-------   1 root root   8192 Jan  7 16:31 dn2id.bdb
-rw-------   1 root root  32768 Jan  7 16:31 id2entry.bdb
-rw-------   1 root root  97140 Jan  7 16:31 log.0000000001
-rw-------   1 root root  12288 Jan  7 16:31 objectClass.bdb
[<0> root@peach ldap]# ll /var/lib/slapd/ -d
drwxr-xr-x  2 root root 4096 Jan  3 15:04 /var/lib/slapd/
[<0> root@peach ldap]# ll /var/lib/slapd/ -a
total 8
drwxr-xr-x   2 root root 4096 Jan  3 15:04 .
drwxr-xr-x  16 root root 4096 Jan  3 15:04 ..
-rw-r--r--   1 root root    0 Jan  3 15:04 suffix_change
[<0> root@peach ldap]# ll /var/run/slapd/ -d
drwxr-xr-x  2 root root 4096 Jan  7 16:31 /var/run/slapd/
[<0> root@peach ldap]# ll /var/run/slapd/ -a
total 8
drwxr-xr-x  2 root root 4096 Jan  7 16:31 .
drwxr-xr-x  6 root root 4096 Jan  7 16:31 ..

# chown ldap /var/run/ldap/ -R
# chown ldap /var/lib/slapd/ -R
# chown ldap /var/run/slapd/ -R
```

Starting slapd again should work now (`# /etc/init.d/slapd start`), and it should run as "ldap"-user now instead of "root":

```
# ps aux|grep slapd
ldap  2039  0.0  0.5 32916 4480 ?  Ss  16:39  0:00 /usr/sbin/slapd -g ldap -u ldap
[...]
```

Part II - Sarge: Miscellaneous ToDos include ...

Apart from updating these notes to the actual Debian 3.1 release versions ... the following steps are some of those that needed to be performed next:

- Get the "hosts"-attribute and the according filtering to work.
- SSL: There was a SSL-Bugs in slapd, which was fixed with 2.2.23-1, with 2.2.23-8 being in the final Sarge release (see bug 205452).
- SSL - question: Do libnss-ldap and libpam-ldap support SSL properly?
- SSL: `/etc/default/slapd`: Activate "SSL"-startup-option there (and NOT in `/etc/init.d/slapd` itself)!
- As a larger part of the project, have Postfix use the LDAP server for its SMTP services.
- Include further Mail-services to use the LDAP users, if necessary.
- ...

User Comments

I would be glad to hear about your opinion, any corrections or additions. (The form is at the end of this file.)

- Tue, 12 Aug 2003 09:20:25 +0200
  Great document! Thank you very much for taking the time to write it. I will spread the word.

- Tue, 12 Aug 2003 12:17:23 +0200
  Great document! Thank you very much for taking the time to write it. I will spread the word.

- Thu, 14 Aug 2003 22:57:35 +0200
  Section: Samba 2.2.x and LDAP
  SubSection: Samba: Introduction
  Paragraph: 1
  There is a replacement GINA for Windows 2000/XP that allows several other methods of authentication.
  Project homepage:
  http://pgina.sourceforge.net/ (old)
  http://pgina.xpasystems.com/ (new)
  pGina LDAPAuth:
  http://pgina.xpasystems.com/plugins/ldapauth.html

- Tue, 26 Aug 2003 16:00:55 +0200
  As I mailed you before - great doc! I put my own notes (based on this to a large extent) into doc format here: http://mawi.org/sambaldap/Samba_and_LDAP_on_Debian.html
  Soon done with the main thing - a \"quick cheat sheet\". Check it out!
  //mawi

- Tue, 16 Sep 2003 12:00:09 +0200
  hi, Amersdorfer, your howto is very good, thank you very much, it will save me much time.

- Fri, 7 Nov 2003 17:29:03 +0100
  just got into ldap, thanks to your howto, tough i let out most of the encryption stuff as this is just a private plaything. I was more than happy to find such a practical howto on this topic
  thx really :)

- Mon, 24 Nov 2003 14:58:00 +0100
  A great document. But the background color and the text color make it a bit difficult to read.
  Congratulations! Kablan BOGNINI, bognini@hotmail.com

- Fri, 12 Dec 2003 06:59:59 +0100
  I would implement it if you could get md5 to work.
  admin at cs . montana . edu

- Mon, 15 Dec 2003 02:57:09 +0100
  Excellent article.

- Mon, 29 Dec 2003 15:34:46 +0100
  A very helpfull page,
  Thanks,
  Robert

- Fri, 9 Jan 2004 21:49:29 +0100
  Good document, thanks!
