# OpenLDAP mit Samba als PDC (neues backend)

## Grundinformation

Verwendetes System: Debian 6.0

Seit Version 2.4 arbeitet OpenLDAP mit einem Onlinebackend. Das erlaubt es die Konfiguration im Betrieb zu bearbeiten ohne den Dienst neu starten zu müssen. Alle Konfigurationen liegen seit dieser Änderung unter `/etc/ldap/slapd.d/`, und nicht mehr in der Datei `slapd.conf`. Man sollte sich von der sichtbaren Klartextstruktur dieser Datenbank-Datei(en) nicht in die Irre leiten lassen. Jede Datei unterhalb von `slapd.d/` enthält zusätzlich alle erforderlichen operationellen Attribute, die für die online- Funktionalität der Online-Konfiguration unabdingbar sind, allen voran natürlich der `modifyTimestamp`. Er sorgt dafür, dass jede online durchgeführte Änderung sofort aktiv wird. Von einer direkten Manipulation unterhalb von `slapd.d/` per Editor ist in jedem Fall abzuraten. Da die Timestamps auf diesem Weg nicht aktualisiert werden.

Eine Änderung von Attributwerten in der Online-Konfiguration sollten stets über **ldap*-Befehle** erfolgen, ob dies nun manuell per LDIF-Files und den entsprechenden Kommandozeilentools oder über ein grafisches Werkzeug geschieht, spielt keine Rolle.

**Ziel:** Ziel ist es eine schöne Weboberfläche namens Ldap-Account-Manager mit dem man bequem User für Windows und Linux im gesamten Netzwerk nur einmal anlegt, Hosts und Domänen zuweisen kann, ACLs vergben, und wenn man möchte LDAP auch als Adressdatenbank nutzen. Ein langer weg, aber schaffbar.

## Installation

```
aptitude install openldap-utils ldap-server samba smbclient samba-doc samba-tools smbldap-tools finger phpldapadmin ldap-account-manager libsasl2-modules-gssapi-mit sasl2-bin libsasl2-modules-ldap
```

Es wird hier nur das Passwort abgefragt. Alles weitere muss man erst selber konfigurieren.

# Konfiguraton von LDAP 2.4

Die `slapd.d/`-Konfiguration wird - wie ein normaler Tree - in Form eines LDAP-Verzeichnisbaumes gespeichert. Die verwendeten Objektklassen und Attribute (die allesamt üblicherweise mit olc* beginnen) werden dabei über ein vordefiniertes bzw. hartverdrahtetes Schema (cn=schema) unseres slapd zur Verfügung gestellt. z.B.

| | |
|---|---|
| Alte slapd.conf Direktive | **rootdn cn=ldapadmin,dc=local,dc=site** |
| Neue slapd.d/ Direktive | **olcRootDN: cn=ldapadmin,dc=local,dc=site** |

Um einen Account zu erstellen muss man sich zuerst eine brauchbare Konfiguration schreiben, die spielt man dann direkt in die Onlinekonfiguration ein. Man erstellt sich also eine Datei; vorzugsweise nennen wir sie `slapd.conf` und fügt folgende brauchbare Zeilen ein. Im Verzeichnis `/etc/ldap/slapd.d/` besteht ein symbolischer Link der auf `slapd.conf` zeigt. Vorerst setzen wir noch ein verschlüsseltes Passwort mit

```
slappasswd -s <Klartextpasswort>
{SSHA}7/esxW/+XxfabHAu6zDfd32ap3/BBJFM+8U
```

Den verschlüsselten Teil trägt man dann auch in die slapd.conf ein. `rootpw {SSHA}7/esxW/+XxfabHA4563bDfd32ap3/BBJFM+8U`

