DB2 - LDAP

Introduction

LDAP is Lightweight Directory Access Protocol. LDAP is a global directory service, industry-standard protocol, which is based on client-server model and runs on a layer above the TCP/IP stack. The LDAP provides a facility to connect to, access, modify, and search the internet directory.

The LDAP servers contain information which is organized in the form of a directory tree. The clients ask server to provide information or to perform some operation on a particular information. The server answers the client by providing required information if it has one, or it refers the client to another server for action on required information. The client then acquires the desired information from another server.

The tree structure of directory is maintained same across all the participating servers. This is a prominent feature of LDAP directory service. Hence, irrespective of which server is referred to by the client, the client always gets required information in an error-free manner. Here, we use LDAP to authenticate IBM DB2 as a replacement of operating system authentication.

There are two types of LDAP:

1. Transparent
2. Plug-in

Let us see how to configure a transparent LDAP.

Configuring transparent LDAP

To start with configuration of transparent LDAP, you need to configure the LDAP server.

LDAP server configuration

Create a SLAPD.conf file, which contains all the information about users and group object in the LDAP. When you install LDAP server, by default it is configured with basic LDAP directory tree on your machine.

The table shown below indicates the file configuration after modification.

The text highlighted with yellow the code box means for the following:

DBA user-id = “db2my1”, group = “db1my1adm”, password= “db2my1” Admin user-id = “my1adm”, group = “dbmy1ctl”.

```
# base dn: example.com
dn: dc=example,dc=com
dc: example
o: example
objectClass: organization
objectClass: dcObject
# pc box db
dn: dc=db697,dc=example,dc=com
dc: db697
o: db697
objectClass: organization
objectClass: dcObject
#
# Group: dbadm
#
dn: cn=dbmy1adm,dc=db697,dc=example,dc=com
cn: dbmy1adm
objectClass: top
objectClass: posixGroup
gidNumber: 400
```

```
objectClass: groupOfNames
member: uid=db2my1,cn=dbmy1adm,dc=db697,dc=example,dc=com
memberUid: db2my1
#
# User: db2
#
dn: uid=db2my1,cn=dbmy1adm,dc=db697,dc=example,dc=com
cn: db2my1
sn: db2my1
uid: db2my1
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
uidNumber: 400
gidNumber: 400
loginShell: /bin/csh
homeDirectory: /db2/db2my1
#
# Group: dbctl
#
dn: cn=dbmy1ctl,dc=db697,dc=example,dc=com
cn: dbmy1ctl
objectClass: top
objectClass: posixGroup
gidNumber: 404
objectClass: groupOfNames
member: uid=my1adm,cn=dbmy1adm,dc=db697,dc=example,dc=com
memberUid: my1adm
#
# User: adm
#
dn: uid=my1adm,cn=dbmy1ctl,dc=db697,dc=example,dc=com
cn: my1adm
sn: my1adm
uid: my1adm
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
uidNumber: 404
gidNumber: 404
loginShell: /bin/csh
homeDirectory: /home/my1adm
```

Save the above file with name '/var/lib/slapd.conf', then execute this file by following command to add these values into LDAP Server. This is a linux command; not a db2 command.

```
ldapadd r- -D 'cn=Manager,dc=example,dc=com" -W -f
/var/lib/slapd.conf
```

After registering the DB2 users and the DB2 group at the LDAP Server, logon to the particular user where you have installed instance and database. You need to configure LDAP client to confirm to client where your server is located, be it remote or local.

LDAP client configuration

The LDAP Client configuration is saved in the file 'ldap.conf'. There are two files available for configuration parameters, one is common and the other is specific. You should find the first one at '/etc/ldap.conf' and the latter is located at '/etc/openldap/ldap.conf'.

The following data is available in common LDAP client configuration file

```
# File: /etc/ldap.conf
# The file contains lots of more entries and many of them
# are comments. You show only the interesting values for now
host localhost
base dc=example,dc=com
ldap_version 3
```

```
pam_password crypt
pam_filter objectclass=posixAccount
nss_map_attribute uniqueMember member
nss_base_passwd dc=example,dc=com
nss_base_shadow dc=example,dc=com
nss_base_group dc=example,dc=com
```

You need to change the location of server and domain information according to the DB2 configuration. If we are using server in same system then mention it as 'localhost' at 'host' and at 'base' you can configure which is mentioned in 'SLAPD.conf' file for LDAP server.

