EDB Postgres LDAP Authentication Guide

Configuring LDAP without SSL

September 2, 2016

EDB Postgres LDAP Authentication Guide
by EnterpriseDB Corporation

EnterpriseDB Corporation, 34 Crosby Drive Suite 100, Bedford, MA 01730, USA
T +1 781 357 3390 F +1 978 589 5701 E info@enterprisedb.com www.enterprisedb.com

Table of Contents

1 Introduction

LDAP (Lightweight Directory Access Protocol) is an authentication protocol that utilizes a standalone server to satisfy authentication requirements for users connecting to an EDB Postgres Advanced Server database. An LDAP server uses a tree-style organization (a directory) that makes it possible for a single server to manage multiple tiers of users.

This guide will walk you through the process of configuring LDAP authentication (without SSL) for an EDB Postgres Advanced Server database. The examples that follow use OpenLDAP; for detailed information about using OpenLDAP, please visit the OpenLDAP project site at:

http://www.openldap.org/doc/admin24/index.html

To recreate the example that follows, you will need an OpenLDAP server running version 2.4.40. You can use the following RPM packages to install OpenLDAP:

```
openldap-2.4.40

openldap-clients-2.4.40

openldap-servers-2.4.40
```

You will also need an EDB Postgres Advanced Server database capable of supporting LDAP authentication running on a Red Hat Enterprise Linux or CentOS (version 6.x or later) server.

Please note that your system configuration may require you to customize the installation and configuraton steps.

2 Installing and Configuring an LDAP Server

In this section, we will walk you through installing and configuring an LDAP server without SSL. The steps in this section should be performed on the host of the LDAP server, and require `root` privileges:

1. Use `yum` to install the OpenLDAP server:

   ```
   sudo yum -y install openldap openldap-clients openldap-servers
   ```

2. Use the `slappasswd` command to generate a salted hash of a password that will be used in the LDAP configuration file. When prompted, enter a password, and confirm the password:

   ```
   # slappasswd
   New password:
   Re-enter new password:
   {SSHA}FQVGOlgUH6vPowoagnydmmHYk5rOsqaO
   ```

3. LDAP uses a configuration file named `olcDatabase={2}bdb.ldif` that is stored in `/etc/openldap/slapd.d/cn\=config`. You must modify the file, adding the root user and the password hash. When included in this file, the root user will have permissions to add other users, groups, organizational units, etc.

   Use your choice of editor to modify following the following parameters, setting the values:

   ```
   olcSuffix: dc=edb,dc=com
   olcRootDN: cn=Manager,dc=edb,dc=com
   olcRootPW: {SSHA}FQVGOlgUH6vPowoagnydmmHYk5rOsqaO
   ```

   If an entry for `olcRootPW` does not exist, then please add that line.

   Note: OpenLDAP will not accept a value of `root` for the `cn` value within the configuration file.

4. You must also modify the `olcDatabase={1}monitor.ldif` file, changing the `dn.base` value to match the value specified in the `olcRootDN` parameter in the `olcDatabase={2}bdb.ldif` file.

   Use your choice of editor to modify following the following parameters, setting the value for the `olcAccess` parameter:

```
olcAccess: {0}to *  by
dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=au
th" read  by dn.base="cn=manager,dc=edb,dc=com" read  by *
none
```

The `root` user for your LDAP system is `cn=manager,dc=edb,dc=com`. The `root` user's password is the hash value returned by `slappasswd`. In this example, the hash value was returned by the value, `p@ssw0rd`.

5. You can hide the password hashes from users who should not have permission to view them by modifying the `olcDatabase={2}bdb.ldif` file. . Use your choice of editor to add the following parameters to the end of the file:

```
olcAccess: {0}to attrs=userPassword by self write by
dn.base="cn=Manager,dc=edb,dc=com" write by anonymous auth by
* none
olcAccess: {1}to * by dn.base="cn=Manager,dc=edb,dc=com" write
by self write by * read
```

6. To enable logging, modify the `cn\=config.ldif` file (located in `/etc/openldap/slapd.d`.), setting the `olcLogLevel` parameter as follows:

```
olcLogLevel: -1
```

Specify a logging level of `-1` to add all debugging entries to the log file.