slapd.conf

```
<#allow bind_v2

# Schema and objectClass definitions
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/samba.schema
include         /etc/ldap/schema/collective.schema
include         /etc/ldap/schema/corba.schema
include         /etc/ldap/schema/duaconf.schema
include         /etc/ldap/schema/dyngroup.schema
include         /etc/ldap/schema/java.schema
include         /etc/ldap/schema/misc.schema
include         /etc/ldap/schema/openldap.schema
include         /etc/ldap/schema/ppolicy.schema


# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile         /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile        /var/run/slapd/slapd.args
```

```
# Read slapd.conf(5) for possible values
loglevel        none

# Where the dynamically loaded modules are stored
modulepath      /usr/lib/ldap
moduleload      back_hdb

# The maximum number of entries that is returned for a search operation
sizelimit 500

# The tool-threads parameter sets the actual amount of cpu's that is
used
# for indexing.
tool-threads 1

#######################################################################
# Specific Backend Directives for hdb:
# Backend specific directives apply to this backend until another
# 'backend' directive occurs
backend         hdb

#######################################################################
# Specific Backend Directives for 'other':
# Backend specific directives apply to this backend until another
# 'backend' directive occurs
#backend                <other>

database        config
rootdn          cn=config
rootpw          {SSHA}7/eWD/+XxW9bHAu6zDoAGp3/BBJFM+8U

#######################################################################
# Specific Directives for database #1, of type hdb:
# Database specific directives apply to this databasse until another
# 'database' directive occurs
database        hdb

# The base of your directory in database #1
suffix          "dc=darkwolf,dc=lan"

# rootdn directive for specifying a superuser on the database. This is
needed
# for syncrepl.
# rootdn          "cn=ldap-dc1,dc=darkwolf,dc=lan"

# Where the database file are physically stored for database #1
directory       "/var/lib/ldap"

# The dbconfig settings are used to generate a DB_CONFIG file the first
# time slapd starts.  They do NOT override existing an existing
DB_CONFIG
```

```
# file.  You should therefore change these settings in DB_CONFIG
directly
# or remove DB_CONFIG and restart slapd for changes to take effect.

# For the Debian package we use 2MB as default but be sure to update
this
# value if you have plenty of RAM
dbconfig set_cachesize 0 2097152 0

# Sven Hartge reported that he had to set this value incredibly high
# to get slapd running at all. See http://bugs.debian.org/303057 for
more
# information.

# Number of objects that can be locked at the same time.
dbconfig set_lk_max_objects 1500
# Number of locks (both requested and granted)
dbconfig set_lk_max_locks 1500
# Number of lockers
dbconfig set_lk_max_lockers 1500

# Indexing options for database #1
index           objectClass eq
# Save the time that the entry gets modified, for database #1
lastmod         on

# Checkpoint the BerkeleyDB database periodically in case of system
# failure and to speed slapd shutdown.
checkpoint      512 30

# Where to store the replica logs for database #1
# replogfile    /var/lib/ldap/replog

# The userPassword by default can be changed
# by the entry owning it if they are authenticated.
# Others should not be able to see it, except the
# admin entry below
# These access lines apply to database #1 only
access to attrs=userPassword,shadowLastChange
        by dn="cn=admin,dc=darkwolf,dc=lan" write
        by anonymous auth
        by self write
        by * none

# Ensure read access to the base for things like
# supportedSASLMechanisms.  Without this you may
# have problems with SASL not knowing what
# mechanisms are available and the like.
# Note that this is covered by the 'access to *'
# ACL below too but if you change that as people
```

```
# are wont to do you'll still need this if you
# want SASL (and possible other things) to work
# happily.
access to dn.base="" by * read

# The admin dn has full write access, everyone else
# can read everything.
access to *
        by dn="cn=admin,dc=darkwolf,dc=lan" write
        by * read>
```

Als nächstes müssen wir sicher gehen das auch alle erforderlichen Schemas in `/etc/ldap/schema` vorhanden sind. Wenn wir uns das Verzeichnis ansehen werden wir feststellen dass, das Schema für Samba fehlt. Dies besorgen wir uns aus dem zuvor installierten samba-doc Paket.

```
gunzip /usr/share/doc/samba-doc/examples/LDAP/samba.schema.gz
cp -v /usr/share/doc/samba-doc/examples/LDAP/samba.schema /etc/ldap/schema
```

Den Tree durchsuchen ohne Passwort, nur Daten abfragen

```
ldapsearch -x -b dc=darkwolf,dc=lan '(objectclass=*)'
```

Den Tree durchsuchen und gleichzeitig das Passwort testen

```
ldapsearch -x -W -D cn=admin,dc=darkwolf,dc=lan '(objectclass=*)'
```

Gefunden wird logischer weise nichts, da ja noch keine Einträge in „admin“ vorhanden sind. Aber wir haben ein fehlerfreies Suchergebnis.

