Configuring OpenLDAP for Authentication

Joshua Blanton

February 14, 2003

1 What is LDAP?

LDAP is the Lightweight Directory Access Protocol, which is nothing more than a standard for querying a simple database. LDAP is commonly used for storing user information and authentication information, however, and this aspect of its use is what this paper will attempt to discuss.

2 Why use LDAP?

There are a few reasons to use LDAP: a network with multiple systems wishing to share login information might use LDAP to synchronize logins and passwords; someone wishing to export a publicly available directory of users/people/things might use LDAP as the server; of course, someone might just want to play with LDAP because it's interesting. Most systems have no need for a system like LDAP, and in fact will only notice drawbacks (slower authentication, another point of failure, etc.) — LDAP is largely useful for networked computers, although it can be run stand-alone. This paper will discuss setting up an LDAP server to serve login information for a network of computers.

3 Setting up OpenLDAP

First things first — OpenLDAP must be installed on the server. Install both the server and client software, if your distribution splits the two. All of the paths and setup scripts in this example document are correct for Red Hat 7.3, and may need to be adjusted on different systems.

Once the server and client software has been installed, edit the `/etc/openldap/slapd.conf` file; we will begin configuration now. Find the `suffix` line and change the value from the default to `"dc=DOMAIN,dc=TLD"` (where `DOMAIN` is the domain name of your system, or anything unique, and `TLD` is the top-level domain of your system, or something else unique; it should be mentioned that if you have a three-level domain name, only use the bottom two…The third level will be used everywhere after this point, but should not be entered here). Change the `rootdn` value to reflect the "root" user that you would like to use — this user does not have to exist on your current machine (for example, `rootdn "cn=ldaproot,dc=my,dc=domain,dc=org"` will be the user `ldaproot` at the "domain" `my.domain.org` (note the use of the three-level domain name).

Create the `rootpw` value using everyone's favorite bloated scripting language, perl, in the following manner:

```
   perl
print crypt ('MYPASSWORD','SL');
D̂
```

where `MYPASSWORD` is the password you would like, and `SL` is any random two-letter string (called a salt). This creates a password in the old unix `crypt()` format, which according to OpenLDAP's documentation is the strongest encryption supported. Once you type in your perl "script," you should see a line of garbage printed out immediately before your shell prompt; copy that line of garbage into your `rootpw` value and your password is set.

I should take a moment to mention that I don't think that `crypt` passwords are the only passwords supported by OpenLDAP; however, all of the documentation I saw suggested that I use one. I think that SSHA passwords and MD5 passwords also work, and these passwords can be generated with the `slappasswd` utility — I would certainly try it at some point, if my system weren't already authenticating from my LDAP server.

4 Migrating Current Data to LDAP

If your system already has usernames and passwords that you would like to migrate into the database, this is the time to create the initial entries; a company called PADL created some fairly extensive migration tools for NIS, regular linux passwords, and a few other authentication types. In the directory `/usr/share/openldap/migration` there are several perl and shell scripts that will generate a database and enter it into your LDAP database, but first some initialization is in order.

Edit the file `/usr/share/openldap/migration/migrate_common.ph` and change its variables to reflect those of your local network; in particular, change the `DEFAULT_MAIL_DOMAIN`, `DEFAULT_MAIL_HOST`, and `DEFAULT_BASE` (which is of the format `"dc=my,dc=domain,dc=org`. Once these variables have been set, run `./migrate_all_offline.sh` (if you use standard linux authentication, like `/etc/passwd` and `/etc/shadow`). In a few seconds you should have a fully-populated database — a problem occurred when I ran this script, however, as it died trying to enter a protocol called `tp++` from `/etc/protocols`; I solved this by commenting out the protocol. Another thing to check — when I used this script, the files it created in `/var/lib/ldap` were owned by `root`, so I had to `chown ldap.ldap /var/lib/ldap/*` before the server could read its own database. Hopefully this is a bug in my particular distribution of the PADL scripts, but I suspect it is not.

5 Access Controls

LDAP defaults to allowing everyone to read all data on the server; this is sub-optimal, given that the server contains all password information on the system at this point (although the passwords are encrypted, many users use very poor passwords that could be brute-forced in a trivial amount of time). There may be default access controls already in the LDAP configuration file (`/etc/openldap/slapd.conf`), but those rules did not seem to be sufficient for my wants. I ended up with an authentication setup like this:

```
access to dn="(.*,)ou=People,dc=my,dc=domain,dc=org" attrs=cn,uid,gecos,\
mail,entry
```

```
    by self write
    by dn="cn=ldaproot,dc=my,dc=domain,dc=org" write
    by * read
access to dn="(.*,)ou=People,dc=my,dc=domain,dc=org" attr=userPassword
    by self write
    by dn="uid=ldapauth,ou=People,dc=my,dc=domain,dc=org" read
    by dn="cn=root,dc=my,dc=domain,dc=org" write
    by * auth
access to *
    by self write
    by anonymous auth
    by dn="uid=systemqueries,ou=People,dc=my,dc=domain,dc=org" read
    by dn="uid=ldapauth,ou=People,dc=my,dc=domain,dc=org" read
    by dn="cn=root,dc=my,dc=domain,dc=org" write
    by * read
```

