LINUX JOURNAL

Paranoid Penguin - Authenticate with LDAP

By Mick Bauer
Created 2003-08-01 01:00

The directory server is running, so now it's time to configure crypto and add some users.

Last month, we did some of the preliminary work in setting up an OpenLDAP server. We installed the base, server and, where applicable, client packages for OpenLDAP, and we entered some basic configuration information into the file /etc/openldap/slapd.conf (slapd is OpenLDAP's server dæmon).

This month, we configure TLS encryption, start the dæmon and begin building an LDAP database. We won't have a finished authentication server yet, but we'll be pretty close. Next month, in this series' third and final installment, we'll get there.

TLS for Secure LDAP Transactions

By default, OpenLDAP transactions over networks are conducted in clear text. If you're using OpenLDAP, for example, as a centralized address book server on a trusted network, that's probably fine. But if you're using it to authenticate users, regardless of whether the networks involved are trusted, you really ought to encrypt your LDAP communications to protect your users' passwords from eavesdroppers.

The LDAP v.3 protocol, support for which was introduced in OpenLDAP 2.0, provides encryption in the form of Transport Layer Security (TLS), the same mechanism used by web browsers and mail transport agents. TLS is the successor to SSL, the Secure Sockets Layer. All you need to do to take advantage of this is create a server certificate on your LDAP server; add a couple more lines to /etc/openldap/slapd.conf, and optionally, tweak slapd's startup options.

To generate a server certificate, you need OpenSSL. This already should be present on your system, because binary OpenLDAP packages depend on OpenSSL.

What sort of certificate you should use as your LDAP certificate actually is a fairly subtle question. Will the server need a certificate that has been signed by some other certificate authority (CA), such as Thawte or VeriSign? That is, will your LDAP clients need to see an externally verifiable certificate when connecting to your server? Or, will your organization be its own CA? If so, will the LDAP server also act

as your local CA, issuing and signing both its own and other hosts' and users' certificates?

If your needs match any of those scenarios, you'll need to do a bit more work than I'm able to describe here. Suffice it to say that the certificate slapd uses can't have a password associated with it-its key can't be DES-encrypted-so a self-signed certificate, though technically a CA certificate, shouldn't be used as an actual CA certificate for signing other certificates. If you want to use your LDAP server as a real CA, you'll need to create two keys, a password-protected CA key and a password-free slapd key. Vincent Danen's article "Using OpenLDAP for Authentication" (www.mandrakesecure.net/en/docs/ldap-auth.php [1]) discusses this.

For many if not most readers, it will be enough to create a self-generated TLS-only certificate to be used by slapd and slapd alone. If you don't care about being a CA and you don't need your LDAP clients to be able to verify the server certificate's authenticity via some third party, you can create your certificate like this:

```
bash-$> openssl req -new -x509 -nodes \
-out slapdcert.pem -keyout slapdkey.pem \
-days 365
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
....++++++
.........++++++
writing new private key to 'slapdkey.pem'
```

On this command line I told OpenSSL to generate a new X.509 certificate, without password protection, with the certificate (public key) stored in the current working directory in the file slapdcert.pem, and the private key stored in the file slapdkey.pem, with a lifetime of 365 days,

After issuing this command, you will be prompted for distinguished name information for the new certificate and key. For OpenLDAP's purposes, the most important field here is the common name. This must be set to your LDAP server's DNS name, which is the name your LDAP clients will see associated with this certificate. If your LDAP server's IP address, for example, reverse resolves to bonzo.lamemoviesfromthepast.com but its server certificate shows a CN of bonzo.lm.com, LDAP clients will reject the certificate and therefore will be unable to negotiate TLS connections (with unpredictable results, depending on your client software).

Once you've got certificate and key files, copy them into /etc/openldap if you weren't already in that directory when you created them. Make sure that both of these are owned by ldap or whatever user your Linux distribution runs slapd as, Red Hat and SuSE use ldap, and that your key file has very strict permissions, such as `-r--------`. Your certificate file may, however, be world-readable, because this contains a public

key.

It's possible to specify the same filename after both the -out and -keyout options, resulting in both the certificate and private key being stored in a single file. This is fine if you don't intend to share the certificate. Keeping the two separate, however, allows you to distribute the server certificate while still keeping the server (private) key secret.

