Installing & Configuring OpenLDAP

Technical White Paper

PROCESS™ SOFTWARE

29-SEP-2005/MLL

Introduction

Despite what most books on the subject would lead you to believe, LDAP really isn't that difficult to use. The best way to learn LDAP is by running your own LDAP server. This white paper covers the installation and basic configuration of the OpenLDAP LDAP server.

Lots of sites use OpenLDAP in preference to commercial solutions because:

- It's free. Most IT departments have a fixed yearly budget, and system administrators would rather spend the little money they get on new hardware rather than software they can get for free.
- It's open source. Anybody can download and look at the guts of the server, and you can also change how any part of the server works to fit your needs.
- It's compliant with the latest version of the LDAP protocol: LDAP Version 3 (LDAPv3).
- OpenLDAP is open source and written in C, so it can be compiled to run on almost any platform your site has lying around: Linux, Solaris, Tru64, VMS, Windows, MacOS, OpenBSD, HP-UX, etc.

Getting & Installing Prerequisites

Like most large open source software packages, OpenLDAP depends on several other open source packages to run. A database to store the directory data in is required, and most sites will want their LDAP server to support Transport Layer Security (TLS). TLS provides end-to-end encryption of data being sent to and from the LDAP server. Since that data might contain user passwords, encryption may be required.

Database

OpenLDAP stores its directory data in indexed files that follow the DBM format. The use of indexed files greatly speeds up searching data, which is what most LDAP servers spend the majority of their time doing. The most common database management software that supports DBM is Berkeley DB, available from SleepyCat Software. You should use the most current version of Berkeley DB that's available. When this white paper was written, the most current version was 4.3.28.

To install Berkeley DB, perform the following steps:

1. Download the most current version of Berkeley DB to your working directory on the system you want to install OpenLDAP on. The download URL is http://www.sleepycat.com/download/index.shtml

2. Uncompress and un-archive the distribution package:

```
$ gzip -d db-4.3.28.NC.tar.gz
$ tar xfv db-4.3.28.NC.tar
db-4.3.28.NC

[...]

db-4.3.28.NC/docs/utility/index.html
db-4.3.28.NC/docs/index.html
$
```

3. Change your current directory to the `db-x.x.xx.NC/build_unix` directory, and run the `configure` script located in the `dist` subdirectory. By default, Berkeley DB is configured to install itself into a non-standard

installation directory. We're going to tell the `configure` script to install Berkeley DB into the `/usr/local/` directory.

```
$ cd db-4.3.28.NC/build_unix
$ ../dist/configure --prefix=/usr/local/
checking build system type... sparc-sun-solaris2.9
checking host system type... sparc-sun-solaris2.9
checking if building in the top-level or dist directories... no
checking if --disable-cryptography option specified... no
checking if --disable-hash option specified... no

[...]

config.status: creating include.tcl
config.status: creating db.h
config.status: creating db_config.h
$
```

4. Compile the Berkeley DB package by running `make`.

```
$ make
/bin/sh ./libtool --mode=compile cc -c -I. -I../dist/..  -D_REENTRANT -O2
../dist/../mutex/mut_pthread.c
mkdir .libs
cc -c -I. -I../dist/.. -D_REENTRANT -O2 ../dist/../mutex/mut_pthread.c  -
fPIC -DPIC -o .libs/mut_pthread.o

[...]

cc -O2 -o .libs/db_verify .libs/db_verify.o .libs/util_cache.o
.libs/util_sig.o  ./.libs/libdb-4.3.so -lrt  -R/usr/local/lib
creating db_verify
/bin/sh ./libtool --mode=execute true db_verify
$
```

5. Install the Berkeley DB package by running `make install` as the root user.

```
$ sudo[1] make install
Installing DB include files: /usr/local/include ...
Installing DB library: /usr/local/lib ...

[...]

cp -p .libs/db_verify /usr/local/bin/db_verify
Installing documentation: /usr/local/docs ...
$
```

[1] The `sudo` command grants selective root access for individual commands, without requiring the use of the root account's password. If you're not using `sudo` on your UNIX systems, you should be.

TLS

While TLS isn't required to run OpenLDAP, the encryption capability it provides is something most sites will want. The freely available OpenSSL package provides TLS libraries suitable for use with OpenLDAP. As you might expect, you should use the most current version of OpenSSL that's available. When this paper was written, the most recent version was 0.9.8.

