How to use LDAP with GnuPG

GnuPG.com

2020-12-23

Contents

1 How to use LDAP with GnuPG

In GnuPG the handling of LDAP is done by its Dirmngr component. This is due to the architecture of the system where Dirmngr is the sole process responsible for network related tasks. Network access is required for:

- CRL fetching and caching for S/MIME
- OCSP checking
- S/MIME (X.509) certificate search via LDAP
- OpenPGP keyserver access (HTTP, LDAP, etc.)
- Checking for software updates (if enabled)

In the following we describe how S/MIME and OpenPGP certificate search is implemented. If you want to skip this background information feel free to continue with the next section where LDAP installation and configuration is described. In any case we need to explain a few terms used with LDAP:

DIT *Directory Information Tree* also known as *naming context*. This is is often referred to as the *LDAP directory*. It is where the data for a single organization described by a DNS name is stored (e.g. "example.org").

DN *Distinguished Name* is the key for an entry in the DIT. It is a similar concept as used in the DNS system.

RDN *Relative Distinguished Name* is a component or part of a DN. For example the DN "cn=admin,dc=example,dc=com" consist of the 3 RDNs "cn=admin", "dc=example", and "dc=com". Each RDN has a name (e.g. "cn" for *common name* or "dc" for *domain component*) and a values (e.g. "admin").

LDIF *LDAP Data Interchange Format* is a description for the human readable data exchange format used with LDAP.

1.1 OpenPGP

To serve OpenPGP certificates via LDAP a dedicated schema needs to be installed. The schema supported by GnuPG was originally defined by PGP Inc. in the end of the 1990ies. This is today still the schema installed on LDAP servers for access by PGP or GnuPG. However, this schema has a couple of deficits which need to be fixed. For that reason we have defined additional attributes. These new attributes eventually allow to lookup certificates by their fingerprints and not just by the shorter and thus non-unique Key-ID. The new schema also supports storing of information on the subkeys and the UTF-8 encoded mail addresses. Current versions of GnuPG do not yet make use of these new attributes but for new LDAP installations it is highly recommended to use the new schema so that a future version of the software can make use if these attributes.
Note that the OpenPGP certificates are stored in the DIT under a separate organizational unit using the long Key-ID to distinguish them. An example for such an DN is:

```
pgpCertID=63113AE866587D0A,ou=GnuPG Keys,dc=example,dc=com
```

or for Active Directory

```
cn=C312[...]0A,cn=GnuPG Keys,dc=example,dc=com
```

This design means that entries stored under "GnuPG Keys" are not connected to the users commonly found on an LDAP server. This allows to store arbitrary OpenPGP certificates in the directory and is commonly used to make the certificates of external communication partners easily available.

1.2 S/MIME

Standard X.509 LDAP semantics apply for S/MIME certificate search. The current version of Dirmngr (2.2.23) supports 3 pattern formats which are translated from GnuPG's User-ID syntax, as given to the gpg and gpgsm commands, to the LDAP syntax:

Mail Indicated by a leading left angle and translated to the query:

```
"<ADDRSPEC>" -> "mail=ADDRSPEC"
```

Subject DN Indicated by a leading slash. The DN is formatted according to RFC-2253 rules and thus directly usable for an LDAP query.

Substring search If no other syntax matches or the pattern is prefixed with an asterisk the User-ID is translated to:

```
"USERID" -> "(|(sn=*USERID*)(|(cn=*USERID*)(mail=*USERID*)))"
```

or in other word a substring search on the serial-number, the common-name, and the mail attribute is done.

The result is expected to be in one of the attributes "userCertificate", "cACertificate", or "x509caCert". In cases where we are looking for the issuer certificate only "cACertificate" is used. "ObjectClass=*" is always used a filter.
Note: The attribute "mail" with the OID 0.9.2342.19200300.100.1.3 was originally defined with this OID under the name "rfc822Mailbox" using a different although similar syntax. Take care: This is not an UTF-8 encoded mail address and in theory GnuPG should use IDN mapping here. However, it is questionable whether any real world installation would be able to handle such a mapping.

2 How to install OpenLDAP

To install a standard LDAP server to provide S/MIME certificate lookup follow the instructions of your OS vendor. For example on Debian based systems this is:

```
apt-get install slapd ldap-utils libsasl2-modules
```

Follow the prompts during installation, set an initial admin password, and, most important, the domain you want to serve. Note that we use "example.com" in following. If you ever need to change the configuration on a Debian based system you can do so by running

```
dpkg-reconfigure slapd
```

Serving LDAP requests for S/MIME (X.509) certificates will then work out of the box. Use your standard tools to maintain these entries. Some hints on how to manually add certificates can be found below in the section "Useful LDAP Commands".
Please read on if you want to serve also OpenPGP certificates.