Naturally, it isn't enough to have certificate/key files in place; you need to tell slapd to use them. As with most slapd configuration, this happens in /etc/openldap/slapd.conf. Listing 1 shows the sample slapd.conf entries from last month's column (in case you've forgotten what we covered), plus three additional ones: TLSCipherSuite, TLSCertificateFile and TLSCertificateKeyFile.

Listing 1. Customized Part of /etc/openldap/slapd.conf

```
database        ldbm
suffix          "dc=wiremonkeys,dc=org"
rootdn          "cn=ldapguy,dc=wiremonkeys,dc=org"
rootpw          {SSHA}zRsCkoVvVDXObE3ewn19/Imf3yDoH9
directory       /var/lib/ldap
TLSCipherSuite  HIGH:MEDIUM:+SSLv2
TLSCertificateFile      /etc/openldap/slapdcert.pem
TLSCertificateKeyFile   /etc/openldap/slapdkey.pem
```

TLSCipherSuite specifies a list of OpenSSL ciphers from which slapd will choose when negotiating TLS connections, in decreasing order of preference. To see which ciphers are supported by your local OpenSSL installation, issue this command:

```
openssl ciphers -v ALL
```

In addition to those specific ciphers, you can use any of the wild cards supported by OpenSSL, which allow you to specify multiple ciphers with a single word. For example, in Listing 1 TLSCipherSuite is set to HIGH:MEDIUM:+SSLv2; as it happens, HIGH, MEDIUM and +SSLv2 all are wild cards.

HIGH means "all ciphers using key lengths greater than 128 bits"; MEDIUM is short for "all ciphers using key lengths equal to 128 bits", and +SSLv2 means "all ciphers specified in the SSL protocol, version 2, regardless of key strength". For a complete explanation of OpenSSL ciphers, including all supported wild cards, see the ciphers(1) man page.

TLSCertificateFile and TLSCertificateKeyFile are more obvious. They specify the paths to your certificate file and private key file, respectively. If both certificate and key are combined in a single file, you can specify the same path for both parameters.

slapd Startup Options

We've done everything we need (on the server end) for TLS encryption to work. Only one detail to consider remains. Should we force the use of TLS for all LDAP requests from the network or keep it optional?

By default, slapd listens for LDAP connections on TCP port 389 and accepts either clear text or TLS-encrypted connections on that port. However, if you're using LDAP for authentication, you probably don't want to make TLS optional. A better approach in that case is to have slapd listen for clear text-only LDAP connections on TCP 389 on the loopback interface only and have slapd listen for TLS-enabled (ldaps) connections on TCP 636 (the standard port for ldaps) for all other local addresses.

This behavior is controlled by slapd's startup option -h, used to specify the URLs to which slapd will respond. For example, `slapd -h ldap://127.0.0.1/ ldaps:///` tells slapd to listen on the loopback address (127.0.0.1) for ldap connections to the default ldap port (TCP 389) and to listen on all local addresses for ldaps connections to the default ldaps port (TCP 636).

If you run Red Hat 7.3 or later, this actually is the default behavior: /etc/init.d/ldap checks /etc/openldap/slapd.conf for TLS configuration information and if it finds it, sets the -h option exactly like the one in the previous paragraph's example. If you run SuSE 8.1 or later, you can achieve the same thing by editing /etc/sysconfig/openldap such that the value for OPENLDAP_START_LDAPS is yes, and then editing /etc/init.d/openldap to set the value for SLAPD_URLS to ldap://127.0.0.1. This variable is defined early in the script, with a default value of ldap:///.

Other Linux distributions may have different ways of passing startup options like -h to slapd, but hopefully by now you get the idea and can figure out how to make slapd's listening ports work the way you want.

Testing

So, does our TLS-enabled LDAP server actually work? A quick local test will tell us. First, start LDAP:

```
bash-$ /etc/init.d/ldap start
```

Next, use the ldapsearch command to do a simple query via loopback:

```
bash-$ ldapsearch -x -H ldaps://localhost/ \
-b 'dc=wiremonkeys,dc=org' '(objectclass=*)'
```

Naturally, your own LDAP server will have a different base DN than dc=wiremonkeys,dc=org. If you prefer, you can run this last command from a remote host, specifying the LDAP server's name or IP address in place of localhost in the -h option. If the LDAP server returns a dump of the LDAP database, which actually is empty at this point, followed by the string `result: 0 Success`, your test has succeeded.