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 2
result: 32 No such object

# numResponses: 1
```

## Erzeugen und Einspielen des Adminusers

So, jetzt müssen wir die `slapd.conf` nur mehr in das Onlineprofil übernehmen. Der Vorgang

gestaltet sich relativ einfach:

```
/etc/init.d/slapd stop
mv slapd.d slapd.d.bck
mkdir slapd.d
slaptest -f slapd.conf -F slapd.d
chown -R openldap:openldap slapd.d
/etc/init.d/slapd start
```

Nun könnten wir die `slapd.conf` sorglos löschen. Von dem ist aber abzuraten, da wir bei späteren Konfigurationsänderungen diese mit dieser Datei leichter vollziehen können. Wir können nun beginnen Daten einzutragen.

# LDAP mit Daten füttern

Daten mit Hilfe einer LDIF einspielen

```
ldapadd -f database.ldif -x -D "cn=admin,dc=darkwolf,dc=lan" -W
```

Mit Hilfe eines Webminterface (empfohlen)

phpLDAPadmin (also known as PLA) is a web-based LDAP client. It provides easy, anywhere-accessible, multi-language administration for your LDAP server. Its hierarchical tree-viewer and advanced search functionality make it intuitive to browse and administer your LDAP directory. Since it is a web application, this LDAP browser works on many platforms, making your LDAP server easily manageable from any location. phpLDAPadmin is the perfect LDAP browser for the LDAP professional and novice alike. Its user base consists mostly of LDAP administration professionals. Aufrufen lässt sich das Interface unter http://localhost/phpldapadmin

**Konfiguration**

- /etc/phpldapadmin/config.php

LDAP Account Manager ist ein Webfrontend für die Verwaltung diverser Kontotypen in einem LDAP-Verzeichnis. Es wurde in PHP geschrieben. Im Gegensatz zu Programmen wie phpLDAPadmin liegt der Schwerpunkt auf der kontobasierten Verwaltung. Der Benutzer erhält eine abstraktere Sicht auf das LDAP-Verzeichnis. Das Programm ist unter der GNU General Public License lizenziert. Aufrufen lässt sich das Interface unter http://localhost/lam

**Konfiguration**

- direkt im Webinterface
- /var/lib/ldap-account-manager/config/lam.conf, oder eben das verwendete Profil

# Spezielle Schemen

Dies wäre z.B. die Hostextension. Mit ihr ist es möglich User nur auf bestimmte Hosts zu zulassen.

Hierfür sind die zwei folgenden Schemen notwendig.

- ldapns.schema (für dieses Schema wird clientseitig immer das Paket libpam-ldap benötigt, da sonst der Hostfilter nicht beachtet wird, pam muss man bei der Installation die Dateien common-* überschreiben lassen) Der Filter zieht aber nicht mehr sobald ein SSH Schlüssel hinterlegt wurde.
- openssh-lpk_openldap.schema

# Verschlüsselter Datenaustausch mit TLS

TLS-Konfiguration mit SASLMech EXTERNAL

Transport Layer Security (TLS), weitläufiger bekannt unter der Vorgängerbezeichnung Secure Sockets Layer (SSL), ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet. Seit Version 3.0 wird das SSL-Protokoll unter dem neuen Namen TLS weiterentwickelt und standardisiert, wobei Version 1.0 von TLS der Version 3.1 von SSL entspricht. Im OSI-Modell ist TLS in Schicht 6 (der Darstellungsschicht) angeordnet. Im TCP/IP Modell ist TLS oberhalb der Transportschicht (zum Beispiel TCP) und unterhalb Anwendungsprotokollen wie HTTP oder SMTP angesiedelt. In den Spezifikationen wird dies dann zum Beispiel als „HTTP over TLS“ bezeichnet. Sollen jedoch beide Protokolle zusammengefasst betrachtet werden, wird üblicherweise ein „S“ für Secure dem Protokoll der Anwendungsschicht angehängt (zum Beispiel HTTPS). TLS arbeitet transparent, so dass es leicht eingesetzt werden kann, um Protokollen ohne eigene Sicherheitsmechanismen abgesicherte Verbindungen zur Verfügung zu stellen. Zudem ist es erweiterbar, um Flexibilität und Zukunftssicherheit bei den verwendeten Verschlüsselungstechniken zu gewährleisten. Im ersten Schritt benötigen wir Zertifikate, OpenSSL hilft uns dabei. Um eine verschlüsselte Verbindung zwischen der Client-Applikation und dem Server aufzubauen, benötigen wir in unserer Konfiguration mehrere Dinge: das eigentliche Zertifikat, den Schlüssel und eine sogenannte **Certification Authority (CA)**. Diese CA **(oder auch rootCA)** wird benötigt, um den unsignierten Zertifikats-Request zu signieren und damit ein deute zu beglaubigen; Marke: Du kriegt jetzt einen Stempel und Unterschrift - **test and approved**.