Pluggable Authentication Model *PAM* is an API for authentication services. This is common interface for LDAP authentication with an encrypted password and special LDAP object of type posixAccount. All LDAP objects of this type represent an abstraction of an account with portable Operating System Interface *POSIX* attributes.

Network Security Services *NSS* is a set of libraries to support cross-platform development of security-enabled client and server applications. This includes libraries like SSL, TLS, PKCS S/MIME and other security standards.

You need to specify the base DN for this interface and two additional mapping attributes. OpenLDAP client configuration file contains the entries given below:

```
host localhost
base dc=example,dc=com
```

Till this you just define the host of LDAP serve and the base DN.

Validating OpenLDAP environment

After you configured your LDAP Server and LDAP Client, verify both for communication.

Step1: Check your Local LDAP server is running. Using below command:

```
ps -ef | grep -i ldap
```

This command should list the LDAP deamon which represents your LDAP server:

```
/usr/lib/openldap/slapd -h ldap:/// -u ldap -g ldap -o slp=on
```

This indicates that you LDAP server is running and is waiting for request from clients. If there is no such process for previous commands you can start LDAP server with the 'rcldap' command.

```
rcldap start
```

When the server starts, you can monitor this in the file '/var/log/messages/ by issuing the following command.

```
tail –f /var/log/messages
```

Testing connection to LDAP server with ldapsearch

The ldapsearch command opens a connection to an LDAP server, binds to it and performs a search query which can be specified by using special parameters '-x' connect to your LDAP server with a simple authentication mechanism by using the –x parameter instead of a more complex mechanism like Simple Authentication and Security Layer *SASL*

```
ldapsearch –x
```

LDAP server should reply with a response given below, containing all of your LDAP entries in a LDAP Data Interchange Format*LDIF*.

```
# extended LDIF
```

```
#
# LDAPv3
# base <> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
# example.com
dn: dc=example,
dc=com  dc: example
o: example
objectClass: organization
objectClass: dcObject
# search result
search: 2
result: 0 Success
# numResponses: 2
# numEntries: 1
```

Configuring DB2

After working with LDAP server and client, you need to configure our DB2 database for use with LDAP. Let us discuss, how you can install and configure your database to use our LDAP environment for the DB2 user authentication process.

Configuring DB2 and LDAP interaction plug-ins

IBM provides a free package with LDAP plug-ins for DB2. The DB2 package includes three DB2 security plug-ins for each of the following:

- server side authentication
- client side authentication
- group lookup

Depending upon your requirements, you can use any of the three plug-ins or all of them. This plugin do not support environments where some users are defined in LDAP and others in the operating Systems. If you decide to use the LDAP plug-ins, you need to define all users associated with the database in the LDAP server. The same principle applies to the group plug-in.

You have to decide which plug-ins are mandatory for our system. The client authentication plug-ins used in scenarios where the user ID and the password validation supplied on a CONNECT or ATTACH statement occurs on the client system. So the database manager configuration parameters SRVCON_AUTH or AUTHENTICATION need to be set to the value CLIENT. The client authentication is difficult to secure and is not generally recommended. Server plug-in is generally recommended because it performs a server side validation of user IDs and passwords, if the client executes a CONNECT or ATTACH statement and this is secure way. The server plug-in also provides a way to map LDAP user IDs DB2 authorization IDs.

Now you can start installation and configuration of the DB2 security plug-ins, you need to think about the required directory information tree for DB2. DB2 uses indirect authorization which means that a user belongs to a group and this group was granted with fewer authorities. You need to define all DB2 users and DB2 groups in LDAP directory.