If you get an error about an invalid certificate, try adding this line to your client system's /etc/openldap/ldap.conf file:

```
TLS_REQCERT     allow
```

This allows your OpenLDAP or OpenLDAP-based client software (for example, gq) to accept self-signed server certificates.

LDAP Schema

You're almost ready to start populating the LDAP database. On the one hand, tools like gq and ldapbrowser can reduce the ugliness and toil of LDAP data entry and administration greatly. But to get to the point where these tools can be used, you first have to settle on a combination of LDAP schema, and this is where things can get unpleasant.

For purposes of this discussion, two types of LDAP data matter, attributes and object classes. Attributes are the things that make up a record. A user's phone number, e-mail address, nicknames and so on are all attributes. You can use as many or as few attributes in your LDAP database as you like. You even can invent your own. But for a record to contain a given attribute, that record must be associated with the proper object class.

An object class describes the type of record you're trying to build. It defines which attributes are mandatory for each record and which attributes are optional. You might think, "that's easy, then I simply need to choose an object type that provides the group of attributes I want to store for my users and associate each user record with that object class." If you thought that, you'd be only partly right.

In practice, you'll probably want to use attributes from a variety of object classes. "Well, fine", you think, "I'll just specify multiple object classes in each user record and get my full complement of attributes *à la carte*. Whatever." Right again, but there's more to it than that. Chances are the object classes that provide the attributes you need are spread across a number of schema files (these are text files, each of which contains a list of attributes and the object classes that reference them). So, even before you can begin composing your user records, each containing a stack of object class statements and a bigger stack of attribute settings, first you need to make sure /etc/openldap/slapd.conf contains include statements for all the

schema files you need, usually present in /etc/openldap/schema.

For example, suppose that because we're going to use our sample LDAP server for authentication, we want to make sure that no matter what, we're able to specify the attributes userid and userPassword. Doing a quick `grep` of the files in /etc/openldap/schema shows that uid appears in the file inetorgperson.schema in the MAY list (of allowed attributes) for the object class inetOrgPerson. This has two ramifications. First, /etc/openldap/slapd.conf needs to contain this line:

```
include   /etc/openldap/schema/inetorgperson.schema
```

Second, whenever I create a user record, I need to make sure that an `objectclass: inetOrgPerson` statement is present.

Creating and Adding User Records

So, how do you create user records? Ideally, with the GUI of your choice. Last month I mentioned gq, which is a standard package in many distros; another excellent tool is ldapbrowser, available at www.iit.edu/~gawojar/ldap [2]. Initially, however, you'll probably want to add at least your organizational entry manually, by creating an ldif file and writing it to the database via the ldapadd command. An ldif file is a text file containing a list of attribute/object class declarations, one per line; a simple one follows:

```
dn: dc=wiremonkeys,dc=org
objectclass: top
objectclass: organization
o: Wiremonkeys of St. Paul
```

Here, we're defining the organization wiremonkeys.org. We specify its distinguished name, associate it with the object classes' top (mandatory for all records) and organization and specify the organization's name (Wiremonkeys of St. Paul), which is the only mandatory attribute for these two object classes.

To write this record to the database, issue this command:

```
bash-$ ldapadd -x -H ldaps://localhost/ \
-D "cn=ldapguy,dc=wiremonkeys,dc=org" \
-W -f wiremonkeys_init.ldif
```

As with most OpenLDAP commands, -x specifies simple password authentication, -H specifies the LDAP server's URL, -D specifies the DN of the administrator account and -W causes a prompt for the administrator's password. The -f option specifies the path to our ldif file.

Confused yet? I've packed a lot of information into this month's column, but our LDAP server is very nearly done. To finish yours without waiting for next month, see the OpenLDAP Administrator's Guide at www.openldap.org/doc [3] for more information about TLS, startup flags, schema and ldif files.

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).

Links

[1] http://www.mandrakesecure.net/en/docs/ldap-auth.php
[2] http://www.linuxjournal.com/
[3] http://www.openldap.org/doc

Source URL: http://www.linuxjournal.com/article/6876