7. Create a directory in which the log files will reside `(/var/log/slapd)` and change the directory owner to `ldap`:

```
mkdir -p /var/log/slapd
chown ldap:ldap /var/log/slapd
```

8. Modify the /etc/rsyslog.conf file, adding the following entry:

```
local4.*                /var/log/slapd/slapd.log
```

9. Restart the `rsyslog` service:

```
service rsyslog restart
```

10. Modify chkconfig, ensuring that OpenLDAP is configured to start when the system starts up, and start the OpenLDAP `slapd` service

```
# chkconfig slapd on
# service slapd start
```

2.1 Creating an LDAP Tree

An LDAP directory is analogous to a tree. Each node within the tree is called an *entry* and may represent a user, a group, an organizational unit, a domain controller, or other object. Each entry has a distinguished name (`dn`) and one or more attribute/value pairs. The attributes of each entry are determined by the LDAP schema; the examples that follow use the InetOrgPerson schema (distributed with OpenLDAP).

The attribute type associated with each entry specifies the position of the entry within the LDAP tree; for example, an attribute type of:

- `o` identifies an organization
- `ou` identifies an organizational unit
- `cn` identifies a common name

You can use the hierarchy within the tree to simplify authentication management for groups of users.

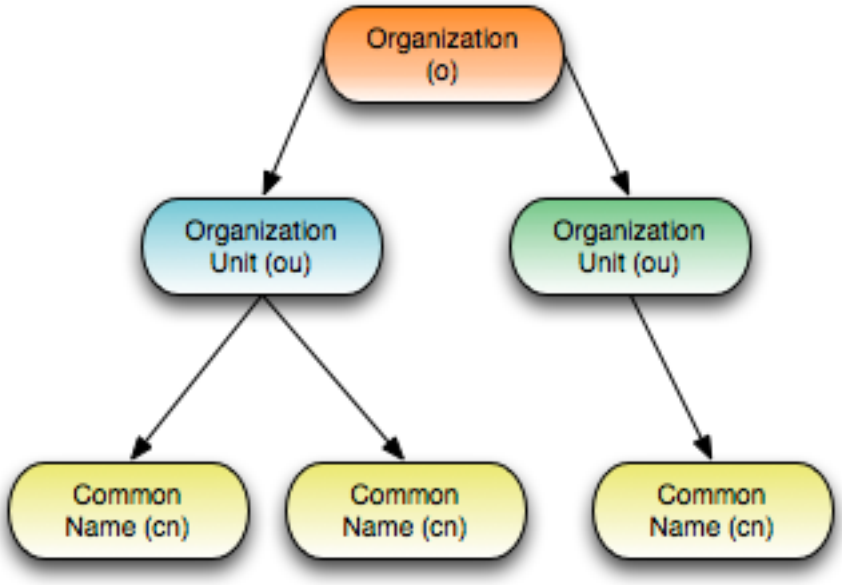

Figure 3.1 – The LDAP directory tree hierarchy.

For more information about identifying and assigning attribute types, please see:

http://www.openldap.org/doc/admin24/intro.html

The first entry in our LDAP tree is for the `root` entry; the root entry should correspond to the `root` identity added to the `olcDatabase={2}bdb.ldif` file.

In the example that follows, we add a tree entry for `edb.com`; to do this, we add an entry to the LDAP tree that specifies:

- a distinguished name (`dn`) entry with a value of `dc=edb,dc=com`.
- a domain component (`dc`) of `edb`.
- a name for the organization (`o`) node within the LDAP tree of `edb`.

1. Create a file called `edb.ldif`. You can delete this file once its content has been added to LDAP; in this example, we will create it in the `/tmp` folder:

```
# cd /tmp
# vi edb.ldif
```

2. Add following entries to the `edb.ldif` file:

```
dn: dc=edb,dc=com
objectClass: dcObject
objectClass: organization
dc: edb
o : edb
```

3. Use the `ldapadd` command to add the contents of the `edb.ldif` file to LDAP:

```
ldapadd -f edb.ldif -D cn=Manager,dc=edb,dc=com -w p@ssw0rd
```

   Please note that you can include the `-w` argument to specify the password in plain text on the command line, or use the `-W` argument to instruct OpenLDAP to prompt you for the password.

4. Use the `ldapsearch` command to verify the entry has been added successfully:

```
# ldapsearch -x -LLL -b dc=edb,dc=com
dn: dc=edb,dc=com
objectClass: dcObject
objectClass: organization
dc: edb
o: edb
```

5. Modify your firewall, allowing connections to the LDAP server on port `389` (the default LDAP port). Providing detailed information about configuring your firewall is beyond the scope of this tutorial.

2.2 Adding an Entry to the LDAP Tree

Each time you add an entry to the LDAP tree, you will create a temporary file that contains an entry for the entity, and then use the `ldapadd` command to update the LDAP Tree. Since the file will be deleted later, you can create the file in `/tmp`.