2.1 Installation of the OpenPGP Schema

Assuming a standard OpenLDAP installation, it is easy to add a new schema to store OpenPGP certificate. We describe this now step by step.
First you need to download the two LDIF files

- `https://gnupg.org/misc/gnupg-ldap-schema.ldif`
- `https://gnupg.org/misc/gnupg-ldap-init.ldif`.

As administrator (root) on your LDAP server use the command

```
ldapadd -v -Y EXTERNAL -H ldapi:/// -f ./gnupg-ldap-schema.ldif
```

to install the schema. The options given to the ldapadd tool are:

-v Given some diagnostic output (be verbose). To be even more verbose you may use `-vv` or `-vvv`. The diagnostics are written to stdout.

-Y Specify the authentication mechanism. Here we use `EXTERN` which is in this case local socket based authentication (ldapi).

-H The URL to access the LDAP server. Only scheme, host, and port are allowed. In our case we use `ldapi:///` to request a connection on the standard OpenLDAP socket (usually this is `/var/run/slapd/ldapi`).

-f Specify a file with data to add to the directory. The file used here is the specification of the keyserver schema. If this option is not used ldapadd expects this data on stdin.

The new schema should now be installed. Check this by using this command:

```
ldapsearch -Q -Y EXTERNAL -L -H ldapi:/// \
   -b 'cn=schema,cn=config' cn | grep cn:
```

(on Unix the backslash indicates that the line is continued with the next line)
The options not used by ldapsearch which have not yet been explained above are:

-Q Be quiet about authentication and never prompt.

-b Specify the search base. In this case we want the internal OpenLDAP schema which stores the server's own configuration.

The final argument `cn` restricts the output to the DN and the CN attribute; the grep then shows only the latter. With a freshly installed OpenLDAP system you should get an output like:

```
cn: schema
cn: {0}core
cn: {1}cosine
cn: {2}nis
cn: {3}inetorgperson
cn: {4}gnupg-keyserver
```

This tells you that the keyserver schema has been installed under (in this case) the index "{4}".
The next step is to connect the new schema with your DIT. This means that entries to actually store the certificates and meta data are created. This way GnuPG will be able to find the data. For this you need to edit the downloaded file `gnupg-ldap-init.ldif` and replace all the RDNs with name "dc" with your own. For example, in our own LDAP we would change

```
dn: cn=PGPServerInfo,dc=example,dc=com
```

to

```
dn: cn=PGPServerInfo,dc=gnupg,dc=com
```

and do that also for the other 3 appearances of the "dc" RDNs. In case you use a 3-level domain, add another "dc" in the same way you did when setting up OpenLDAP. With that modified file run

```
ldapadd -v -x -H ldapi:/// -D 'cn=admin,dc=example,dc=com' \
  -W -f ./gnupg-ldap-init.ldif
```

Remember to change the "dc" RDNs also here to what you actually use. We use simple authentication by means of these options:

-x Use simple authentication

-D The Bind-DN used to bind to the LDAP directory

-W Ask for the admin's passphrase. You may also use a lowercase `-w` followed by the passphrase but that would reveal the passphrase in the shell's history etc.

All users with access right to the LDAP server may now retrieve OpenPGP certificates. But wait, we also need a user allowed to insert or update OpenPGP certificates. Choose a useful name for that user and create a file `newuser.ldif`. In our example domain we name that user "LordPrivySeal" and thus the file is:

```
dn: uid=LordPrivySeal,ou=GnuPG Users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: uidObject
sn: Lord Keeper of the Privy Seal
cn: Lord Privy Seal
userPassword: {SSHA}u6oxl9ulaS57RPyjApyPcE7mNECNK1Tg
```

The `userPassword` has been created by running

```
/usr/sbin/slappasswd
```

entering the password, and paste the output into the file (the password used in the above example is "abc").
Now run

```
ldapadd -v -x -H ldapi:/// -D 'cn=admin,dc=gnupg,dc=com' \
   -W -f ./newuser.ldif
```

