

## Authentication on Linux Using Open LDAP, Part Two

David Del Elson 2001-07-09

### Authentication on Linux Using Open LDAP, Part Two

by David "Del" Elson

last updated July 9, 2001

---

This is the second part of a two-part series devoted to discussing LDAP authentication on Linux. The [first installment](#) offered an overview of LDAP including: installing and configuring OpenLDAP, migrating to OpenLDAP, and setting up LDAP queries. This installment will look at setting up PAM and NSS for LDAP, securing the root account, some LDAP tools, securing OpenLDAP, OpenLDAP and SSL, and some concerns about OpenLDAP. Just a note, while this series focuses on RedHat 7.1, most of the same principles apply to Debian and other Linux distributions.

## Setting up PAM and NSS for LDAP Using authconfig

First, a warning: this only works with authconfig on Red Hat versions 7.0 or later. If you are using Red Hat 6.2 or earlier, you will need to read the man pages regarding the configuration files and edit them all by hand. Note also that there is no `/etc/pam.d/system-auth` file on Red Hat 6.2.

Setting up PAM and NSS for LDAP requires editing a configuration file in `/etc/pam.d` (`/etc/pam.d/system-auth`), modifying the NSS configuration file, which is `/etc/nsswitch.conf`, and editing the `pam_nss` and `pam_ldap` configuration file, which is `/etc/ldap.conf`. Fortunately, Red Hat provides a utility called `authconfig` that will do this automatically for you. To use this, run `authconfig` from the command line. Within `authconfig` this is a fairly straightforward exercise. Mark the box labelled 'Use LDAP'. In the next boxes, you should enter a value for the server (this will be the same IP address you used in `/etc/openldap/ldap.conf`), and a Base DN (this will be the same as the Base DN that you specified in `/etc/openldap/slapd.conf`)

Once you have finished that, everything should be set up. I find that the best way to test this is as follows:

- First, find an account in your LDAP directory, using a command like `ldapsearch -x 'uid=someaccount'`. This account should be one that has been copied from the `/etc/passwd` file that you had on the system before setting up LDAP;
- Check that the account exists, using `'finger someaccount'`. You should get a response showing the user id, name, and other details;
- Using `vi` or another text editor, edit your `/etc/passwd` file and remove the account; and,
- Using `finger` again, check to see if the account still exists. If everything is going correctly, it will - the account details are now being fetched from LDAP instead of from `/etc/passwd`.

## Secure the Root Account

The `slapd.conf` file that was set up earlier contains an entry for the root DN as well as a password. This

password is used as a fallback password for the root DN in case the entry is not found in the directory. Of course, when you first set up LDAP with an empty database, this entry was required. Now it is not, so you should comment it out, as follows:

```
rootdn          "uid=root,ou=People,o=MyCompany,c=AU"
# rootpw       secret
```

Your root password should have been obtained from the `/etc/passwd` or `/etc/shadow` file and inserted into LDAP. To test this, repeat the above search using the `-D` flag of `ldapsearch` to attempt a logon to your LDAP directory, as follows:

```
ldapsearch -x -D 'uid=root,ou=People,o=MyCompany,c=AU' -W 'uid=root'
```

`ldapsearch` will ask you for a password - enter your root password here and, if everything goes according to plan, this should work correctly.

## Command Line LDAP Tools

OpenLDAP comes with a range of Unix command line tools for LDAP directory management. These tools are useful as a reference, and since similar tools are provided with many other commercial LDAP systems, it is a good idea to learn these. The tools come with comprehensive man pages; however, it may be best to read some LDAP information on the Internet and/or an LDAP administration guide in order to learn some of the terminology.

The tools are:

- **ldapsearch** - for searching for entries in LDAP;
- **ldapadd** - for adding entries to LDAP;
- **ldapmodify** - for editing entries in LDAP;
- **ldapdelete** - for deleting entries from LDAP;
- **ldappasswd** - for changing an LDAP entry's password; and,
- **ldapmodrdn** - for renaming LDAP entries.

These tools mostly take a common set of command line parameters, which include options such as:

- **v** - verbose mode;
- **k** - use Kerberos authentication;
- **x** - use simple authentication
- **c** - continuous operation (don't abort on error)
- **f** - read the information from a file;
- **D** - specifies the LDAP DN to bind with;

- **W** - ask for a password for simple authentication; and,
- **H** - use a particular LDAP server, eg: -H ldap://ldap.mycompany.com.

## Other LDAP tools

There is a wide variety of LDAP tools available either for free or commercially on the Internet. A quick search on [freshmeat](#) for LDAP will reveal several. Here are a couple that I quite like.