To install OpenSSL, perform the following tasks:

1. Download the most current version of OpenSSL to your working directory on the system you want to install OpenLDAP on. The download URL is http://www.openssl.org/source/

2. Uncompress and un-archive the distribution package:

```
$ gzip -d openssl-0.9.8.tar.gz
$ tar xfv openssl-0.9.8.tar
openssl-0.9.8/apps/
openssl-0.9.8/apps/app_rand.c

[...]

openssl-0.9.8/VMS/VMSify-conf.pl
$
```

3. Change your current directory to the `openssl-0.x.x` directory, and run the `config` script. Just like we did with Berkeley DB, we're going to tell OpenSSL to install itself in the `/usr/local/` directory. We're also going to instruct it to build shared versions of its libraries, which OpenLDAP is going to need.

```
$ cd openssl-0.9.8
$ ./config shared --openssldir=/usr/local
Operating system: sun4u-whatever-solaris2
Configuring for solaris-sparcv9-gcc

[...]

make[1]: Leaving directory `~/openssl-0.9.8/test'

Configured for solaris-sparcv9-gcc.
$
```

4. Compile the OpenSSL package by running `make`.

```
$ make
making all in crypto...
 ( echo "#ifndef MK1MF_BUILD"; \

[...]

make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `~/openssl-0.9.8/tools'
$
```

5. Install the OpenSSL package by running `make install` as the root user.

```
$ sudo make install
making all in crypto...
make[1]: Entering directory `~/openssl-0.9.8/crypto'

[...]

cp openssl.pc /usr/local/lib/pkgconfig
chmod 644 /usr/local/lib/pkgconfig/openssl.pc
$
```

Downloading & Installing OpenLDAP

Once Berkeley DB and OpenSSL are installed, it's time to download and install the OpenLDAP package. As always, you should use the latest stable release of the software. When this paper was written, 2.2.26 was the newest stable version of OpenLDAP.

To install OpenLDAP, follow these steps:

1. Download the most current version of OpenLDAP to your working directory. The download URL is http://www.openldap.org/software/download/

2. Uncompress and un-archive the distribution package:

```
$ gzip -d openldap-2.2.26.tgz
$ tar xfv openldap-2.2.26.tar
openldap-2.2.26/
openldap-2.2.26/doc/

[...]

openldap-2.2.26/tests/scripts/test020-proxycache
openldap-2.2.26/tests/scripts/test021-certificate
$
```

3. Change your current directory to the `openldap-2.x.x` directory, and run the `configure` script.

```
$ cd openldap-2.2.26
$ ./configure
Configuring OpenLDAP 2.2.26-Release ...
checking host system type... sparc-sun-solaris2.9

[...]

creating include/ldap_features.h
creating include/lber_types.h
$
```

By default, several versions of UNIX and Linux do not search the `/usr/local/` directory for shareable files. That's a problem when installing OpenLDAP, since it depends on the Berkeley DB and OpenSSL shareables that we installed in `/usr/local/`. On Linux, add `/usr/local/lib/` to the file `/etc/ld.so.conf`, then run the `ldconfig -v` command. On other versions of UNIX, set the `LD_LIBRARY_PATH` environment variable to `/usr/local/lib/`.

4. Build several programs that the OpenLDAP build process depends upon by running `make depend`.

```
$ make depend
Making depend in ~/openldap-2.2.26
  Entering subdirectory include

[...]

make[1]: Leaving directory `~/openldap-2.2.26/doc'

$
```

5. Build OpenLDAP by running `make`.

```
$ make
Making all in ~/openldap-2.2.26
  Entering subdirectory include

[...]

make[1]: Leaving directory `~/openldap-2.2.26/doc'

$
```

6. Run the OpenLDAP test suite to make sure the package was built correctly.

```
$ make test
Initiating LDAP tests for BDB...
Running ./scripts/all...

[...]

>>>>> ./scripts/test021-certificate completed OK.
>>>>> waiting 10 seconds for things to exit
$
```

7. Run `make install` as root to install OpenLDAP on your system.

```
$ sudo make install
Making all in ~/openldap-2.2.26
  Entering subdirectory include

[...]

make[1]: Leaving directory `~/openldap-2.2.26/doc'

$
```

Configuring OpenLDAP

OpenLDAP is controlled primarily by the settings in the `slapd.conf` file. (The executable that handles incoming LDAP requests is named `slapd` - it's probably short for "Standalone Lightweight Access Protocol Daemon".) By default, `slapd.conf` is located in the `/usr/local/etc/openldap/` directory. `slapd.conf` contains a list of configuration variables and their values, separated by white space. Blank lines are ignored, as well as lines that begin with a pound (`#`) sign.