```
cd /etc/ssl
cp openssl.cnf openssl.cnf-orig
```

Das Konfig muss abgeändert werden.

```
[ policy_match ]
# countryName wurde deaktiviert, domainComponent hinzugefügt
domainComponent = match
...
[ policy_anything ]
# countryName wurde deaktiviert, domainComponent hinzugefügt
domainComponent = optional
...
[ req_distinguished_name ]
# countryName wurde deaktiviert, domainComponent hinzugefügt
# die 0. und 1. vor den Einträgen bewirken eine multible
# Verwendbarkeit und Verkettung der Parameter --> darkwolf.lan
...
```

```
0.domainComponent               = TLD Domaenen-Komponente (dc=lan)
0.domainComponent_default       = lan
1.domainComponent               = Zweite Domaenen-Komponente (dc=darkwolf)
1.domainComponent_default       = darkwolf
stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Oesterreich
localityName                    = Locality Name (eg, city)
localityName_default            = Gnas
0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = Open Source IT
...
commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64
# Der Parameter '''commonName''' ist einer der wichtigsten, da TLS den DN
aus dem
# Zertifikat ausliest. Stimmt der DN nicht mit dem echten
'''FQHN/Usernamen'''
# überein, kommt es zu Problemen!
```

**Generieren des Zertifikates:**

Zuerst wird nach einem Zertifikatsnamen und einem Passwort gefragt. Hier etwas beliebiges ausdenken und eingeben. Danach sollte man alles mit ENTER bestätigen können. Am Ende wird man dann wieder nach dem zuvor vergebenen Passwort gefragt.

```
cd /usr/lib/ssl/misc
./CA.pl -newca
```

- Das Zertifikat liegt unter `/usr/lib/ssl/misc/demoCA/cacert.pem`
- Der Schlüssel liegt unter `/usr/lib/ssl/misc/demoCA/private/cakey.pem`

Da wir nun stolze Besitzer einer eigenen CA sind, mit der wir unserer Zertifikate signieren können.

## Erzeugen des Zertifikat-Requests

Nun können wir uns als Nächstes an die Erstellung des eigentlichen Zertifikates machen. Das läuft in zwei Schritten ab. Zunächst erstellen wir ein unsigniertes Zertifikat (einen so genannten `Zertifikats-Request`), welches anschließend mit unserer CA signiert und somit beglaubigt wird.

```
./CA.pl -newreq
```

Das Vorgehen und die Eingaben gestalten sich analog zum vorherigen Schritt. Da wir jedoch nun einen expliziten Zertifikats-Request für unseren Server `ldap-dc.darkwolf.lan` erstellen, muss der **Common Name** im Zertifikats-DN auch exakt so lauten. Die letzte Zeile des Dialogs sagt uns, wo wir den Zertifikats-Request finden können, nämlich in der Datei `newreq.pem`

```
Request is in newreq.pem, private key is in newkey.pem
```

## Signieren des Zertifikates

```
./CA.pl -sign
```

Hierbei signieren wir unseren eben erstellten Zertifikats-Request (newreq.pem) mit unserer CA, am Ende erhalten wir das signierte Zertifikat `newcert.pem`. Als Passwort benötigen wir das des Schlüssels des CA und bestätigen die beiden Signierungs-Abfragen mit [y] (yes).