  - with slapadd -v -l populate.ldif, I received the message \’attribute \’gidNumber\’ cannot have multiple values. The file
  \’http://homex.subnet.at/~max/ldap/basics_subnet.ldif\’ containts lot\’s of spaces here... Removing the emtry \’spaced\’ line by a linefeed solved that..
  [Max, 04-01-16: This was due to a copy’n’paste-error. Fixed, thanks!]
  - After the \’Start openLDAP\’ you can add that you have to be root again (using exit), I tried to start is as my slapd user but that didn\’t work out.
  [Max, 04-01-16: Added a few words to make it clearer.]
  - I believe but not sure that the installing directions in the \’Install OpenLDAP\’ section is wrong.. I found out that my system was not compiled with
  SSL.. Possible the \’Activate SSL:\’ step must be done right over the \’apt-get source slapd\’ step.
  The file \’http://homex.subnet.at/~max/ldap/basics_subnet.ldif\’ containts lot\’s of spaces here... this gave me problems using
  - I believe that the slapd package should be set on hold to.
  But I got it working now!!! yeaaaaaa

- Mon, 9 Feb 2004 12:31:18 +0100
  Nice document. Added a link from my small tutorial on getting it running
  without all the bells and whistles you added. As a Debian developer I wanted
  to show what you can do without recompiling and all that of course. I\’ll try
  to improve the relevant packages so that all the rebuilding steps can be left
  out...
  Torsten Landschoff <torsten@debian.org>

- Tue, 10 Feb 2004 20:59:49 +0100
  This is the good stuff, I wish I found this page a year a go...

- Wed, 11 Feb 2004 19:10:32 +0100
  Thank you for your felpfool Page.
  It\’s just I\’m looking for :)
  Serz

- Tue, 17 Feb 2004 23:08:00 +0100
  Interesting. I\’ve been running LDAP for a while now, and I learned a couple things - nice work. :)

- Wed, 18 Feb 2004 14:50:26 +0100
  Thanks, I was looking for something like this. I\’ve recently come to the conclusion that I\’m looking after too many machines, spread out over the internet to not use LDAP. One question though, about the database poopulation: anyone have experience with populating the database with users/groups/aliases/... for multiple domains (sometimes distinct organisations)?

- Wed, 18 Feb 2004 23:47:15 +0100
  Please note that the GPL is not the \"GNU Public License\"; there\’s no such thing. The GPL is the \"General Public License\".
  [Max, 04-02-19: Ooops, noted and changed accordingly. Thanks for the hint :).]

- Fri, 20 Feb 2004 01:19:17 +0100
  Great tutorial but just a few hangups on the way.
  debhelper package is required in order to build libnss-ldap and libpam-ldap.
  Also for libpam-ldap libpam-dev is required.
  [Max, 04-03-25: "apt-get build-dep" is very helpful here, see its man-page. (I’ll have to integrate information about it in the HOWTO eventually.)]

- Fri, 20 Feb 2004 15:51:58 +0100
  \"Unfortunately, either Debian doesn\’t support it or I simply couldn\’t get this (MD5 crypt) thing to work.\"
  The problem is that you chose to rebuid slapd --with-tls. Without the use of ssl you can use md5 without problems.
  The problem seems to be in the linking of openssl (woody version) function crypt() before the usual libc crypt().

- Mon, 23 Feb 2004 17:05:54 +0100
  http://samba.idealx.org/index.en.html

- Sat, 20 Mar 2004 18:57:39 +0100
  I got md5 working as follows (on sarge, not tested on woody yet):
  one: tell the server to use md5 (add password-hash {MD5} to the configuration)
  two: tell pam to crypt locally: (pam_password md5 and pam_md5 local)
  You might want to check this though, lord knows whatever else I changed while messing around :-)
  [Max, 04-03-23: Removed multiple postings.]

- Wed, 24 Mar 2004 02:03:41 +0100
  Slapd seems not to drop root when started from /etc/init.d/slapd with -u slapd, even htough it does if run \"by hand\"... (Debian woody stable)
  [Max, 04-03-24: I checked again, it runs as "slapd" here. Perhaps your script misses the " -- " before "-u slapd" to identify the rest of the line as command-line arguments to "slapd" (instead of "start-stop-daemon")?]

- Wed, 28 Apr 2004 01:33:58 +0200
  Please leave a comment (but mind the netiquette).Das beste, das ich bisher über die Odyssee, die einem beim Umstellen auf LDAP blüht, lesen und auch verwenden hab können.