### GQ

[GQ](#) is a powerful tool that allows you to look inside your LDAP directory and administer any part of it. GQ allows full control over your LDAP server, including:

- **LDAP object browser** - Tools to add/delete/update LDAP entries; and,
- **LDAP schema browser** - Tools to build entries from other entries, or set up templates.

GQ allows you to get inside your LDAP directory and see the internal workings of LDAP. GQ is available in source code from the web site, and the source code includes a .spec file so that you can build an RPM file for it if you prefer.

## Directory Administrator

[Directory Administrator](#) is an application for managing user and group entries in LDAP directory servers. It provides a friendly interface to manage the user's personal details, address book information, and mail routing (for sendmail versions that support mail routing information stored in LDAP.) I find Directory Administrator's interface simple and intuitive, although it is not as powerful as GQ and doesn't go beyond simple user and group management.

The author of Directory Administrator has been promising a new version for some time, and I have a few items on my wish list for it, the main issue being that its main window provides an unsorted and unstructured view of all users in the directory tree. If there are a large number of users in the LDAP directory, this is unsatisfactory (I have around 1500 in one LDAP tree I manage, and it would not be uncommon for some organizations to have tens of thousands). As a basic tool for managing user information on a small Linux network with an LDAP directory, however, Directory Administrator is a great tool.

## Red Hat Kickstart and OpenLDAP

Occasionally, I like to build systems using Red Hat's Kickstart tool. The latest version of the Red Hat installation system includes some Kickstart options that can be used to connect a system to an LDAP-based network as a client during system installation time. The options are described in detail in the [Red Hat Linux Customization Guide](#) but the main one you need is the "auth" parameter which should read:

```
auth --enableldap --enableldapauth --ldapservers= --basedn=
```

Red Hat also includes some information on OpenLDAP in their [Red Hat Linux Reference Guide](#), especially chapter 4. If you are going to use Kerberos authentication with OpenLDAP then you will also need to read chapter 9.

## Making OpenLDAP More Secure

The main way in which you can improve the security of OpenLDAP is to include secure sockets layer (SSL) and transport layer security (TLS) mode in your client/server connection. This encrypts all of the LDAP traffic using the SSL protocol. OpenLDAP version 2.0 and later have the capability to run in SSL and TLS mode using the OpenSSL libraries, although you should really be using a version later than 2.0.7 to get this capability working properly.

The main steps you need to follow in getting OpenLDAP to work with SSL are:

- Make sure that OpenLDAP is compiled with the OpenSSL libraries; and,
- Generate SSL keys for OpenLDAP.
- Configure OpenLDAP to use the SSL keys
- Test!

Note that there are two different modes of operating OpenLDAP with SSL. These are:

- TLS, otherwise known as Start TLS mode. This is the more modern approach to secure communications, as it uses the same TCP port number to connect to the OpenLDAP server (389) but switches to secure communications using a "Start TLS" command before any data is transferred; and,
- SSL mode, which operates on a different TCP port number (636) from the standard LDAP port, and begins the connection in secure mode.

TLS mode is more flexible than SSL mode (since clients or servers that do not understand SSL can continue communicating by ignoring the Start TLS command), but unfortunately many older LDAP servers and clients do not implement this mode and only use SSL mode. Therefore it is preferable, in my opinion, to get your LDAP servers and clients working in both TLS and SSL mode.

### Compiling OpenLDAP with OpenSSL

Hopefully, this is the easiest part of the job. If you are using a packaged installation of OpenLDAP provided with, for example, Red Hat or Debian Linux, you will find that this has already been done for you. Once again, make sure that you have a recent version of OpenLDAP (from Red Hat, any version that is 2.0.7-3 or later should be fine.)

If you are compiling OpenLDAP from source, you will simply need to give the following additional flag when running the OpenLDAP configure script:

```
--with-tls
```

You also need to make sure that the OpenSSL libraries are present on your system.

## Generating SSL keys for OpenLDAP

To use OpenLDAP in TLS or SSL mode you will need to generate a PEM format SSL key. Assuming that you have the OpenSSL package installed (otherwise you would not be doing this,) you will find a Makefile in the `/usr/share/ssl/certs` directory that contains the appropriate commands to create this key. You can run these with the following simple command:

```
cd /usr/share/ssl/certs
make slapd.pem
```

During the key-building process, you will be asked for a whole raft of details about your server. This includes the country code, state, organization name, e-mail address, server name, etc. Answer these questions as fully as possible - they are encoded in the PEM file. If you do not have OpenSSL's Makefile, or you cannot run the "make" program for some reason, then you will need to build the PEM file manually. Here is a rough outline of the commands needed to do this:

```
/usr/bin/openssl req -newkey rsa:1024 -keyout tempfile1 -nodes -x509 -days 365 -out tempfile2
cat tempfile1 > ldap.pem
echo "" >> ldap.pem
cat tempfile2 >> ldap.pem
rm -f tempfile1 tempfile2
```

Note that this command creates an RSA key and then self-signs that key. Self-signed keys are not usual in (for example) HTTPS communications, as the keys are generally signed by a third party known as a Certification Authority. In this case, however, a self-signed key is perfectly okay, as most LDAP clients do not check the signature of the key. Of course, for more advanced users, it would be entirely possible to generate a key and have it signed by an external Certification Authority. It's up to you.