On the password prompt enter the admin's password (not the one of the new user). Note that the user is created below the "GnuPG Users" organizational unit and not in the standard name space. Thus this is a dedicated user for OpenPGP certificates.
See below how you can list the entire DIT. With a fresh install you should see these DNs:

```
dn: dc=example,dc=com
dn: cn=admin,dc=example,dc=com
dn: cn=PGPServerInfo,dc=example,dc=com
dn: ou=GnuPG Keys,dc=example,dc=com
dn: ou=GnuPG Users,dc=example,dc=com
dn: uid=LordPrivySeal,ou=GnuPG Users,dc=example,dc=com
```

Finally we need to give all users read access to the server's database and allow an authenticated user to modify the database. To do this you need to figure out the used database; run the command

```
ldapsearch -Q -Y EXTERNAL -H ldapi:/// -b 'cn=config' dn | grep olcDatabase=
```

which should give you a list like this:

```
dn: olcDatabase={-1}frontend,cn=config
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}mdb,cn=config
```

The first two databases are for internal purposes, the last one is our database. Now create a file `grantaccess.ldif` with this content:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0} to dn.subtree="dc=example,dc=com"
    by dn.regex="^uid=LordPrivySeal,ou=GnuPG Users,dc=example,dc=com" write
    by * read
```

As usual replace all "dc=example,dc=com" accordingly. Take care not to insert a blank line anywhere. The first line needs to give the DN of the database as determined above. Excute the rules from that file using the command:

```
ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f grantaccess.ldif
```

Now all users have read access and the user LordPrivySeal has write access. In case you want to give several users permissions to update the keys replace the regex line in `grantaccess.ldif` with

```
by dn.regex="^uid=([^,]+),ou=GnuPG Users,dc=example,dc=com" write
```

(take care to insert two spaces at the begin of the line.) Then create those users below the RDN "ou=GnuPG Users".
That's all you need to do at the server.

2.2 Configuration for GnuPG

The easiest way to enable LDAP for S/MIME is to put

```
keyserver ldap.example.com:::dc=example,dc=com:
```

into `gpgsm.conf`. If you prefer to use a dedicated configuration file you can do this with dirmngr by adding a line

```
ldap.example.com:::dc=example,dc=com:
```

to `dirmngr_ldapservers.conf`.
Assuming you want to use the machine running the LDAP server also to maintain OpenPGP certificates, put the following line into the `dirmngr.conf` configuration of a dedicated user for this task:

```
keyserver ldapi:///????bindname=uid=LordPrivySeal
%2Cou=GnuPG%20Users%2Cdc=example%2Cdc=com,password=abc
```

(Enter this all on one line; "%2C" directly at the end of "Seal")
That is a pretty long line with weird escaping rules. Just enter it verbatim but replace the "dc" RDNs accordingly. Remember that `ldapi` uses local socket connection instead of TCP to connect to the server. The password given in that file is the password of the OpenPGP maintainer (LordPrivySeal). Use appropriate permissions for that file to make it not too easy to access that password. See the GnuPG manual for other ways to configure an LDAP keyserver.
With that configuration in place you may add arbitrary OpenPGP keys to your LDAP. For example user "joe@example.org" sends you a key and asks to insert that key. If you feel comfortable with that you should first check the key, import it into your local keyring, and then send it off to your LDAP server:

```
gpg --show-key < file-with-joes-key.asc
```

Looks good? Note the fingerprint of the key and run

```
gpg --import < file-with-joes-key.asc
gpg --send-keys FINGERPRINT
```

That's all. If you want to work from a different machine or use the Kleopatra GUI you need to make sure that ldaps has been correctly configured (for example on the machine `ldap.example.org`) and you need to use this keyserver line:

```
keyserver ldaps://ldap.example.com/????bindname=uid=LordPrivySeal
%2Cou=GnuPG%20Users%2Cdc=example%2Cdc=com,password=abc
```

(Enter this all on one line; "%2C" directly at the end of "Seal")
The easier case is the configuration line for anonymous users which is a mere

```
keyserver ldaps://ldap.example.com
```

This assumes that you have a valid TLS server certificate for that domain and ldaps is enabled on the server.

3 Useful OpenLDAP Commands

3.1 List the entire DIT

To list the entire DIT for the domain "example.com" use this command:

```
ldapsearch -Q -Y EXTERNAL -LLL -H ldapi:/// -b dc=example,dc=com dn
```

This lists just the DNs. If you need the entire content of the DIT leave out the "dn" argument. The option "-LLL" selects useful formatting options for the output.

3.2 Insert X.509 Certficate

If you don't have a handy tool to insert a certificate via LDAP you can do it manually. First put the certificate in binary (DER) format into a file. For example using gpgsm:

