GNU/Linux Administration - Feature #162

Installing OpenLDAP with phpLDAPAdmin on Debian

08/08/2013 11:08 AM - Daniel Curtis

Status:	Closed	**Start date:**	08/08/2013
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Domain Controller	**Estimated time:**	2.00 hours
Target version:		**Spent time:**	2.00 hours

Description

The need to store, access, and modify directory information such as user information, corporate contacts, and/or asset management is necessary for centralized scalable information storage. LDAP is an application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. OpenLDAP will be used as the server, and phpLDAPAdmin will be the interface to add, remove, and modify entries to the LDAP server.

Make sure the host has a Fully Qualified Domain Name

```
hostname --fqdn
```

OpenLDAP will automatically configure itself to the domain name of the host it is installed on.

Install OpenLDAP and utilities

```
sudo apt-get install slapd ldap-utils
```

- Enter LDAP admin password: *password*
  To reconfigure the OpenLDAP server for some reason, such as to reassign the domain name the server is registered to:

  ```
  dpkg-reconfigure slapd
  ```

Configure OpenLDAP for to listen for unencrypted connections from localhost

```
vi /etc/ldap/ldap.conf
```

```
#LDAP Defaults
#See ldap.conf(5) for details
#This file should be world readable but not world writable.

BASE dc=example,dc=com
URI ldap://127.0.0.1
#SIZELIMIT 12
#TIMELIMIT 15
#DEREF never
```

And restart the ldap service:

```
sudo service slapd restart
```

Install and Configure phpLDAPAdmin

```
sudo apt-get install phpldapadmin
```

Configure phpLDAPAdmin

```
vi /etc/phpldapadmin/config.php
```

```
$servers = new Datastore();
$servers->newServer('ldap_pla');
$servers->setValue('server','name','My LDAP Server');
$servers->setValue('server','host','192.168.0.2');
$servers->setValue('server','port',389);
$servers->setValue('server','base',array('dc=example,dc=com'));
$servers->setValue('login','bind_id','cn=admin,dc=example,dc=com');
```

Enable phpLDAPAdmin on Apache

```
ln -s /etc/phpldapadmin/apache.conf /etc/apache2/sites-enabled/phpldapadmin
```

(Optional) Add Server Configuration Administrator Access

The file /etc/ldap/slapd.d/cn=config/olcDatabase={0}config.ldif is usually generated during the installation and contains the initial settings. The configuration itself is stored in the ldap database. So modifying this ldif and restarting slapd does NOT change anything! By default, only the unix account root is able to read and write cn=config. In /etc/ldap/slapd.d/cn=config/olcDatabase={0}config.ldif you will find

```
olcAccess: {0}to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage by * break
```

This indicates, that the unix user with group and user id 0 (actually root) is able to access cn=config. As root you will receive all config values by typing:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config"
```

Generate a password for your new user **cn=admin,cn=config**:

```
slappasswd -h {SSHA}
```

- Enter the password twice and note the hash value

Create a temporary ldif e.g. add_adminconfig.ldif with the following content:

```
vi add_adminconfig.ldif
```

```
dn: cn=config
changetype: modify

#usually cn=admin,cn=config is already set by a fresh slapd install
#dn: olcDatabase={0}config,cn=config
#changetype: modify
#add: olcRootDN
#olcRootDN: cn=admin,cn=config

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}theHashValueGeneratedBefore==

#comment this in, if you like to remove root's permission
#to access cn=config; the fallback to unix root is useful
#if cn=admin,cn=config won't work (e.g. lost the password)
#dn: olcDatabase={0}config,cn=config
#changetype: modify
#delete: olcAccess
```

Now let's add this temporary ldif to the slapd config:

```
ldapadd -Y EXTERNAL -H ldapi:/// -f add_adminconfig.ldif
```

You should now find the hashvalue for your password in the output of:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config"
```

The autodetection of cn=config does not work flawlessly (seems to be a security feature). So we need to add the base-dn in /etc/phpldapadmin/config.php:

```
vi /etc/phpldapadmin/config.php
```

```
/* Array of base DNs of your LDAP server. Leave this blank to have phpLDAPadmin auto-detect it for you. */
$servers->setValue('server','base',array('cn=config','dc=example,dc=org'));
```

Now you can login to phpldapadmin with cn=admin,cn=config and your new password set by the steps above. The usual administrative ldap account cn=admin,dc=example,dc=org is not able to see cn=config.

Related issues:

Related to GNU/Linux Administration - Feature #164: Centralized User Authenti...	**Closed**	**08/09/2013**
Related to GNU/Linux Administration - Feature #163: Installing Kerberos 5 on ...	**Closed**	**08/08/2013**
Related to GNU/Linux Administration - Feature #165: Adding Existing Unix User...	**Closed**	**08/12/2013**
Related to GNU/Linux Administration - Support #166: Backing Up LDAP Directory...	**Closed**	**08/12/2013**

History

#1 - 08/08/2013 11:10 AM - Daniel Curtis

- Description updated

#2 - 08/08/2013 01:04 PM - Daniel Curtis

- Description updated

Configuring LDAPS

In the above example LDAP is configured to only allow connections from itself, and LDAP by default does not encrypt the data it serves. LDAPS is analogous to HTTPS, in that it is the same exact communication protocol except it is wrapped with SSL encryption.

