LINUX JOURNAL

LDAP for Security, Part I

By Mick Bauer
Created 2003-07-01 01:00

OpenLDAP offers the convenience of a common directory across all applications. And if you set it up right, it will make your network more secure, not less.

Suppose you have an Internet Mail Access Protocol (IMAP) server and a bunch of users, but you don't want to give each user a shell account on the server. You'd rather use some sort of central user-authentication service that can be used for other tasks as well. While you're at it, you also need an on-line address book for your organization's e-mail and groupware applications. And suppose, in addition to all that, you also need to provide your users with encryption tools that use X.509 certificates and then manage digital certificates for your entire organization.

Would you believe that one service can address all four scenarios? The Lightweight Directory Access Protocol (LDAP) does all of this and more. And wouldn't you know it, the Open Source community is blessed with a free, stable and fully functional LDAP server and client package that is already part of most Linux distributions: OpenLDAP.

The only catch is LDAP is a complicated beast. To make sense of it, you're going to have to add still more acronyms and some heavy-duty abstractions to your bag of UNIX tricks. But armed with the next few months' Paranoid Penguin columns and a little determination, you'll have the mighty LDAP burro pulling several large plows simultaneously, making your network both more secure and easier to use. In my experience, ``more secure'' and ``simpler for end users'' rarely go hand in hand, so I'm excited finally to be covering OpenLDAP in this column.

LDAP Basics

In a nutshell, LDAP provides directory services, a centralized database of essential information about the people, groups and other entities that comprise an organization. As every organization's structure and its precise definition of essential information may be different, a directory service must be highly flexible and customizable. It's therefore an inherently complex undertaking.

The X.500 protocol for directory services is a case in point. It was designed to provide large-scale directory services for large and complex organizations.

Accordingly, X.500 is itself a large and complex protocol, so much so that a lightweight version of it was created: the Lightweight Directory Access Protocol. LDAP, described in RFC 1777, is essentially a subset of the X.500 protocol, and it's been implemented far more widely than X.500 itself has been.

X.500 and LDAP are open protocols, like TCP/IP; neither is a standalone product. A protocol has to be implemented in some sort of software, such as a kernel module, a server dæmon or a client program. Also like TCP/IP, not all implementations of LDAP are alike or even completely interoperable (without modification). The particular LDAP implementation we cover here is OpenLDAP, but you should be aware that other software products provide alternative implementations. These include Netscape Directory Server, Sun ONE Directory Server and even, in a limited way, Microsoft Active Directory in Windows 2000 Server.

Luckily, LDAP is designed to be extensible. Creating an LDAP database on one platform that is compatible with other LDAP implementations is usually a simple matter of adjusting the database's record formats, or schema, which we'll discuss next month. Therefore, it's no problem to run an OpenLDAP server on a Linux system that can provide address book functionality to users running, say, Netscape Communicator on Macs.

Getting and Installing OpenLDAP

Being such a useful and important tool, OpenLDAP is included in most major Linux distributions. Generally, it's split across multiple packages: server dæmons in one package, client programs in another, development libraries in still another. This article is about building an LDAP server, so naturally you'll want to install your distribution's OpenLDAP server package, plus OpenLDAP runtime libraries if they aren't included in the server package.

You might be tempted to forego installing the OpenLDAP client commands on your server if no local user accounts will be on it and you expect all LDAP transactions to occur over the network. However, these client commands are useful for testing and troubleshooting, so I strongly recommend you install them.

The specific packages comprising OpenLDAP in Red Hat are openldap (OpenLDAP libraries, configuration files and documentation); openldap-clients (OpenLDAP client software/commands); openldap-servers (OpenLDAP server programs); and openldap-devel (headers and libraries for developers). Although these packages have a number of fairly mundane dependencies, including glibc, two are required packages you may not have installed already: cyrus-sasl and cyrus-sasl-md5, which help broker authentication transactions with OpenLDAP.

In SuSE, OpenLDAP is provided in the following RPMs: openldap2-client (in section n1 of SuSE versions 7.3 and 8.0); openldap2 (includes both the OpenLDAP libraries and server dæmons and is found in section n2); and openldap2-devel (found in section n2 for SuSE 7.3 and n4 for SuSE 8.0). As with Red Hat, be sure to install the package cyrus-sasl, located in SuSE's sec1 directory.

In both the 7.3 and 8.0 distributions, SuSE provides packages for OpenLDAP versions 1.2 and 2.0. Be sure to install the newer 2.0 packages listed in the previous paragraph, unless you have a specific reason to run OpenLDAP 1.2. This guideline does not apply to Red Hat or Debian, both of which are standardized on OpenLDAP 2.0 in their current distributions.

For Debian 3.0 (Woody), the equivalent deb packages are: libldap2 (OpenLDAP libraries, in Debian's libs directory); slapd (the OpenLDAP server package, found in the net directory); and ldap-utils (OpenLDAP client commands, also found in the net directory). You'll also need libsasl7 from the Debian libs directory.