Following is a sample `slapd.conf` file that has been split into logical sections. Following every section is a brief explanation of it configuration variables and their possible values. For detailed information about every configuration variable supported by OpenLDAP, see the *OpenLDAP Administrator's Guide* on the OpenLDAP website.

```
# /usr/local/etc/openldap/slapd.conf

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

A schema describes one or more objects that can exist in an LDAP directory. The `core.schema` and `cosine.schema` files include descriptions for very low-level objects[2] that are needed for a bare-bones LDAP directory. The `inetorgperson.schema` file describes the inetOrgPerson object[3] that most LDAP-integrated software uses.

```
loglevel  264
pidfile   /usr/local/var/run/slapd.pid
argsfile  /usr/local/var/run/slapd.args
```

The `loglevel` configuration variable controls the amount of logging that OpenLDAP performs. The possible types of logging are:

Level	Type Of Logging
0	No logging
1	Trace function calls
2	TCP packet-level debugging
4	Detailed trace debugging
8	Connection management
16	Sent and received packets
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Connection, operation, and results statistics
512	Client results statistics
1024	Shell back-end communications
2048	Entry parsing debugging

[2] These base objects are described in RFC 2251, RFC 2252, RFC 2253, RFC 2254, RFC 2255, and RFC 2256.
[3] The inetOrgPerson object is described by RFC 2798.

`loglevel` is an integer bit mask, which means you just add together the level values for each type of logging you want. For example, `loglevel` in our sample `slapd.conf` file is set to 264. A value of 264 means that operational statistics and connection management logging are turned on (256 + 8 = 264).

Setting `loglevel` to a value of -1 turns on all logging. You should only do this if you're trying to debug a problem. Otherwise, OpenLDAP will generate so much log data that it'll swamp the system.

OpenLDAP stores its process ID (PID) in the file named by the `pidfile` configuration variable. Scripts that start and stop the OpenLDAP server use the contents of this file. The `argsfile` variable specifies a file that contains OpenLDAP command-line arguments. The OpenLDAP server will automatically use these arguments every time it's started, which can save you a lot of typing if you want to specify a lot of command line arguments.

```
# TLS options
TLSCipherSuite         HIGH:MEDIUM
TLSCertificateFile     /usr/local/etc/openldap/slapd-cert.pem
TLSCertificateKeyFile  /usr/local/etc/openldap/slapd-key.pem
```

These configuration variables control TLS encryption settings, if you built OpenLDAP with TLS support. `TLSCipherSuite` specifies which encryption methods the server will accept, as well as the order it prefers to accept them in. You can specify either cipher groups (the `HIGH` and `MEDIUM` groups are used in this example), or individual cipher methods (such as `AES:3DES:SHA1:RC4`). Separate each cipher method with a colon.

The `TLSCertificateFile` and `TLSCertificateKeyFile` configuration variables specify the files that contain the server's public certificate and private key, respectively. If you already have TLS certificates[4] for the system you're running OpenLDAP on, you can set these variables to point at them. If you don't have TLS certificates, see *Appendix A - Generating TLS Certificates* at the end of this paper for step-by-step instructions on how to create them.

```
#######################################################################
# BDB database definitions
#######################################################################

database bdb
```

The `database` variable specifies the backend database used by OpenLDAP to store data. The `bdb` module is used to support Berkeley DB, which is what we've built OpenLDAP to support.

```
suffix "dc=apes.process,dc=com"
```

The `suffix` configuration variable specifies the name of the base entry in the directory. All other entries in the directory will be descendents of this object. Unless you're trying to do something unusual, the base entry should be based on the local domain name. In the above example, the LDAP directory would hold entries for the apes.process.com domain.