These lines are fairly easy to understand, if you enter the LDAP mindset; the first line says that the rule is granting access to the values inside `ou=People,dc=my,dc=domain,dc=org` (this is a section of the database pertaining to the users of the system) that have the attributes `cn, uid, gecos, mail, entry, uidNumber, gidNumber, homeDirectory, objectClass`. The next few lines specify who gets said access, and what level of access is being granted; for example, the owner of the entry can write to the entries, everyone can read the entries, and the user `ldaproot` can write to the entries. LDAP permissions are sort of a pyramid, with the lowest permissions being `none`, which means no access; above that is `auth`, which means that a user can use this value to authenticate (log in) to the server; after that comes `search`, which doesn't give read access, but the sub-entries below the entry being modified can be searched; `read` gives read access to the values of attributes; then finally `write` gives write access to an attribute. These attributes compound; if a user has `search` access, he/she also has `auth` access — similarly, if a user has `write` access, he/she also has `auth`, `search`, and `read` access. It should also be mentioned that the *first* matching rule will be followed — so if there is a `by *` rule before anything else, that rule will be the only one followed.

The permissions that I am using are not really what I would like to use, but they are required for the authentication scheme I am using at the moment. For some reason, the linux LDAP authentication stuff uses anonymous requests to look up UID and GID fields (like when using `ls -l`), and so all of that information must be exported — with the exception of the `gecos` (real name) and `mail` (email address) fields, which I added so that email clients can auto-complete addresses using the server.

`Starting the Server` At this point, the server can be started (under Red Hat, run `/sbin/service ldap start`. Your distro may do this differently — I'm sorry. Read the manual for your distro). In fact, it can be queried as well; feel free to test out the server with some `ldapsearch` queries, to make sure it works... First, edit the file `/etc/openldap/ldap.conf` and add the `HOST` and `BASE` entries for your machine (the hostname and root DN, of course). Then try a query like:

```
ldapsearch -x "uid=USERNAME"
```

where `USERNAME` is a user in the database. The server should respond with the login information of said user; if not, check the logs. … which brings me to an important section!

6 Logging

LDAP defaults to logging to the `syslog LOCAL4` section, for which my computer had no configuration. I used a line like the following, placed in `/etc/syslog.conf`, to fix this:

```
   local4.* /var/log/ldap.log
```

which sends all LDAP messages to `/var/log/ldap.log` where they will be stored for my reading pleasure.

7 SSL Tunnels

No one wants to authenticate over the network in plain text, so OpenLDAP supports SSL. To create a key (using OpenLDAP v2.0.x — according to the documentation, OpenLDAP 2.1.x comes with scripts to do this more gracefully; see `http://www.openldap.org/faq/data/cache/185.html`), use the `Makefile` from OpenSSL in `/usr/share/ssl/certs`:

```
   cd /usr/share/ssl/certs
make slapd.pem
chown ldap.ldap slapd.pem
chmod 400 slapd.pem
```

This will create a key that OpenLDAP can use for SSL encryption; when running `make slapd.pem` you will be prompted for a few answers — **make sure** that your `Common Name` for the certificate is the same as the hostname of the server, or else the authentication will fail (also make sure that you always refer to the server with that name, and not as `localhost` or any other hostname or else SSL will fail). After the SSL key has been created, append the line `ssl on` to the `/etc/openldap/ldap.conf` file, add values `TLSCertificateFile` and `TLSCertificateKeyFile` in `/etc/openldap/slapd.conf` pointing to your key (probably `/usr/share/ssl/certs/slapd.pem` if you followed the above example), and restart the server. SSL is now enabled.

8 Authenticating

This is where you can break your system. Perhaps I should say that a little louder: **This is where you can break your system**. If you don't feel comfortable fooling around with authentication on your computer/server (or don't know how to use single-user-mode to undo whatever damage you may do breaking authentication), you probably should not continue. Do so at your own risk — I didn't tell you to do this.