2.2.1 Adding an Organizational Unit (ou) Entry to the LDAP Tree

To create an entry in the LDAP tree that identifies an organization unit, include the `ou` attribute in the entry definition. The following example demonstrates creating an organizational unit named `EDBUsers`:

1. Create a temporary file called `EDBUsers.ldif`:

   ```
   vi /tmp/EDBUsers.ldif
   ```

2. Add entries to the file that identify the node in the hierarchy; include the `ou` attribute and the node name (`EDBUsers`) in the definition:

   ```
   dn: ou=EDBUsers,dc=edb,dc=com
   objectClass: organizationalUnit
   ou: EDBUsers
   ```

3. Use the `ldapadd` command to add organization unit to the LDAP tree:

   ```
   ldapadd -f EDBUsers.ldif -D cn=Manager,dc=edb,dc=com -w
   p@ssw0rd
   ```

2.2.2 Adding an Entry within an Organization Unit

To create an entry in the LDAP tree that identifies a member of an organization unit, include the `ou` attribute of which the entry is a member as part of the entry's distinguished name. The following example adds an entry to the tree that branches off of the `EDBUsers` node:

1. Create a temporary file called `enterprisedb.ldif`.

   ```
   # vi /tmp/enterprisedb.ldif
   ```

2. Add entries to the file that identify the node in the hierarchy; include the common name (`cn`) attribute to specify the common name of the entry (`enterprisedb`), and the organizational unit (`ou`) attribute to identify the parent node (`EDBUsers`):

   ```
   dn: cn=enterprisedb,ou=EDBUsers,dc=edb,dc=com
   cn: enterprisedb
   sn: edb
   objectClass: inetOrgPerson
   userPassword: edb
   uid: enterprisedb
   ```

3. Use the `ldapadd` command to add the entry to the LDAP tree:

   ```
   # ldapadd -f enterprisedb.ldif -D cn=Manager,dc=edb,dc=com
   -w p@ssw0rd
   ```

2.2.3 Adding a Group to the LDAP Tree

An entry in an LDAP tree can also identify a group; the following example defines a group entry named `SuperUser`:

1. Create a temporary file called `group.ldif`:

   ```
   # vi /tmp/group.ldif
   ```

2. Add entries to the file that identify the node in the hierarchy; include the common name (`cn`) attribute to specify the common name of the entry (`SuperUser`), and the organizational unit (`ou`) attribute to identify the parent node (`EDBUsers`):

   ```
   dn: cn=SuperUser,ou=EDBUsers,dc=edb,dc=com
   cn: SuperUser
   objectClass: groupOfNames
   member: cn=enterprisedb,ou=EDBUsers,dc=edb,dc=com
   ```

3. Use the `ldapadd` command to add the entry to the LDAP tree:

   ```
   # ldapadd -f group.ldif -D cn=Manager,dc=edb,dc=com -w
   p@ssw0rd
   adding new entry "cn=SuperUser,ou=EDBUsers,dc=edb,dc=com"
   ```

2.2.4 Adding a User to a Group

Before you add a user to an LDAP group, the user must first be added to the LDAP tree as an individual entry. For more information, see Section 2.2.2.

1. Create a temporary file called `addUserToGroup.ldif`:

   ```
   # vi /tmp/addUserToGroup.ldif
   ```

2. Add entries to the file that identify the node in the hierarchy. Include the common name (`cn`) attribute to specify the common name of the entry (`postgres`), and ensure that the common name (cn) of the parent node (`SuperUser`) and organizational unit (`ou`) attribute in which the node belongs (`EDBUsers`) are included in the distinguished name (`dn`):

   ```
   dn: cn=SuperUser,ou=EDBUsers,dc=edb,dc=com
   changetype: modify
   add: member
   member: cn=postgres,ou=EDBUsers,dc=edb,dc=com
   ```

   Include the `changetype:modify` clause to specify that the entry will modify an existing entry in the LDAP tree.

   Include the add `keyword` to specify that the entry should be added to the `SuperUser` group.

3. Use the `ldapadd` command to modify the entry in the LDAP tree:

   ```
   # ldapadd -f addUserToGroup.ldif -D
   cn=Manager,dc=edb,dc=com -w p@ssw0rd
   ```

3 Configuring LDAP on the Advanced Server Host

After configuring the LDAP server, and defining the entries in the LDAP tree, you're ready to install and configure LDAP on the EDB Postgres Advanced Server host. LDAP authentication is supported in two modes:

- Simple Bind: The Postgres server binds to the distinguished name of the LDAP entry.
- Search and Bind: The Postgres server binds first to the LDAP tree with a username and password, and then searches for the user that is logging in.