  Nicht auszudenken, wie sichs gelesen hätte, wenns in deiner Vatersprache zu lesen gewesen wäre.
  Merci bien et, s\’ il te plait , encore cent fois

[Max, 04-04-29: Removed multiple postings.]

- Sat, 19 Jun 2004 22:53:47 +0200
  Schönes HowTo, schade das nicht einmal auf Kerberos hingewiesen wird, man stolpert zwangsweise darüber wenn man es ldap länger einsetzt und es wäre schöner gewesen alles auf einmal zu migrieren

- Wed, 23 Jun 2004 11:38:12 +0200
  Your link to LDAP Schema Viewer is broken: the target page does not exist.
  May I suggest phpldapadmin instead? It is in debian unstable: apt-get install phpldapadmin
  [Max, 04-07-15: Added a working link to this resource, thanks for the hint.]

- Thu, 24 Jun 2004 17:52:58 +0200
  realy great info, thank you !

- Fri, 25 Jun 2004 05:21:47 +0200
  for ldap clients, how about using LDAP Explorer Tool? It\'s a GPL win32/linux ported client to work with ldap. works great.
  http://ldaptool.sourceforge.net/
  [Max, 04-07-15: Added this link, thanks for the hint.]

- Thu, 22 Jul 2004 11:20:15 +0200
  Thanks Max for your nice doc, do you have an experiance with samba 3 + ldap + squid + qmail .. single sing-on system

- Tue, 31 Aug 2004 03:54:06 +0200
  Great document! Thanks Max
  I\'m Thanks for that you taking the time to write it.

- Thu, 2 Sep 2004 17:39:13 +0200
  Excellent document. May I suggest that you make it downloadable in PDF (or ASCII) format?
  [Max, 04-10-22: Sorry, as (unfortunately!) I didn't use DocBook or sth. like that, I don't have any other version than this HTML/PHP.
  For PDF, I'd suggest to "Print to file" and convert the PS to PDF.
  For ASCII, I'd suggest to run `'lynx -dump http://homex.subnet.at/~max/ > LDAP-Howto.txt'`.]

- Fri, 1 Oct 2004 01:38:42 +0200
  Thanks for this! It\'s really cleared a few things up for me!

- Wed, 6 Oct 2004 17:48:45 +0200
  Great article; confirms by ldap set up is reasonable. I did get root\'s passwd usage fixed because of this document though.
  FYI: all links to mandrakesecure.net are broken; they seemed to have re-arranged their site and dropped all docs.

- Fri, 15 Oct 2004 00:46:47 +0200
  Thanks, this is a very good document.

- Thu, 4 Nov 2004 03:23:06 +0100
  just great. Thanks a lot

- Thu, 18 Nov 2004 06:32:25 +0100
  what do you mean by this --> add "--with-ldapsam \" just before "--with-msdfs"
  yasanthau@pabnk.lk
  [Max, 04-12-08: By adding the line "--with-ldapsam \" in the rules file, you define that Samba is to be compiled with the capability to host its users-database using the LDAP-server. If you compiled Samba manually (using './configure' etc.), you add such options to the configure-script to state which functionality is to be included (and which is not).]

- Mon, 22 Nov 2004 17:11:06 +0100
  I tried to have login working with LDAP, and it works now.
  The crucial point i met is that pam_ldap is OK to verify the user password, but can\'t do anything to tell the system what uidnumber, gidnumber, shell and home dir to use.
  Am i misleding? (Is it possible to have it working with pam_ldap AND WITHOUT nss_ldap?)
  p.harmel@afpamp.org
  [Max, 04-12-08: Removed multiple postings.
  As to my knowledge, you will need both: PAM on the one hand is responsible for "authentication" (meaning to check whether a user is allowed to login using the given password, the system settings, etc.). NSS on the other hand is responsible to perform the "lookups" for login-name/uidnumber, shell, etc. Just with a normal NIS-solution, you can't have one working properly without the other.]

- Thu, 25 Nov 2004 16:33:20 +0100
  due to colour layout I decided to NOT read this page. its too exhausting to read this.
  newspapers have dark letters on light paper. thats the wayit should be
  [Max, 04-12-08: Since this seems to be an issue: I'll change the style-sheet as soon as I've got some time for "eye-candy" again :) ...]
  [Max, 04-12-22: I changed the style-sheet to something, uhm, more common ... hope this helps ...]

- Thu, 25 Nov 2004 17:37:29 +0100
  >> due to colour layout I decided to NOT read this page. its too exhausting to >> read this.
  guess u don´t have to :-)

- Tue, 14 Dec 2004 17:49:35 +0100
  Thank you for your great writeup!