If your distribution of choice doesn't have binary packages for OpenLDAP, or if a specific feature of the latest version of OpenLDAP is lacking in your distribution's OpenLDAP packages or if you need to customize OpenLDAP at the binary level, you always can compile it yourself from source you've downloaded from the official OpenLDAP web site at www.openldap.org [1].

Configuring and Starting slapd

The main server dæmon in OpenLDAP is called slapd, and configuring this program is the first step in getting OpenLDAP working once it's installed. Its configuration is determined primarily by the file /etc/openldap/slapd.conf. The ``OpenLDAP 2.0 Administrator's Guide'' at www.openldap.org/doc/admin20/guide.html [2] has an excellent quick-start procedure for getting slapd up and running: it's in Section 2, starting at Step 8. That document also explains directory services and LDAP concepts in more depth than this article does, using tree/hierarchy diagrams.

Let's walk through this procedure to make sure you get off to a good start. The first thing to do is edit slapd.conf, an example of which is shown in Listing 1. As you can see, slapd.conf is a typical Linux configuration file: each line consists of a parameter name followed by a value.

Listing 1. Customized Part of /etc/openldap/slapd.conf [3]

The first parameter shown in Listing 1, database, specifies what type of database back end to use. Usually, the best choice here is ldbm, which uses the fast dbm database format, but shell (for custom shell-script back ends) and passwd (to use /etc/passwd as the back end) also are valid choices. There may be multiple database definitions, each with its own set of applicable parameters; all the lines in Listing 1 comprise a single database definition.

The next parameter in Listing 1 is suffix, which determines what queries match this database definition. Here, the specified suffix is wiremonkeys.org, expressed in LDAP-speak as a series of domain component (dc) statements, which are parsed from left to right. In other words, if an LDAP client queries our example server for information about the distinguished name (dn) cn=bubba,dc=wiremonkeys,dc=org, our server matches that query against this database definition, as the dn ends with dc=wiremonkeys,dc=org. See the Sidebar ``A Crash Course in X.500 Naming'' for

more information about distinguished names.

Sidebar: A Crash Course in X.500 Naming [4]

The next two entries in Listing 1 have to do with LDAP database administration; rootdn and rootpw specify the user name and password, respectively, that must be supplied by remote or local commands that perform administrative actions on the LDAP database. Interestingly, this entry is used *only* for this purpose. It doesn't show up in regular LDAP database queries.

This addresses the paradox of how to authenticate the actions required to populate the authentication (LDAP) database. Later, after you've populated your LDAP database with real entity records, designate one of them as the administrative account, using slapd.conf access control lists (acls), and delete the rootdn and rootpw entries. I'll cover that step in a future column; for now, rootdn and rootpw suffice.

It's a very, very bad idea to store the value of rootpw as clear text. Instead, you should use the **slappasswd** command to generate a password hash, shown in Listing 2.

Listing 2. The slappasswd Command [5]

As you can see, slappasswd prompts you for a password and prints that password hashed with the algorithm you specify with the -h option. Be sure to enclose this value in curly brackets--see the slappasswd(8C) man page for a list of valid choices. You can copy and paste slappasswd's output directly into slapd.conf, which is precisely what I did to create the rootpw value in Listing 1.

Getting back to Listing 1, the next parameter in this directory definition is directory. Obviously enough, this specifies in which directory on the local filesystem your LDAP directory should be created. Because /var is the customary place for growing files such as logs and databases, Listing 1 shows a value of /var/lib/ldap. This directory must already exist, and make sure it's owned by OpenLDAP's user and group, usually ldap and ldap. Its permissions should be set to 0700 (**-rwx------**).

Technically, that's enough to get started: you can try starting slapd with your ldap startup script, most likely /etc/init.d/ldap, though this may vary among distributions. I encourage you to start adding practice entries to your LDAP database using the **ldapadd** command--the quick-start procedure I mentioned earlier shows how.

Before you begin managing and querying your LDAP database over the network, however, you'll want to configure and enable TLS encryption. This is important, as the simple authentication method used by OpenLDAP sends authentication credentials over the network unencrypted. But I'm out of space for now, so we'll cover that next month. If you can't wait until then, Vincent Danen explains how in his on-line article ``Using OpenLDAP for Authentication'', at www.mandrakesecure.net/en/docs/ldap-auth.php [6], though it is somewhat Mandrake-centric. I'll also discuss some considerations in determining the structure of your LDAP database and show how to build one. Until then, good luck!

> **Mick Bauer**, CISSP, is Linux Journal's security editor and an IS security consultant for Upstream Solutions LLC in Minneapolis, Minnesota. Mick spends his copious free time chasing little kids (strictly his own) and playing music, sometimes simultaneously. Mick is author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

email: mick@visi.com [7]

Links

[1] http://www.openldap.org
[2] http://www.linuxjournal.com/
[3] http://www.linuxjournal.com//articles/lj/0111/6789/6789l1.html
[4] http://www.linuxjournal.com//articles/lj/0111/6789/6789s1.html
[5] http://www.linuxjournal.com//articles/lj/0111/6789/6789l2.html
[6] http://www.mandrakesecure.net/en/docs/ldap-auth.php
[7] http://www.linuxjournal.com/mailto:mick@visi.com