Enable ldaps port in /etc/default/slapd:

```
sudo vi /etc/default/slapd
```

```
SLAPD_SERVICES="ldap://127.0.0.1:389/ ldaps:/// ldapi:///"
```

Configuring the certificate (and possibly the CA used) in slapd config :

Add attributes to cn=config:

```
vi oldSSL.ldif
```

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/cacert.pem

add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/private/server-key.pem

add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/server-cert.pem
```

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f ./olcSSL.ldif
```

By default, slapd runs as user/group openldap, so it can't read the key file. On Debian Lenny, the preferred solution to this dilemma seems to be to chown the key to root:ssl-cert, set permissions to 640 and add the user openldap to group ssl-cert:

```
usermod -a -G ssl-cert openldap
```

In Wheezy, not adding openldap to the ssl-cert group caused this in logs:

> main: TLS init def ctx failed: -1

Symptoms:

In slapd debug output:

> [...] TLS: could not set cipher list HIGH:MEDIUM:-SSLv2. (or similar)

In /var/log/syslog:

> [...] main: TLS init def ctx failed: -1

Diagnosis:

If you try to install the OpenLDAP server (slapd) with Debian Lenny, it comes compiled against the GnuTLS library. It means you cannot use an OpenSSL style directive like TLSCipherSuite HIGH:MEDIUM:-SSLv2 in slapd.conf.

Cure:

In /etc/ldap/slapd.conf, either comment out TLSCipherSuite option to let gnutls choose rather sane default for you, or use something like:

> TLSCipherSuite NORMAL

To get all the supported GnuTLS cipher suite names:

```
aptitude install gnutls-bin
man gnutls-cli
```

And skip to TLS/SSL control options section of man page.

To use only 256 bit cyphers, use this (paranoiac?) setting:

> TLSCipherSuite SECURE256:!AES-128-CBC:!ARCFOUR-128:!CAMELLIA-128-CBC:!3DES-CBC:!CAMELLIA-128-CBC

Another useful tool to test server-supported TLS options is to use gnutls-cli-debug. First add ldaps:/// string to the SLAPD_SERVICES option in /etc/default/slapd, restart slapd and then run

```
gnutls-cli-debug -p 636 <fqdn_of_you_ldap_host>
```

That will show you cryptographic suits your LDAP server supports.

Symptoms (round 2)

If you are getting messages such as

> slapd TLS: can't connect: A TLS packet with unexpected length was received..

or

> Could not negotiate a supported cipher suite.

take a wander by this.

Diagnosis:

How did you generate your certificates? If you generated them using OpenSSL, you're going to run into problems. Debian switched over to using gnutls a while ago, and it doesn't play nice with OpenSSL certificates. So, to fix this, check out the next section.

NOTE: On Debian Squeeze openldap is linked with gnutls as well, but works just fine with certificate generated by openssl.

NOTE about the above note: I don't find it to be the case, except for the CA cert. I ended up having to generate a new key & csr to sign with gnutls's certtool and then signing it with my existing openssl created CA like so: certtool --generate-privkey --outfile ldap.gnutls.key certtool --generate-certificate --load-privkey ldap.gnutls.key --outfile ldap.gnutls.crt --load-ca-certificate ca.crt --load-ca-privkey ca.key

Procedure:

You're going to need the gnutls certificate generator: certtool available in gnutls-bin

Run these two commands to generate a new self-signed key (into the current working directory):

```
certtool --generate-privkey --outfile ca-key.pem
certtool --generate-self-signed --load-privkey ca-key.pem --outfile ca-cert.pem
```

Then, update your certificate locations in /etc/ldap/slapd.conf (TLSCertificateFile points to ca-cert.pem and TLSCertificateKeyFile points to ca-key.pem), comment out TLSCACertificateFile, and change TLSVerifyClient to never.

In /etc/ldap/ldap.conf, comment out TLS_CACERT and change TLS_REQCERT to never.

Since the certificate is self-signed, we can't have gnutls trying to verify it (hence the never), otherwise it will never run.

Then restart your services, and you're good (assuming all your links point properly to ldaps://url/).

The openssl way to generate a SSL Key and CSR:

```
openssl req -nodes -sha256 -newkey rsa:2048 -keyout /path/to/PrivateKey.key -out /path/to/CertificateRequest.csr
```

#3 - 02/16/2015 02:25 PM - Daniel Curtis

- Project changed from 22 to GNU/Linux Administration

- Category set to Domain Controller