  I was reading through the changelog.Debian for slapd (Sarge) and it seems unclear if the problems with ldap and GNUTLS/OpenSSL-TLS have been resolved in the current version. Do you know what the current (Dec/04) state is? Thanks again for this document! David.
  [Max, 05-01-03: Apart from slapd having support for GNUTLS included (see `/usr/share/doc/slapd/README.Debian`, I don't know anything about Sarge yet. Are there *known problems with slapd's SSL-support?*

- Thu, 16 Dec 2004 21:20:55 +0100
  Update to my last comment -
  Things have signifcantly changed with LDAP under Debian Sarge. DO NOT follow this howto blindly if you are running testing/unstable. In particular, do not delete the default directories and files. Your LDAP install will just not work and you will have to apt-get remove --purge and reinstall. In

slapd.conf the database line must be the full path to /var/lib/ldap/.... and the module you want to use. This is still a very intersting write up and the suggestions on how to NOT run ldap as root are very valuable - but as the title at top says, this is for WOODY, Not Sarge.
[Max, 04-12-22: Thanks for your information! I'm going to install a Sarge-based LDAP-server soon and am already wondering what it will be like ...]
[Max, 05-06-25 -- Update: As you can see, I added some notes on pre-release version of Sarge I worked with. Though I didn't complete the full process, the basics worked, and I tried to stick with the default packages and configurations ...]

- Sat, 25 Dec 2004 12:55:56 +0100
  Hey,
  I\'m planning to get a Debian Sarge fileserver running in the next few weeks, serving files via FTP and Samba to a bunch of clients (not as a Domain Controller). Because I want all the login/user stuff in LDAP I\'m looking around for information. I\'m not new to LDAP, but this page provides a lot of information, and I see you\'re maintaining it. Thanks! If you appreciate, I can send you some of my experiences.
  Greetings, Hans van Kranenburg (The Netherlands)

- Sun, 26 Dec 2004 14:44:56 +0100
  Very informative, I\'d bookmarked this, partly because I knew I\'d need to refer to some parts of it later, partly because the colour scheme was difficult to read (which I noticed you\'ve changed), and was exhausted!
  Almost didn\'t recognise the site when I re-visited, although a welcome change to the colour scheme! It\'s great to see you\'re maintaining this HOWTO, Keep up the effort!
  syzygy.au

- Wed, 16 Feb 2005 23:17:37 +0100
  Great documentation thank you very much. I have met some terrible documentations assuming that i knew everything and i had every tool. Your doc is JUST GREAT.
  gchelmis@aueb.gr

- Fri, 1 Apr 2005 06:31:26 +0200
  Amazing, After trying multiple times, using diffrent HOWTO\'s yours is the only one that worked and was up to date. Thanks!

- Fri, 1 Apr 2005 07:44:13 +0200
  Hey, what am I doing here? Some guy posted a link, and I just followed it, being the follower kind of person that I am, but anyways. Yeah I figured I\'d type something here as to be useful. Maybe.

- Fri, 29 Apr 2005 00:44:55 +0200
  I have look briefly at alot of documentations on this subject, but they were piecely done. Your guide may not be up to date on latest Samba 3; however, it was easy to read interms of the fonts/layout/and style - to the point and concise. You explained things clearly. I wish Linux documentations had this in general. Layout (look and feel) counts! Great job and thank you.

- Wed, 18 May 2005 09:14:02 +0200
  JXplorer (http://pegacat.com/jxplorer/) is a LDAP browser deluxe.
  Peter Hopfgartner

- Tue, 24 May 2005 20:08:29 +0200
  Max,
  Tanks a lot for this doc! I\’ve been looking for a tutorial like this. It\’s really works. Tank you.
  Claudio Lobo (from Brazil)

- Tue, 21 Jun 2005 15:53:48 +0200
  I would like to thank you for this great guide,
  It really helped me out after a recent sarge install,
  had everything up & running in about an hour (samba with ldap authentication)
  Great job! Thanks!

- Thu, 23 Jun 2005 10:22:37 +0200
  According to \"./sign.sh new.cert.csr\"
  I would like to ask what is ./sigh.sh?
  thanks
  [Max, 05-06-25: This script is used to sign the "certificate signing request" to have a signed certificate be created. Here’s a copy of sign.sh.]

- Thu, 23 Jun 2005 12:54:13 +0200
  Thanks for a great document! Just set up ldap on sarge (stable). It seemed to be a lot of work until I read this document. Also, the libnss-ldap and libpam-ldap seem to support ssl without a hitch.
  Kalle Happonen