```
rootdn "cn=Directory Manager,dc=apes.process,dc=com"
rootpw secret
```

Every LDAP directory has a root distinguished name (DN), which is roughly analogous to the root user on a UNIX system. The user specified by the `rootdn` variable can read, write, and search any part of the directory. You can name the root DN anything you want, but `cn=Manager` or `cn=Directory Manager` are commonly

[4] With a few exceptions, SSL certificates can do double duty as TLS certificates.

used. Note that the value of the `suffix` variable is appended to the end of the root DN's name. The `rootpw` configuration variable specifies the password for the root DN account.

```
directory /usr/local/var/openldap-data
```

The `directory` variable specifies the location on the file system where the actual database files that contain directory data are stored. Make sure that this directory exists, and that it's only accessible by the user that OpenLDAP will be running as (running the command **`chmod 700 /usr/local/var/openldap-data`** will set the correct permissions).

```
# Indices to maintain
index   objectClass     eq
```

The `index` variable specifies the object attributes that OpenLDAP should maintain indexes for. An index can greatly reduce the amount of time needed to find an object based on a particular attribute. The default index setting will create an index for the `objectClass` attribute, which should be sufficient for all but the most highly loaded sites. Tuning OpenLDAP indexes is outside the scope of this white paper, but there should be little reason to do so at most sites.

The complete `slapd.conf` configuration file looks like:

```
# /usr/local/etc/openldap/slapd.conf

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

loglevel        264
pidfile         /usr/local/var/run/slapd.pid
argsfile        /usr/local/var/run/slapd.args

# TLS options
TLSCipherSuite          HIGH:MEDIUM
TLSCertificateFile      /usr/local/etc/openldap/slapd-cert.pem
TLSCertificateKeyFile   /usr/local/etc/openldap/slapd-key.pem


#######################################################################
# BDB database definitions
#######################################################################

database        bdb
suffix          "dc=my-domain,dc=com"

rootdn          "cn=Manager,dc=my-domain,dc=com"
rootpw          secret

directory       /usr/local/var/openldap-data

# Indices to maintain
index   objectClass     eq
```

Running OpenLDAP

Now that OpenLDAP is configured for your site, it's time to start it up. Starting OpenLDAP is simple - just run `slapd` as root.

```
$ sudo /usr/local/libexec/slapd
$
```

You can verify that OpenLDAP started by checking the system process listing for the `slapd` process:

```
$ ps -ef | grep slapd
    root 23932     1  0 09:52:03 ?        0:00 /usr/local/libexec/slapd
$
```

The recommended way to shut down OpenLDAP is to send the `slapd` process an interrupt signal (`SIGINT`). This lets `slapd` write out all cached data and exit gracefully. You can get `slapd`'s process ID either by running `ps` (like in the above example) or by looking at the contents of the `/usr/local/var/run/slapd.pid` file.

```
$ sudo kill -INT `cat /usr/local/var/run/slapd.pid`
$
```

You should avoid using `kill -9` to stop `slapd` at all costs. Using drastic means to shut down OpenLDAP will corrupt the directory data.

If you want OpenLDAP to automatically start when the system boots and automatically stop when the system halts, you can create a simple initialization script in the `/etc/init.d` directory. Your script should look something like this:

```
#!/sbin/sh

# NAME: /etc/init.d/openldap

case “$1” in
’start’)
  [ -f /usr/local/libexec/slapd ] && /usr/local/libexec/slapd
  ;;
’stop’)
  kill -INT `cat /usr/local/var/run/slapd.pid`
  ;;
*)
  echo “Usage: $0 { start | stop }”
  exit 1
esac

exit 0
```

Populating An OpenLDAP Directory

An LDAP directory that doesn't contain any data isn't very useful. The `ldapmodify` tool is used to add new entries to the directory or modify existing entries. The details of new entries are specified in a special type of text file called an LDIF file[5]. The full details of LDIF are beyond the scope of this white paper, but most LDAP reference books contain a complete description.

We're going to create an LDIF file that contains an entry for the root node, an entry for an organizational unit named "people", and two `inetOrgPerson` objects. You'll want to alter the `dn` attributes to correspond with the suffix that you specified in the `slapd.conf` file.