**Important note:** once you have generated the keys, it is important that they are made 'read only' by the user that is running the LDAP server. For Red Hat Linux, OpenLDAP's server (slapd) runs as an unprivileged user called 'ldap' in a group called 'ldap'. To make this user and group the owner of the key that you have just generated, run the following command:

```
chown ldap.ldap slapd.pem
```

## Configuring OpenLDAP to Use SSL Keys

To configure OpenLDAP to use the SSL key you have just generated, you need to modify the `/etc/openldap/slapd.conf` file. The following lines will need to be added to the file, (they can be anywhere in the file, but I suggest putting them in close to the top, after all of the 'include' statements):

```
TLSCipherSuite HIGH:MEDIUM:+SSLv2
TLSCertificateFile /usr/share/ssl/certs/slapd.pem
TLSCertificateKeyFile /usr/share/ssl/certs/slapd.pem
```

Note that there are different allowable values for the `TLSCipherSuite` line, but the above line is the one that I recommend.

You may also need to modify your LDAP startup script. If you are using Red Hat Linux version 7.0 or later then you won't need to do this. Otherwise, you should locate the line in your startup script (probably `/etc/rc.d/init.d/ldap` or `/etc/init.d/ldap`) that contains the line to start `slapd`, and modify it to look like this:

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

It is the options after `"-h"` that we are most interested in at this point. You will notice that we are telling the LDAP server to work in both `ldap` and `ldaps` (secure) mode. Theoretically, once this is working, you could turn `ldap` mode off and only use the secure mode, but this is not recommended, as many LDAP clients don't support this secure mode. Note that there may be other options on the line that runs `slapd` in your startup script: you should leave those intact.

You will now need to restart your LDAP server in order for it to re-read the `slapd.conf` file:

```
/etc/rc.d/init.d/ldap stop
/etc/rc.d/init.d/ldap start
```

## Testing OpenLDAP and SSL

The first step in testing that your LDAP server is listening in SSL mode is to run the following command:

```
netstat -a | grep LISTEN
```

Looking through the output of that command, you should see that your `slapd` server is listening on port 389 (`ldap`) as well as port 636 (`ldaps`). If SSL mode is not working, then it will be listening on port 389 (`ldap`) only.

Fortunately, OpenSSL comes with not only a handy key generator, but also an SSL line tester. It works by running the following command:

```
openssl s_client -connect localhost:636 -showcerts
```

Note that I have elected to use the direct connection to LDAP in SSL mode rather than going through TLS - the latter is possible, but somewhat trickier.

If all goes to plan, you should see some output from your LDAP server in the command window, showing information about the certificate, as well as the certificate, session IDs, and master keys, that are in use by the server.

You could now find an LDAP client that supports SSL or TLS mode, such as GQ, turn on the option in the client that enables TLS mode, and see if it works. Voila, you now have a secure LDAP server!

## Issues with OpenLDAP

As much as I like free software, I have found a number of issues in the time that I have been using OpenLDAP. For instance, in comparison to some LDAP servers such as iPlanet DS and Novell's NDS, it is quite slow. This is particularly noticeable when both read and write operations are running at the same time.

I have also discovered a number of bugs in the OpenLDAP servers and libraries. These cause several issues, including things like having NSS lookups fail on occasion under load, having SSL transactions terminate unexpectedly, and having directory recovery problems after a system crash. To their credit, the OpenLDAP team maintains an active and publicly accessible [bug tracking system](#) where the status of such issues can be tracked and monitored. On occasion I have managed to repair certain bugs by reverting to previous versions of the NSS LDAP libraries.

As alternatives to OpenLDAP, both Novell and iPlanet have (non-free, non-open source) directory servers available on Linux. Both of these work fine with the standard LDAP NSS and PAM libraries, although Novell provide their own PAM and NSS libraries (for a price). Unfortunately, an indepth discussion of Novell's and iPlanet's directory service offerings will have to wait for another time, as it is outside the scope of this article.

### Relevant Links

[Authentication on Linux Using Open LDAP, Part One](#)

*David "Del" Elson, SecurityFocus*

[Privacy Statement](#)

Copyright 2006, SecurityFocus