OpenLDAP: Practical application

ORGANISING PRINCIPLE

VOLKER SCHWABEROW

The Lightweight Directory Access Protocol directory service brings structure and order to the chaos of server administration. And with OpenLDAP and Linux, administrators don't even need to incur any licence costs in the process.

The IT world has always been preoccupied with the subject of uniform, centralised user administration, and it is more topical than ever these days, thanks to developments such as Single Sign-On and Public Key Infrastructures. In order to ensure uniform user administration, administrators nowadays use NIS or yellow pages.

Should your requirements be more substantial, however, or if you would perhaps like to include applications in a centralised user data concept, only a scaleable solution will do. This will include facilities for data replication as well as for creating distributed architectures. LDAP (Lightweight Directory Access Protocol) provides the foundation for such a solution in its role as a central network information service.

LDAP is integrated into NDS (Novell Directory Services) and Microsoft's Active Directory. It is an open standard for an information service based on a tree-like database structure. Compared to a normal database, its main advantage is attribute-related storage. LDAP has developed from X.500DAP, but it uses the TCP/IP stack instead of the OSI stack. Its developers have tried to simplify the data structure as compared to X.500, which means, for instance, that data is stored as plain text. This storage method also simplifies the interrogation of LDAP trees, as the client side does not have to deal with any complicated encoding.

LDAP provides a link to X.500, and at the same time minimises the effort for networks and network software (clients). Version 1 of LDAP was created at Michigan University. Only since version 2 has it been possible to use LDAP in the classic client/server model without putting the main burden onto the clients. In the meantime, there is already a white paper for version 3, and its essential principles are beginning to enter LDAP implementations, leading to improvements in the data model.

Data structure

In the LDAP data structure an object class defines a collection of different attributes, which can be used to describe a directory entry. There are predefined object classes that can be used for defining locations, organisations or companies, people or groups.

Object classes can be used to create entries. A typical entry in an LDAP tree looks something like this:

```
cn=Volker Schwaberow, ou=IT, o=MyCompany, c=DE
```

This is what is called a Distinguished Name (DN), or unambiguous name, in an LDAP tree (see figure 1). The entry shows the attributes *cn*, *ou*, *o* and *c*.

You will notice at first glance that LDAP is structured hierarchically, with the DN being read from right to left, similar to the structure of the Domain Name Service on the Internet. As has already been mentioned, attributes are used within a DN. Common attributes in the default LDAP schema are: Common Name (CN), Organisational Unit (OU), State (S) and Country (C).

These attributes are each assigned to an object class through definition. Frequently used object classes are, for example, *Organisation*, *organisationalUnit*, *Person*, *organisationalPerson* and *Country*. The defined object classes determine what an entry can contain. The entry for a person on the LDAP tree can, for instance, contain the telephone number of the person, their General

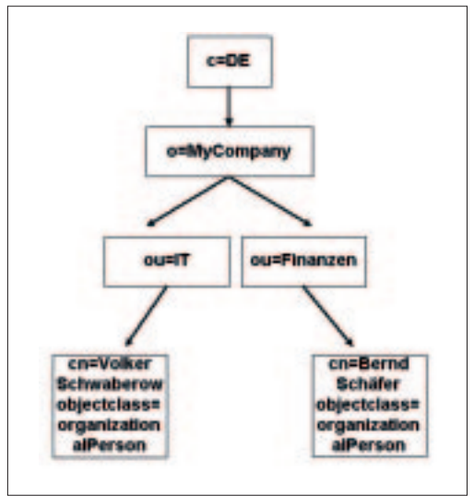

Figure 1: The LDAP tree for the example as a diagram.

Public Key or, if required, a JPEG picture that can be displayed by an LDAP client. The possibilities are virtually endless, only depending on the respective application. If the solution to a requirement doesn't already exist as an attribute or a class it must be implemented in the server's LDAP schema.

User or organisation data can be set up in this schema using text files in LDI format. LDIF (Lightweight Data Interchange Format) has a large variety of different applications. Data entry is one example, another is exporting existing LDAP trees into LDIF files. Due to their LDI format it is easy to maintain LDAP data. A useful overview of LDIF can be found at Netscape. This relates specifically to the Netscape Directory Server, but also contains generally useful information.