IMAGE

The LDIF file openldap.ldif should contain the code below:

```
#
# LDAP root object
# example.com
#
dn: dc=example,
dc=com
dc: example
o: example
objectClass: organization
objectClass: dcObject
```

```
#
# db2 groups
#
dn: cn=dasadm1,dc=example,dc=com
cn: dasadm1
objectClass: top
objectClass: posixGroup
gidNumber: 300
objectClass: groupOfNames
member: uid=dasusr1,cn=dasadm1,dc=example,dc=com
memberUid: dasusr1
dn: cn=db2grp1,dc=example,dc=com
cn: db2grp1
objectClass: top
objectClass: posixGroup
gidNumber: 301
objectClass: groupOfNames
member: uid=db2inst2,cn=db2grp1,dc=example,dc=com  memberUid: db2inst2
dn: cn=db2fgrp1,dc=example,dc=com
cn: db2fgrp1
objectClass: top
objectClass: posixGroup
gidNumber: 302
objectClass: groupOfNames
member: uid=db2fenc1,cn=db2fgrp1,dc=example,dc=com
memberUid: db2fenc1
#
# db2 users
#
dn: uid=dasusr1,
cn=dasadm1,
dc=example,dc=com
cn: dasusr1
sn: dasusr1
uid: dasusr1
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
uidNumber: 300
gidNumber: 300
loginShell: /bin/bash
homeDirectory: /home/dasusr1
dn: uid=db2inst2,cn=db2grp1,dc=example,dc=com
cn: db2inst2
sn: db2inst2
uid: db2inst2
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
uidNumber: 301
gidNumber: 301
loginShell: /bin/bash
homeDirectory: /home/db2inst2
dn: uid=db2fenc1,cn=db2fgrp1,dc=example,dc=com
cn: db2fenc1
sn: db2fenc1
uid: db2fenc1
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
uidNumber: 303
gidNumber: 303
loginShell: /bin/bash
homeDirectory: /home/db2fenc1
```

Create a file named ‘db2.ldif’ and paste the above example into it. Using this file, add the defined structures to your LDAP directory.

To add the DB2 users and DB2 groups to the LDAP directory, you need to bind the user as ‘rootdn’ to the LDAP server in order to get the exact privileges.

Execute the following syntaxes to fill the LDAP information directory with all our objects defined in the LDIF file ‘db2.ldif’

```
ldapadd –x –D “cn=Manager, dc=example,dc=com” –W –f <path>/db2.ldif
```

Perform the search result with more parameter

```
ldapsearch –x |more
```

Preparing file system for DB2 usage

Creating instance for our LDAP user db2inst2. This user requires home directory with two empty files inside the home directory. Before you create a new instance, you need to create a user who will be the owner of the instance.

After creating the instance user, you should have to create the file ‘.profile’ and ‘.login’ in user home directory, which will be modified by DB2. To create this file in the directory, execute the following command:

```
mkdir /home/db2inst2
mkdir /home/db2inst2/.login
mkdir /home/db2inst2/.profile
```

You have registered all users and groups related with DB2 in LDAP directory, now you can create an instance with the name ‘db2inst2’ with the instance owner id ‘db2inst2’ and use the fenced user id ‘db2fenc1’, which is needed for running user defined functions *UDFs*or stored procedures.

```
/opt/ibm/db2/V10.1/instance/db2icrt –u db2fenc1 db2inst2
DBI1070I Program db2icrt completed successfully.
```

Now check the instance home directory. You can see new sub-directory called ‘sqllib’ and the .profile and .login files customized for DB2 usage.

Configuring authentication public-ins for LDAP support in DB2

Copy the required LDAP plug-ins to the appropriate DB2 directory:

```
cp            ///v10/IBMLDAPauthserver.so
/home/db2inst2/sqllib/security/plugin/server/.

cp            ///v10/IBMLDAPgroups.so
/home/db2inst2/sqllib/security/plugin/group/.
```

Once the plug-ins are copied to the specified directory, you toned to login to DB2 instance owner and change the database manager configuration to use these plug-ins.

```
Su – db2inst2
db2inst2> db2 update dbm cfg using svrcon_pw_plugin
IBMLDAPauthserver
db2inst2> db2 update dbm cfg using group_plugin
IBMLDAPgroups
db2inst2> db2 update dbm cfg using authentication
SERVER_ENCRYPT
db2inst2> db2stop
db2inst2> db2start
```

This modification comes into effect after you start DB2 instance. After restarting the instance, you need to install and configure the main DB2 LDAP configuration file named “IBMLDAPSecurity.ini” to make DB2 plug-ins work with the current LDAP configuration.