```
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
```

## Extraktion der Passphrase

Wir müssen nun das Passwort aus dem Zertifikats-Request bzw. der Request-Key-Datei extrahieren, da wir sonst bei jedem Start von Slapd das Zertifikat beglaubigen und das Passwort eingeben müssten. Der Passworteintrag wird bei diesem Vorgang in die `ldapkey.pem` geschrieben.

```
cd /usr/lib/ssl/misc
openssl rsa -in newkey.pem -out ldapkey.pem
```

Um alles schön übersichtlich zu halten, legen wir nun unterhalb unseres eigentlichen OpenLDAP-Konfigurationsordners (/etc/ldap) einen Ordner names `'certs'` an. Dorthin kopieren wir die zuvor generierten Dateien: `cacert.pem, newcert.pem` und `ldapkey.pem`. Anschließend nicht vergessen, den Ordner und die Zertifikate für den User/die Gruppe lesbar zu machen, mit deren Rechten unser slapd läuft.

```
mkdir /etc/ldap/certs
cp /usr/lib/ssl/misc/demoCA/cacert.pem /etc/ldap/certs/
cp /usr/lib/ssl/misc/newcert.pem /etc/ldap/certs/
cp /usr/lib/ssl/misc/ldapkey.pem /etc/ldap/certs/
chown -R openldap:openldap /etc/ldap/certs/
chmod -R 440 /etc/ldap/certs/
chmod +x /etc/ldap/certs/
```

## Anpassung der slapd.conf auf unserem ldap-dc.darkwolf.lan für die Verwendung von TLS

Folgende Einträge müssen wir in der slapd.conf (globale Sektion) vornehmen damit LDAP auch verschlüsselt kommuniziert.

```
TLSCertificateFile      /etc/ldap/certs/newcert.pem
TLSCertificateKeyFile   /etc/ldap/certs/ldapkey.pem
TLSCACertificateFile    /etc/ldap/certs/cacert.pem
TLSVerifyClient         allow
# Verfahren fuer Handshake, sollte auf demand gesetzt sein
```

'`Allow`' besagt, dass auch für den Fall, dass kein oder ein ungültiges Zertifikat vom Client kommt, die Ausführung trotzdem fortgesetzt wird. Sobald die TLS-Funktionalität ausgiebig und erfolgreich getestet wurde, sollte der Parameter später in jedem Fall auf '`demand`' umgestellt werden.

### Weitere Server-/Client-/User-Zertifikate und Anpassung

Die Erstellung aller weiteren Zertifikate entspricht den bereits in den Schritten 6.1, 6.2 und 6.3 beschriebenen Vorgehensweisen. Erstellen wir nun gleich unseren Client `tftp.darkwolf.lan`. Wichtig ist das der `Common Name` jedes weiteren Servers/Client/Users entsprechend seines **FQHN's bzw. LDAP-cn's** angepasst wird.

**Suchanfragen mit TLS:**

```
ldapsearch -x uid=ml -b dc=darkwolf,dc=lan '(objectclass=*)' -ZZ

ldapsearch -d -1 -x uid=ml -b dc=darkwolf,dc=lan '(objectclass=*)' -ZZ

ldapsearch -h ldapserver.local -ZZ -x -D "cn=Manager,dc=darkwolf,dc=lan" -W

ldapsearch -h ldapserver.local -ZZ -x -D "cn=Manager,dc=darkwolf,dc=lan" -W
-d 255
```

## Linux Clientanbindung an LDAP

Bei der Installation der Pakete werden einige Daten wie cn des Administrators der LDAP-Datenbank abgefragt, ebenso wie Hostname/IP des LDAP-Servers. Diese können auch jederzeit nach der Installation verändert werden. Daher ist es am einfachsten, während der Installation die Abfragen einfach mit ⏎ zu bestätigen.

### Installation LDAP-Client