Configuration 1: slapd.conf

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world-readable.
#
include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf
schemacheck     on
referral        ldap://myserver.mycompany.de/
pidfile         /var/run/slapd.pid
argsfile        /var/run/slapd.args
#######################################################
# ldbm database definitions
#######################################################
database        ldbm
suffix          "o=MyCompany, c=DE"
rootdn          "cn=Manager, o=MyCompany, c=DE"
rootpw          mypassword
directory       /var/ldap/openldap-ldbm
defaultaccess   none
access to attr="userpassword"
  by self write
  by * compare
access to *
  by self write
  by dn=".+" read
  by * none
access to *
  by dn="^$$" none
  by * read
```

After these LDAP basics, let's get down to business: OpenLDAP. This open source project emerged several years ago from a server project at Michigan University. OpenLDAP consists of a scaleable server with matching LDAP clients and since version 2 it finally supports the protocols in the white paper for LDAP version 3.

Installation of OpenLDAP

After downloading the current source distribution, there are some more requirements to meet before you can compile. One of these is a database compatible with LDAP's own LDBM. LDBM-compatible databases are, for instance, Berkeley DB2 or GDBM (GNU Database Manager). OpenLDAP can also make use of other backends. Since all distributions contain one of the two databases mentioned above, there should be no big problems here.

Now you might as well unpack the sources:

```
tar xvfz openldap-stable-DATUM.tgz
```

Change to the directory where you have unpacked the source and execute

```
./configure -prefix=/usr/local/
```

If that has worked, execute:

```
make depend
make
```

Once OpenLDAP has been compiled, you ought to run the functionality tests. To do this, change to the directory *tests* below the source tree and start *make*.

If the tests (database, server functionality, etc.) are completed successfully, you can install the OpenLDAP servers and clients with the usual

```
make install
```

command. The following is now located under */usr/local*: *slapd* and the replication daemon *slurpd*, along with gateways for X.500 (fax, mail, etc.) are in *libexec*.

The OpenLDAP clients can be found under *bin*. Here, the following three commands are of particular interest:

- *ldapadd* for adding entries to an LDAP directory
- *ldapmodify* for amending entries
- *ldapsearch* for searching the LDAP tree

These commands work locally as well as remotely with access to a remote LDAP server.

Configuration of OpenLDAP

First of all, for simplicity reasons, move the directory */usr/ local/etc/openldap* to */etc/openldap* so that the configuration files are in the right place. Now change to */etc/openldap*, where you will find the following files for the supplied OpenLDAP clients:

- *lldap.conf*, basic client settings
- *lldapfilter.conf*, LDAP search filter configuration
- *lldapsearchprefs.conf*, other object-related filter settings

- *lldaptemplates.conf*, display-related client settings

The following files are for OpenLDAP servers:

- *lslapd.conf*, configuration for the slapd daemon
- *lslapd.at.conf*, predefined attributes
- *lslapd.oc.conf*, predefined object classes

Firstly, open *slapd.conf* and amend it as shown in the Configuration 1: slapd.conf box. You don't need to touch classes and attributes at this point. But before you start working with OpenLDAP, you should at least have a look at the default attributes and classes.

Should you need to change the model, perhaps due to the incorporation of special server software (such as Netscape SuiteSpot products), you can set up additional attributes or classes in the *slapd.conf* file using *include File Name*. Note also that your manager password should of course not be stored as plain text in *slapd.conf* as it is shown in the example in Configuration 1. OpenLDAP accepts SHA, MD5 or CRYPT as password encryption.

The two *include* statements in the *slapd.conf* file are used for loading the standard attributes and classes. The *database* line is important. In it, an LDBM-compatible database is selected as the backend. The keyword *suffix* defines the DN for queries that can run against our server.

The entry *rootdn* should be self-explanatory, this DN has all operating rights and is used by the administrator to bind to LDAP tree operations. The *directory* line determines where *slapd* deposits its database. This directory needs to be created beforehand and must be assigned the file rights *rwx* for the user (*chmod 0700*). The following lines contain the access rights or ACLs (Access Control Lists) for the LDAP tree. The default access is *none*. Each authenticated user is then given *self write* rights to the attribute *userpassword*. Thus each user can change their own password within the LDAP tree. The last ACL permits *anonymous binds*, so that an address book application without a dedicated LDAP tree user can access the server.

After you have implemented the basic settings for the *slapd* server, you should try to start it:

```
/usr/local/libexec/slapd -f /etc/openldap/sl➥
apd.conf
```

Now check whether the server has started correctly, ideally using *ps -ax* |*grep slapd*. By default, the LDAP service runs on port 389. That should also be documented in */etc/services*, of course. Please note that SLAPD can be compiled with TCP wrapper support to create additional security.

Configuration 2: MyCompany.ldif