IBMLDAPSecurity.ini file contains

```
;-------------------------------------------------------------
; SERVER RELATED VALUES
;-------------------------------------------------------------
; Name of your LDAP server(s).
; This is a space separated list of LDAP server addresses,
; with an optional port number for each one:
; host1[:port] [host2:[port2] ... ]
; The default port number is 389, or 636 if SSL is enabled.
LDAP_HOST = my.ldap.server
;-------------------------------------------------------------
; USER RELATED VALUES
;-------------------------------------------------------------
rs
; LDAP object class used for use USER_OBJECTCLASS = posixAccount
; LDAP user attribute that represents the "userid"
; This attribute is combined with the USER_OBJECTCLASS and
; USER_BASEDN (if specified) to construct an LDAP search
; filter when a user issues a DB2 CONNECT statement with an
; unqualified userid. For example, using the default values
; in this configuration file, (db2 connect to MYDB user bob
; using bobpass) results in the following search filter:
OrgPerson)(uid=bob)
; &(objectClass=inet USERID_ATTRIBUTE = uid
representing the DB2 authorization ID
; LDAP user attribute, AUTHID_ATTRIBUTE = uid
;-------------------------------------------------------------
; GROUP RELATED VALUES
;-------------------------------------------------------------
ps
; LDAP object class used for grou GROUP_OBJECTCLASS = groupOfNames
at represents the name of the group
; LDAP group attribute th GROUPNAME_ATTRIBUTE = cn
; Determines the method used to find the group memberships
; for a user. Possible values are:
; SEARCH_BY_DN - Search for groups that list the user as
; a member. Membership is indicated by the
; group attribute defined as
; GROUP_LOOKUP_ATTRIBUTE.
; USER_ATTRIBUTE - A user's groups are listed as attributes
; of the user object itself. Search for the
; user attribute defined as
TRIBUTE to get the groups.
; GROUP_LOOKUP_AT GROUP_LOOKUP_METHOD = SEARCH_BY_DN
; GROUP_LOOKUP_ATTRIBUTE
; Name of the attribute used to determine group membership,
; as described above.
llGroups
; GROUP_LOOKUP_ATTRIBUTE = ibm-a GROUP_LOOKUP_ATTRIBUTE = member
```

Now locate the file IBMLDAPSecurity.ini file in the current instance directory. Copy the above sample contents into the same.

```
Cp
//db2_ldap_pkg/IBMLDAPSecurity.ini
/home/db2inst2/sqllib/cfg/
```

Now you need to restart your DB2 instance, using two syntaxes given below:

```
db2inst2> db2stop

Db2inst2> db2start
```

At this point, if you try ‘db2start’ command, you will get security error message. Because, DB2 security configuration is not yet correctly configured for your LDAP environment.

Customizing both configurations

Keep LDAP_HOST name handy, which is configured in slapd.conf file.

Now edit IMBLDAPSecurity.ini file and type the LDAP_HOST name. The LDAP_HOST name in both the said files must be identical.

The contents of file are as shown below:

```
;-------------------------------------------------------------
; SERVER RELATED VALUES
;-------------------------------------------------------------
LDAP_HOST = localhost
;-------------------------------------------------------------
; USER RELATED VALUES
-----------------------------
;-------------------------------
USER_OBJECTCLASS = posixAccount
USER_BASEDN = dc=example,dc=com
USERID_ATTRIBUTE = uid
AUTHID_ATTRIBUTE = uid
;-------------------------------------------------------------
; GROUP RELATED VALUES
;-------------------------------------------------------------
GROUP_OBJECTCLASS = groupOfNames
GROUP_BASEDN = dc=example,dc=com
GROUPNAME_ATTRIBUTE = cn
GROUP_LOOKUP_METHOD = SEARCH_BY_DN
GROUP_LOOKUP_ATTRIBUTE = member
```

After changing these values, LDAP immediately takes effect and your DB2 environment with LDAP works perfectly.

You can logout and login again to ‘db2inst2’ user.

Now your instance is working with LDAP directory.