```
aptitude install libnss-ldap libpam-ldap nscd portmap finger libpam-
foreground
```

Die Pakete nscd und portmap sind Abhängigkeiten, welche unerlässlich sind, um sich mit einem LDAP Server zu authentifizieren. Sie benötigen keine weitere Konfiguration.

| Ausdruck | Beispielwert | Kommentar |
|---|---|---|
| Host | ldap-dc.darkwolf.lan | Die Host-Adresse vom LDAP Server, die IP (hier 10.0.0.2) statt des Hostnamens beschleunigt die Authentifizierung geringfügig. |
| BASEDN | dc=darkwolf,dc=lan | Suchbasis auf dem Server (engl. Base Distinguished Name) |
| ROOTBINDDN | cn=admin,dc=darkwolf,dc=lan | Der LDAP Server Administrator Account Name |
| ldap_version | 3 | Heute meist Version 3, allerdings ist Version 2 des LDAP Protokolls noch vertreten. |

| Ausdruck | Beispielwert | Kommentar |
|---|---|---|
| secret | geheimesPW | das LDAP - Passwort des LDAP - Administrator Accounts (ROOTBINDDN) |

## Konfiguration des Client

PAM (Pluggable Authentication Modules [for Linux]) ist eine praktische Sammlung von Modulen, die die Authentifizierung für Benutzer, aber auch Maschinen oder Dienste übernehmen. Folgend werden einige Module dahingehend verändert, damit eine erfolgreiche Benutzerauthentifizierung mittels LDAP möglich und effizient wird. Wenn ein Verzeichnis (/etc/pam.d/) existiert ist in der Regel /etc/pam.conf leer, da alles modular konfiguriert wird, um individuell auf Wünsche der Authentifizierung eingehen zu können. Damit die Authentifizierung funktioniert, müssen die bestehenden Einträge in /etc/pam.d/common-* auskommentiert und folgende hinzugefügt werden:

```
nano /etc/pam.d/common-auth
auth    sufficient      pam_ldap.so
auth    required        pam_unix.so use_first_pass nullok_secure
```

`auth` ist für die Sicherheit zuständig, ob der User, der er vorgibt zu sein auch wirklich dieser ist und somit ein Passwort verlangt.

```
nano /etc/pam.d/common-account
account sufficient      pam_ldap.so
account required        pam_unix.so use_first_pass
# account [default=bad success=ok user_unknown=ignore] pam_ldap.so | Diese
Eintrag sollte genommen werden falls man in der /etc/ldap.conf mit den
Optionen pam_groupdn und pam_member_attribute arbeitet, da sonst
# diese Optionen nicht greifen.
```

`account` prüft die Berechtigung, ob ein User den Dienst benutzen darf, oder ob vielleicht sein Passwort abgelaufen ist.

```
nano /etc/pam.d/common-password
password   sufficient pam_ldap.so
password   required   pam_unix.so use_first_pass nullok obscure min=4 max=8
md5
```

`password` spezifiziert, wie ein Passwort gewechselt wird und wie dieses zusammengesetzt werden muss.

```
nano /etc/pam.d/common-session
session required        pam_unix.so
session optional        pam_foreground.so
```

`session` übernimmt alle wichtigen Prioritätsaufgaben und regelt die Prozesse wie das zur Verfügung stellen des Heimatverzeichnisses.

```
nano /etc/ldap/ldap.conf
HOST    192.168.1.5 # IP des LDAP Servers
```

```
BASE    dc=darkwolf,dc=lan # BASE DN
```

Die meisten Daten wurden vom Assistenten schon eingebracht. Sie sollte durchgesehen und event. Ergänzungen eingebracht werden. In der ldap.conf sind die wichtigsten LDAP-Angaben untergebracht, die weit ins Detail verfeinert werden können, zunächst reicht es jedoch wenn, dort die Adresse des LDAP-Servers und die Suchbasis, in der man sich authentifizieren will, angegeben wird. Durch das Konfigurationsscript wird die URI des LDAP-Servers verlangt. Die Vorgabe in dieser Zeile lautet ldapi:/// anschließend wird die Adresse (IP:Port) des LDAP-Servers verlangt, dabei muss **ldapi:*/ auf ldap:*** geändert werden.

## NSS - Modul - Konfiguration

NSS (System Databases and Name Service Switch configuration) gibt im System an, wo sich welche Datenbanken (Aliase, Benutzer, Gruppen, Netzwerke…etc.) befinden und in welcher Reihenfolge sie abgefragt oder genutzt werden sollen. So kann man für jede Datenbank individuell angeben wie der LookUp Prozess funktionieren soll. (/etc/nsswitch.conf)