For detailed information about using LDAP with Advanced Server, please see the PostgreSQL core documentation at:

https://www.postgresql.org/docs/current/static/auth-methods.html

Use yum to install *openldap-clients* on the host of the EDB server:

```
yum install openldap-clients
```

3.1 Configuring Simple Bind Authentication

Use the following steps to configure simple bind LDAP authentication on an Advanced Server host:

1. Use yum to install `openldap-clients` on the host of the EDB server:

   ```
   yum install openldap-clients
   ```

2. When the installation completes, assume the identity of the EDB Postgres service user, and use the `ldapsearch` to verify the search string for the `postgres` user:

   ```
   -bash-4.2$  ldapsearch -x -W -h ldapkerb.cloudapp.net -D
   "cn=postgres,ou=EDBUsers,dc=edb,dc=com" -b
   "ou=EDBUsers,dc=edb,dc=com"
   Enter LDAP Password:
   ```

   If the command is successful, it will return:

   ```
   # extended LDIF
   #
   # LDAPv3
   # base <ou=EDBUsers,dc=edb,dc=com> with scope subtree
   # filter: uid=postgres
   # requesting: ALL
   #
   # postgres, EDBUsers, edb.com
   dn: cn=postgres,ou=EDBUsers,dc=edb,dc=com
   cn: postgres
   sn: edb
   objectClass: inetOrgPerson
   userPassword:: ZWRi
   uid: postgres

   # search result
   search: 2
   result: 0 Success

   # numResponses: 2
   # numEntries: 1
   ```

   If you receive an error message, please check at the `slapd.log` file (located in `/var/log/slapd`).

3. Modify the `pg_hba.conf` file of EDB Postgres Advanced Server instance, adding an entry that allows connections from the LDAP server:

```
host  all   all    0.0.0.0/0  ldap
ldapserver=ldapkerb.cloudapp.net ldapprefix="cn="
ldapsuffix=",ou=EDBUsers,dc=edb,dc=com"
```

Please note: you can replace the CIDR address in the above command (`0.0.0.0/0`) with the IP address of the LDAP server. The example shown above demonstrates a simple-bind LDAP configuration.

4. Reload the `pg_hba.conf` file configuration parameters:

```
service ppas-9.5 reload
```

3.2 Configuring Search+Bind Authentication

Use the following steps to configure `search+bind` configuration on an Advanced Server host:

1. 1. Use yum to install `openldap-clients` on the host of the EDB server:

   ```
   yum install openldap-clients
   ```

2. When the installation completes, assume the identity of the EDB Postgres service user, and use the ldapsearch to verify the ldapsearch + bind using bindDB and bindDN password as given below:

   ```
   ldapsearch -x -W -h ldapkerb.cloudapp.net -D
   "cn=enterprisedb,ou=EDBUsers,dc=edb,dc=com" -b
   "ou=EDBUsers,dc=edb,dc=com" "uid=postgres"
   ```

   If the command is successful, it will return:

   ```
   ldapsearch -x -W -h ldapkerb.cloudapp.net -D
   "cn=enterprisedb,ou=EDBUsers,dc=edb,dc=com" -b
   "ou=EDBUsers,dc=edb,dc=com" "uid=postgres"
   Enter LDAP Password:
   # extended LDIF
   #
   # LDAPv3
   # base <dc=edb,dc=com> with scope subtree
   # filter: uid=postgres
   # requesting: ALL
   #

   # postgres, EDBUsers, edb.com
   dn: cn=postgres,ou=EDBUsers,dc=edb,dc=com
   cn: postgres
   sn: edb
   objectClass: inetOrgPerson
   userPassword:: ZWRi
   uid: postgres

   # search result
   search: 2
   result: 0 Success

   # numResponses: 2
   # numEntries: 1
   ```

5. Modify the `pg_hba.conf` file of EDB Postgres Advanced Server instance, adding an entry that allows connections from the LDAP server:

```
host   all   all   0.0.0.0/0   ldap
ldapserver=ldapkerb.cloudapp.net
ldapbasedn="ou=EDBUsers,dc=edb,dc=com"
ldapbinddn="cn=enterprisedb,ou=EDBUsers,dc=edb,dc=com"
ldapsearchattribute=uid ldapbindpasswd=edb
```

6. Reload the `pg_hba.conf` file configuration parameters:

```
service ppas-9.5 reload
```

7. Verify the database connection using psql command as given below:

```
-bash-4.2$ psql -h localhost -U postgres edb
Password for user postgres:
psql.bin (9.5.2.7)
Type "help" for help.

edb=>
```

4 Troubleshooting:

If you encounter an error message, please check:

- the EDB Postgres server log file (located in the `pg_log` directory under your Advanced Server installation).
- the `postgresql.log` file.
- the LDAP server log file `slapd.log` (In our example, located in `/var/log/slapd`)