```
dn: o=MyCompany, c=DE
o: MyCompany
l: Gelsenkirchen
streetaddress: Emscherstr. 41
postalCode: 45891
telephonenumber: 0209-4711
objectclass: organization
dn: cn=Manager, o=MyCompany, c=DE
cn: Manager
sn: Manager
objectclass: person
dn: ou=IT, o=MyCompany, c=DE
ou: IT
objectclass: top
objectclass: organizationalUnit
dn: ou=Finance, o=MyCompany, c=DE
ou: Finance
objectclass: top
objectclass: organizationalUnit
dn: cn=Volker Schwaberow, ou=IT, o=MyCompany, c=DE
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Volker Schwaberow
sn: Schwaberow
telephonenumber: 0209/4712
dn: cn=Bernd Schlaefer, ou=Finance, o=MyCompany, c=DE
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Bernd Schlaefer
sn: Schlaefer
telephonenumber: 0209/4713
```

First directory entries

Once the server is running you can make the first entries in the directory. In order to do this, transfer the contents of the Configuration 2: MyCompany.ldif box into a separate file with the extension *LDIF*. First, the organisation needs to be set up. In our example it is *o=MyCompany, c=DE*. Note that correct formatting of your *LDIF* file is crucial and that it must be created exactly as specified in the example for the entries to work.

For the sake of simplicity, save your initial *LDIF* in the */etc/openldap* directory as well - in case you need it again at a later point. The finished *LDIF* file is then imported with the command line

```
ldapadd -D "cn=Manager, o=MyCompany, dc=DE" ➥
-W < LDIF-Filename
```

After the manager account has been declared, two more organizational units are set up, the IT and Finance departments. Finally, two people are

Configuration 3: MyCompany_modified.ldif

```
dn: cn=Volker Schwaberow, ou=IT, o=MyCompany, c=DE
changetype: modify
replace: telephonenumber
telephonenumber: 0209/4716512
```

assigned to these departments, one is the author, and the other is one Mr. Bernd Schlaefer.

When the entries have been added you can perform your first directory search. This is done using the command *ldapsearch*. Assuming you want to search and list all entries in the directory, the correct command is:

```
ldapsearch -D "cn=Manager, o=MyCompany, c=DE
" -b "o=MyCompany, c=DE" "(objectclass= *)"
```

Note that in large installations this should be used with care. Now search for all occurrences of *cn* beginning with the string *Vol* - that should return an entry. The command for this is:

```
ldapsearch -D "cn=Manager, o=MyCompany, c=DE
" -b "o=MyCompany, c=DE" "(cn=Vol*)"
```

One thing that should be apparent from these two examples is that search attributes require round parentheses.

Of course, the author's telephone number or another value could change. In this case you should create an *LDIF* file along the lines of the example in the *LDIF Modify File* (see Configuration 3: MyCompany_modified.ldif box) which contains the author's data, including the amended telephone number. This LDIF file is imported into the system using the following command:

```
ldapmodify -D"cn=Manager, o=MyCompany, c=DE"
-W < LDIF-File
```

The method shown does not distinguish itself in terms of user friendliness for one entry, but can be well worth it for large-scale changes. Debugging allows you to detect annoying error sources before modification.

Directory interrogation by mail clients

After performing these three local standard operations on the directory, you should try to interrogate it using a mail client. Type the following URL into your browser:

```
ldap://myserver.mycompany.de/o=MyCompany, c
=DE??base
```

Netscape will ask whether you want to add this server to your LDAP setting. Once it has been set up, the LDAP server can be interrogated via the *Address Book*. If addressing has been configured correctly, the entire LDAP directory will be searched for hits whenever a name is entered in the mail client's *To* field. After entering

```
ldap://myserver.mycompany.de/o=MyCompany, c=
DE??sub
```

Listing 1: PHP-Demo