Edit `/etc/nsswitch.conf` to include entries for ldap (in my system, I simply relpaced `nis` with `ldap` in all cases); if you use the default order, your system will fall back on the LDAP server only after a failure with the standard flat-file text files; this allows local users not on the network system. Edit the file `/etc/ldap.conf` to have the correct `host`, `base`, and `rootbinddn` values. The `rootbinddn` value is a little bit scary, as this is the user that your machine will connect to the LDAP server as when running as root; this means that, if anyone roots any machine on the network, they have all of the access rights of the `rootbinddn` user on the LDAP server. This is also scary because the password for the `rootbinddn` user is in plaintext, in the file `/etc/ldap.secret` (which should be made read-only by `root` only, of course). For these reasons, I chose to create a user called `ldapauth` on the LDAP server that can read all files (the authentication setup example above reflects this) but not write any — this is less dangerous, I feel, than having write access to the database. This also means, however, that a local root user cannot change all users' passwords; however, specific users can be given the right to write to all database entries, negating the need for root to do this. The creation of the `ldapauth` user will be discussed momentarily.

After these files have been edited, run `authconfig`, choose the LDAP option, and fill in the appropriate values for server hostname and base. This will edit your `PAM` configuration files appropriately, and then the system will attempt to look up users that are not local from the LDAP database.

`Adding Users` Adding users is no longer as trivial as it once was; now a root user must use `ldapadd` to create a user. First a text file must be created, containing the following lines:

```
dn: uid=ldapauth,ou=People,dc=my,dc=example,dc=org
uid: ldapauth
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
gecos: LDAP Authentication User
cn: ldapauth
uidNumber: 400
mail: foo@hasnomailaddress.my.example.org
gidNumber: 400
homeDirectory: /root
```

...and another text file created, containing the following lines (if you wish to have different groups for each user):

```
dn: cn=ldapauth,ou=Group,dc=my,dc=example,dc=org
objectClass: posixGroup
objectClass: top
cn: ldapauth
gidNumber: 400
```

After these two files are created, run:

```
ldapadd -x -D"cn=ldaproot,dc=my,dc=domain,dc=org" -W -f FILENAME
```

for each file. This adds a user with the login ID of `ldapauth`, a UID of 400, email address `foo@hasnomailaddress.my.example.org`, and home directory `/root`. To set the user's password, run:

```
ldappasswd -x -D"cn=ldaproot,dc=my,dc=domain,dc=org" -W
"uid=ldapauth,ou=People,dc=my,dc=example,dc=org"
```

where the final string can be changed to match the user whose password is being changed. This is not the preferred way to change a regular user's password, but since this user will only ever be accessing the database (and never logging in to an actual computer) there is no problem.

9 Adding real users

The above example gives a basic idea of how user addition works... However, real users have much more information in their accounts. To add a user, fill out the following template:

```
dn: uid=testuser,ou=People,dc=my,dc=domain,dc=org
uid: testuser
objectClass: posixAccount
objectClass: top
objectClass: account
objectClass: shadowAccount
cn: Joe K. Testuser
gecos: Joe K. Testuser
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
mail: testuser@my.domain.org
uidNumber: 1400
gidNumber: 1400
homeDirectory: /home/testuser
```

...and if you intend to use separate groups for each user, this as well:

```
dn: cn=testuser,ou=Group,dc=my,dc=example,dc=org
objectClass: posixGroup
objectClass: top
cn: testuser
gidNumber: 1400
```

Then these two files' entries can be added to the database using the same command that was used to add the `ldapauth` user, only replacing the filenames with the ones recently created. I intend to, at some point near in the future, automate this process with a script — however, I have not as yet. The user password, by the way, can be generated using the `slappasswd` command, and can be added to the user's entry by creating a file with the following lines:

```
dn: cn=testuser,ou=Group,dc=my,dc=example,dc=org
changetype: modify
add: userPassword
userPassword: OUTPUT FROM SLAPPASSWD
```

This is entered into the database using `ldapmodify`, which takes the same arguments as `ldapadd`. A user can change his/her password in this same method, and so I also intend to write a script that changes passwords in this manner…

10 Debian

Face it - Debian sucks. Apparently Debian `-stable` doesn't feel the need to ship with an ssl-enabled LDAP toolset, so `-unstable` must be used; on top of that, it doesn't support the `start_tls` ssl option, and so the `ldaps` settings must be used. Other than that, the files are in different places (`/etc/libnss_ldap.conf` and `/etc/pam_ldap.conf`) but set up the same.

11 Fighting it Until it Works

I will be totally honest — this is a non-trivial thing to set up. The LDAP server runs out-of-the-box, but the linux authentication tools leave something to be desired in the documentation and usability departments. I have found a few websites that actually give information on authentication, but for the most part I check the logs of the LDAP server and try and fix permissions. I'm confident that there are settings that I'm missing that make authentication a million times better, but I don't know what they are yet — when I find them I'll update this document.

Useful places to find information:

```
http://www.mandrakesecure.net/en/docs/ldap-auth.php
http://www.padl.com/Contents/Documentation.html
http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/ref-guide/s1-ldap-redhattip
http://www.openldap.org/faq/data/cache/2.html
```