```
gpgsm --export berta.boss@example.com >berta.crt
```

Then create a file `addcert.ldif`:

```
dn: CN=Berta Boss,dc=example,dc=com
objectclass: inetOrgPerson
cn: Berta Boss
sn: Boss
gn: Berta
uid: berta
mail: berta.boss@example.com
usercertificate;binary:< file:///home/admin/berta.crt
```

(Note that an absolute file name is required.)
Finally run

```
ldapadd -x -H ldapi:/// -D 'cn=admin,dc=example,dc=com' -W -f adduser.ldif
```

3.3 Change RootDN Password:

Create temporary file named `passwd.ldif`:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: XXXX
```

For XXXX insert the output of slappasswd and run

```
ldapmodify -Q -Y EXTERNAL -H ldapi:/// -f passwd.ldif
```

followed by

```
ldappasswd -x -D cn=admin,dc=example,dc=com -W -S
```

and enter the new and old password again.

3.4 Show ACLs

```
ldapsearch -Q -Y EXTERNAL -H ldapi:/// -b 'cn=config' olcAccess
```

3.5 Show a list of databases

```
ldapsearch -Q -Y EXTERNAL -H ldapi:/// -b 'cn=config' | grep ^olcDatabase:
```

3.6 Change the log level

To debug access problems, it is useful to change the log level:

```
printf "dn: cn=config\nchangetype: %s\nreplace: %s\n%s: %s\n" \
  modify olcLogLevel olcLogLevel ACL | ldapadd -Q -Y EXTERNAL -H ldapi:///
```

to revert replace "ACL" by "none".

4 How to use with Active Directory

4.1 Extending the AD Schema

The Active Directory on Windows is actually an LDAP server but configuration differs from OpenLDAP. The used schema is the same but the data objects are slighly different. To extend the schema the LDIF format is used but with variants of the files used for OpenLDAP. Thus please download these two files:

- `https://gnupg.org/misc/gnupg-ldap-ad-schema.ldif`
- `https://gnupg.org/misc/gnupg-ldap-ad-init.ldif`.

Important: Backup your Active Directory before you extend the schema. There are **no ways to revert changes** made to a schema. You should also first try this all on a test system and not on a production system.
To extend the schema become Adminstrator on your Primary Domain Controller and open a shell (Command Prompt). Copy the above mentioned ldif files to your working directory and run the following command:

```
ldifde -i -v -f gnupg-ldap-ad-schema.ldif
  -c "DC=EXAMPLEDC" "DC=example,DC=org"
```

This is one line and the last string (`"DC=example,DC=org"`) needs to be replaced with your actual domain. If the command succeeds you have extended the schema to store OpenPGP keys at a well known location. The next step is to provide information and space in the tree. This is done similar to the above, namely:

```
ldifde -i -v -f gnupg-ldap-ad-init.ldif
  -c "DC=EXAMPLEDC" "DC=example,DC=org"
```

You may now check your work with ADSI (enter "adsiedit"). Compare with this screenshot and notice the two marked entries.
The last step is to setup permissions. This depends on your policy. Here we assume that all authenticated users get read access to all OpenPGP keys and only certain users may insert or update those keys.
What you need to do in all cases is to give the group *Everyone* read access to the `CN=PGPServerInfo` object. This allows the clients to notice that the schema has been installed and where to look further.
The actual keys will be stored under `CN=GnuPG Keys`. Thus give all users of the *AuthenticatedUsers* group read access and use the Advanced button to set *Applies to* to *This object and all descendant objects*.
To insert and update keys, use a group or users and give them permissions for `CN=GnuPG Keys` to *Read*, *Write*, *Create all child objects*, and *Delete all child objects*. As above make sure that these permissions apply to *This object and all descendant objects*.
In case you want to access the keys also from non-Windows boxes, it is probably best to created a dedicated guest user for read access.

4.2 Using GnuPG with AD

Using the Active Directory is really easy since GnuPG 2.2.26: You only need to put

```
keyserver ldap:///
```

into `dirmngr.conf` and Windows takes care of authentication. Note that we use 3 slashes and not ldaps because AD takes care of protecting the traffic.
GnuPG can be advised to consult the local AD similar to a Web Key Directory. For this put

```
auto-key-locate local,ntds,wkd
```

into `gpg.conf` so that a missing key is first looked up in the AD before a WKD query is done.