```
#NAME: users.ldif
#root node
dn: dc=apes.process,dc=com
dc: apes.process
objectClass: dcObject
objectClass: organizationalUnit
ou: Apes Incorporated

#people organizational unit
dn: ou=people,dc=apes.process,dc=com
ou: people
objectClass: organizationalUnit

#entry for "Charlton Heston"
dn: cn=Charlton Heston,ou=people,dc=apes.process,dc=com
cn: Charlton Heston
sn: Heston
mail: heston@apes.process.com
telephoneNumber: 508-555-1212
objectclass: inetOrgPerson

#entry for "Roddy McDowall"
dn: cn=Roddy McDowall,ou=people,dc=apes.process,dc=com
cn: Roddy McDowall
sn: McDowall
mail: mcdowall@apes.process.com
telephoneNumber: 508-555-1234
objectclass: inetOrgPerson
```

It's *very* important that there be exactly one blank line between each of the objects in the LDIF file. The line has to be completely blank - no white space characters like tabs or spaces are allowed. If you get error messages that follow the form of "`additional info: objectClass: value #1 provided more than once`", then you have a non-blank line between objects somewhere in the file.

Now, use `ldapmodify` to add the entries to the directory. Make sure OpenLDAP is running before you run `ldapmodify`. Note that you need to supply `ldapmodify` with the root DN and password that you specified in `slapd.conf`, since it needs write access to the directory to add the entries.

```
$ ldapmodify -D "cn=Directory Manager,dc=apes.process,dc=com" -w secret \
   -x -a -f users.ldif
adding new entry "dc=apes.process,dc=com"
```

[5] The LDAP Data Interchange Format (LDIF) is defined in RFC 2849.

```
adding new entry "ou=people,dc=apes.process,dc=com"
adding new entry "cn=Charlton Heston,ou=people,dc=apes.process,dc=com"
adding new entry "cn=Roddy McDowall,ou=people,dc=apes.process,dc=com"
$
```

If you want, you can use the `ldapsearch` command line tool to verify that the entries have been added to the directory.

```
$ ldapsearch -x -b "dc=apes.process,dc=com"
# extended LDIF
#
# LDAPv3
# base <dc=apes.process,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# apes.process.com
dn: dc=apes.process,dc=com
dc: apes.process
objectClass: dcObject
objectClass: organizationalUnit
ou: Apes Incorporated

[...]

# search result
search: 2
result: 0 Success

# numResponses: 5
# numEntries: 4
$
```

About Process Software

Process Software is a premier supplier of communications software solutions to mission critical environments. With over 20 years in business, we were early innovators of email software and anti-spam technology. Process Software has a proven track record of success with thousands of customers, including many Global 2000 and Fortune 1000 companies.

PROCESS™
SOFTWARE

Appendix A - Generating TLS Certificates

Recent versions of OpenSSL include a certificate generation script that makes it very easy to generate your own TLS certificates. The script is named `CA.pl`, and is located in the `/usr/local/misc/` directory. To generate TLS certificates suitable for use with OpenLDAP, perform the following steps:

1. Run `CA.pl` as the root user.

```
$ sudo /usr/local/misc/CA.pl -newcert
Generating a 1024 bit RSA private key
..............++++++
........................++++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase:I am the Walrus.
Verifying - Enter PEM pass phrase:I am the Walrus.
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Massachusetts
Locality Name (eg, city) []:Boston
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Process Software
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:John Doe
Email Address []:jdoe@process.com
Certificate is in newcert.pem, private key is in newkey.pem
$
```

2. The new private key is password protected, which means you have to re-enter the pass phrase every time you start OpenLDAP. That's going to cause problems if you want to have OpenLDAP automatically start when the system is booted (or if you don't want to remember yet another password). Strip out the password using the `openssl` command line tool.

```
$ /usr/local/bin/openssl rsa -in newkey.pem -out newkey1.pem
Enter pass phrase for newkey.pem:I am the Walrus.
writing RSA key
$
```

3. Move the public certificate and private key files to the locations specified by `TLSCertificateFile` and `TLSCertificateKeyFile` in your `slapd.conf` file.

```
$ sudo mv newcert.pem /usr/local/etc/openldap/slapd-cert.pem
$ sudo mv newkey1.pem /usr/local/etc/openldap/slapd-key.pem
```

4. If you desire, you can have your certificate signed by a certificate authority (CA) like Thawte or VeriSign. Visit the website of your CA of choice for more information.