```
nano /etc/libnss-ldap.conf
host 192.168.1.5
base dc=darkwolf,dc=lan
ldap_version 3
```

Auch hier werden die Suchbasis und der Host angegeben.

### Vor der nächsten Änderung ist abzuraten!!!

Falls Sie bemerkt haben, dass eine der Zeilen, die Sie in Ihre /etc/ldap.conf kopiert haben, auskommentiert war (die rootbinddn Zeile): Sie brauchen sie nicht, außer Sie wollen das Passwort eines Benutzers als Superuser ändern. In diesem Fall müssen Sie das root-Passwort im Klartext in die Datei /etc/ldap.secret schreiben. Dies ist GEFÄHRLICH und sollte chmod 600 erhalten. Von dem ist also abzuraten. Richtiger weise habe ich es aber dokumentiert.

```
nano /etc/libnss-ldap.secret
geheimesPW in Klartext

chmod 640 /etc/libnss-ldap.secret
```

Das Passwort sollte auch hier in einer Datei gespeichert sein, die vor fremden Zugriffen geschützt ist!

### nano /etc/nsswitch.conf

Nachdem man die nsswitch.conf editiert hat, könnte kein sudo mehr erfolgreich sein, wenn man sich vertan hat oder etwas Anderes nicht stimmt, denn dann möchte Ubuntu bereits auf den LDAP-Server zugreifen. Sorgfalt ist also geboten. Wenn man sich trotz aller Vorsicht ausgesperrt, hilft nur noch ein Start im Rettungsmodus oder von einer Live-CD um den Fehler zu korrigieren oder die alte nsswitch.conf wieder herzustellen. Im Artikel Recovery Modus ist auch ein weiterer Hinweis, wie man an das Root-Filesystem gelangt, um Fehler zu korrigieren oder Einträge rückgängig zu machen.

```
nano /etc/nsswitch.conf

passwd:         files ldap
```

```
group:          files ldap
shadow:         files ldap
hosts:          files dns
networks:       files
```

### Erster Login und Clientdebugging

Zuerst sollte man noch die erforderlichen Dienste neu laden oder den PC/VM neu starten.

```
/etc/init.d/nscd restart
service portmap restart
/etc/init.d/libnss-ldap restart
```

Jetzt ist es an der Zeit, die Konfiguration zu testen.

```
getent passwd
```

Es sollten die Benutzeraccounts vom LDAP-Repository angezeigt werden.

Funktionierts? Dann:

```
su <benutzerID des LDAP-Servers>
```

Wird man nun nach einem Passwort gefragt, ist die LDAP-Authentifizierung schon aktiv. Ausser natürlich bei root.

## LDAP mit SAMBA als PDC

Als ersters übergeben wir Samba das LDAPpasswort. Dies wird verschlüsselt in einer Datenbankfile gespeichert.

```
smbpasswd -w <ldappasswort>
Setting stored password for "cn=admin,dc=darkwolf,dc=lan" in secrets.tdb
```

Mit der Installation des nächsten Tools hat man die Möglichkeit auch von einem Windowsrechner sein Passwort wie gewohnt zu ändern. Diese Änderungen gelten dann für das LDAP.

```
aptitude install make gcc libc-dev
wget http://www.nomis52.net/data/mkntpwd.tar.gz
tar zxvf mkntpwd.tar.gz
cd mkntpwd
make
cp mkntpwd /usr/local/bin/
```

Nachdem man folgende Beispieldateien konfiguriert hat, kann man sich die Struktur im LDAP anlegen lassen. Hierbei wird auch die Domäne angelegt.

```
zcat /usr/share/doc/smbldap-tools/examples/slapd.conf.gz
/usr/share/doc/smbldap-tools/smbldap-tools.pdf.gz
/usr/share/doc/smbldap-tools/configure.pl.gz
```

```
smbldap-populate
```

Am Ende wird man gefragt nach einem neue Rootpasswort. Dies wäre ein LDAP-Rootuser z.B. zum Hinzufügen eines Windowsrechner zur Domäne. Hier sollte man vorzugsweise das gleiche Passwort wie für den LDAPserver verwenden. Den User sollte man auch nur bei Verwendung aus dem Webinterface aktivieren.

Mit dem nächsten Befehlt sieht man ob man Zugriff auf die Datenbank hat, und wie die SID des Domänencontrollers heist.

```
net getlocalsid
net getdomainsid
```