```
<?
        // LDAP Example by V. Schwaberow
        // for Linux Magazine 2001
        echo "<html><head><title>LDAP Test</title></head><body>";
        //Establish TCP connection with LDAP database.
        //This is the IP address of the LDAP server.
        $connection = ldap_connect("192.168.10.248");
        if ($connection){
        // Anonymous bind, sufficient for query according to ACL!
        // However, amendments require a valid DN.
        $result = ldap_bind($connection);
        // Search for all CN attribute entries in LDAP tree.
        $search = ldap_search($connection,"o=MyCompany,c=de","cn=*");
        // Array of search results.
        $entries=ldap_get_entries($connection,$search);
        echo "<table cellspacing=\"0\" cellpadding=\"0\" border=\"2\">";
        // Table loop
        for ($i=0;$i<$entries["count"];$i++){
        $cn=$entries[$i]["cn"][0];
        $dn=$entries[$i]["dn"];
        $phone=$entries[$i]["mail"][0];
        echo "<tr>";
        echo "<td align=\"right\">".$i."</td>";
        echo "<td>".$cn."</td>";
        echo "<td>".$dn."</td>";
        echo "<td>".$phone."</td>";
        echo "</tr>";
        }
        // Close server connection.
        ldap_close($connection);
}
else   {
        echo "Connection to LDAP server cannot be established!";
        }
        echo "</table></body></html>";
?>
```

The author

Volker Schwaberow is the Internet Security Engineer of Sozialwerk St. Georg e.V, a social service provider that is part of the German charity organisation Caritas. He deals with LDAP, PKI and SSO, amongst other things. His first Linux experiences were gained in 1995. Since 1998 he has been working with it on a daily basis and prefers it to any Bluescreen, in private as well as for work.. The author likes reading books, listening to music and programming under C/C++, Java, Perl and PHP.

in the browser, the whole directory is represented as HTML.

Amendments using an editor instead of LDIF files

So what happens if you only want to amend a small entry? You would need a lot of patience to do everything through LDIF files. This is where LDAP editors come in useful. Some recommendations are GQ for Gtk and Kldap. An LDAP browser allows the representation of the LDAP tree as a structural diagram, which simplifies amendments.

Programming languages and LDAP

LDAP's support by various programming languages offers interesting possibilities. It's quick to create an extensive address management system for your home intranet with these, if you use languages such as PHP. Listing 1 shows an LDAP tree listing by anonymous bind using PHP.

That would be enough to create a simple email address management system for a user. The LDAP tree could even be administered via a PHP script; you could, for example, give another department general access to the telephone numbers in the LDAP tree.

Perl also offers many opportunities of integrating LDAP into a script. The CPAN archive contains the Net-LDAPapi interface. However, the Perl module Net-LDAP, which can be found at sourceforge, is much more useful. You can find a short Perl example in Listing 2. There is, of course, also the possibility of programming LDAP-bindings under Java or C/C++; in which case the author's sympathies lie with the Java variant.

Other possibilities and security aspects

In centralised network information services it is possible to bind standards to an authentication for the specified directory. This is done by exchanging the login PAM module. Linux server logins can be verified against a directory. This solution is also valid for Apache, Squid, Qmail and other server services.

The market is overrun with vendors selling such Single Sign-On solutions for a lot of money. However, this feature is also available free, for example in the Plugable Authentication Module from PADL Software. It is open source, and performs better in everyday use than many an expensive SSO solution. OpenLDAP can also be used as a PKI server (Public Key Infrastructures). The Oskar PKI project deserves a special mention in this context.

You should already be familiar with the access control lists (ACLs), which restrict access to attributes according to various criteria. However, at the moment access to the directory service is completely unencrypted. That can be changed using something called an SSL wrapper. By default, LDAP servers use port 389. Most clients also support SSL-encrypted LDAP. A frequently used SSL wrapper is, for example, Sslwrap.

The future

Looking at what the common current network administration methods you will notice that there are too many different services that keep setups and user data locally on servers. Centralising this data would simplify things considerably. That also explains why enterprise products come complete with an LDAP interface.

We can only hope that companies will recognise the potential of Linux, and open source in particular, in this area. The integration of directory services based on open source solutions can minimise costs and improve the stability and reliability of the system at the same time. ■

Listing 2: Net::LDAP

```
#!/usr/bin/perl
#
# Net::LDAP Test for Linux Magazine
#
use Net::LDAP;
# New Net::LDAP Object
      $ldap = Net::LDAP->new(`myserver.mycompany.de');
# anonymous bind
      $ldap->bind;
# The query
      $state = $ldap->search (
                              base => "o=MyCompany, c=DE",
                              filter => "(objectclass=*)"
                             );
# Entry output
      foreach $buffer ($state->all_entries)
      {
            print $buffer->dump;
      }
```

Info

IETF - Internet Engineering Task Force: http://www.ietf.org
RFC1779 - A String Representation of Distinguished Names: ftp://ftp.isi.edu/in-notes/rfc1779.txt
RFC1778 - The String Representation of Standard Attribute Syntaxes: ftp://ftp.isi.edu/in-notes/rfc1778.txt
RFC1777 - Lightweight Directory Access Protocol: ftp://ftp.isi.edu/in-notes/rfc1777.txt
LDAP project of Michigan University: http://www.umich.edu/~dirsvcs/ldap/index.html
OpenLDAP project: http://www.openldap.org
Netscape hints on LDIF format: http://developer.netscape.com/docs/manuals/directory/admin30/ldif.htm#1043950
LDAP browser/editor V2.8.1: http://www.iit.edu/~gawojar/ldap
GQ - LDAP-Browser for Gtk: http://biot.com/gq
Kldap - LDAP browser for KDE: http://www.mountpoint.ch/oliver/kldap
NetLDAPapi - Perl-API for LDAP access: http://search.cpan.org/search
Net::LDAP - Perl-API for LDAP access: http://perl-ldap.sourceforge.net
PADL's LDAP PAM: http://www.padl.com/pam_ldap.html
Oskar PKI: http://oscar.dstc.qut.edu.au
Sslwrap: http://www.rickk.com/sslwrap
Jens Banning: LDAP under Linux, Addison-Wesley, ISBN 3827318130

■