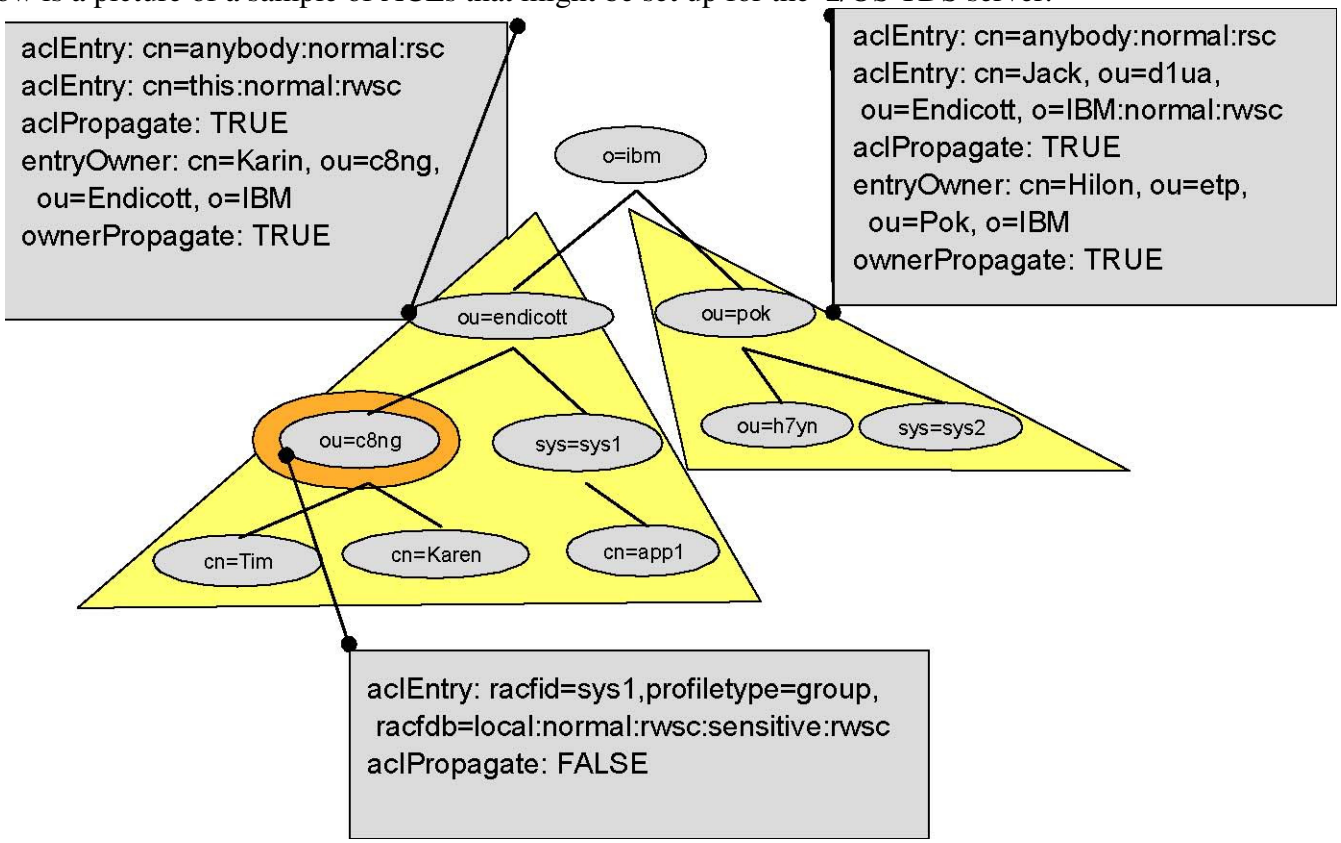
Hands-on Lab: Optional Lab #1

Building and using ACLs (Access Control Lists) for the non-SDBM data.

Background:

The z/OS TDS server can be very flexible in how it protects access to its data. The data in this discussion is the LDAP data that is stored in the TDBM (DB2) and LDBM (USS file) backend stores. The RACF (SDBM) data that is accessed via the z/OS TDS server is protect by RACF controls. Only users with the RACF authority can access data via the SDBM. Therefore, ACLs are not used or needed for the SDBM data. But ACLs are needed to protect the data that is stored in the TDBM and LDBM backend stores. The z/OS TDS server comes set up with a default ACL and this propagated down to all the entries as the z/OS TDS server is configured. Once the z/OS TDS server is configured and ready for operations, the ACLs can be set up to meet the demands of the security policy and applications. This exercise will demonstrate how ACLs can use displayed, added, and modified to the z/OS TDS server. It will also show how ACLs can be propagated to lower entries within the directory tree structure or overridden if needed. And lastly, we will demonstrate how special z/OS features, such as RACF and pseudo-DNs can be used to add flexibility to the protection of the TDBM and/or LDBM data.

Below is a picture of a sample of ACLs that might be set up for the z/OS TDS server.



Exercise Instructions:

Phase 1 - Verify that your z/OS TDS server is running correctly (you can bypass this phase if you have just completed the DSCONFIG lab):

1. Logon to the MVS system with the information provided by the lab instructor(s).

2. Get into SDSF in DA option. Check to see if your z/OS TDS server is running. Your z/OS TDS server's proc (STC) name will be provided by the lab instructor.

3. In OMVS, check to be sure that there is a subdirectory in your home directory called **tdslab/tdsacl**. This contains several Unix scripts that execute LDAP commands to set up your ACLs and to test that they are correct. To check if this subdirectory exist (and to place yourself in the subdirectory), issue the following command from you home directory (your home directory is **/sharelab/sharyxx** – where **y** is your room letter and **xx** is your userid number):

cd tdslab/tdsacl

4. Edit the file called **tdbmivp** by entering the following command: **oedit tdbmivp**

5. Change '####' to your unsecured port number. Save and exit the file by pressing the PF3 key.

6. Run the **tdbmivp** file. This will display data from the TDBM (the DB2 database).

7. Now we want to repeat the last 3 steps for the SDBM. Edit the file called **sdbmivp**. Change the '??' to your userid number. Change '!' to your room letter. Change '%%%%%%%%' to your RACF password. Change '####' to your unsecure port number. Exit and save the file. Run the **sdbmivp** file. This will display data from the SDBM (the RACF database). If both of these commands run correctly, we know that your z/OS TDS server is set up correctly and that we have connectivity to it.

Phase 2 - Check the default ACL that comes with the LDAP server:

8. Edit the **listrootacl** file. Change '####' to your unsecured port number and save the file. This file will display the initial default ACL that is currently protecting your LDAP Server. Run the **listrootacl** script file. Examine the result that is display on your screen. Notice that **cn=anybody** (which is similar to the RACF UACC) can do anything to normal and senitive data but can not even view critical data.

9. To demonstrate the differences between cn=anybody and an authenticate user edit the following scripts: Change '####' to your unsecured port number in the **listunauth** file. In this ldap request, we are doing an ldapsearch without an authenticated user. Notice that we get the information about the cn=admin,o=sharedb2 user and that there is no userpassword displayed. The userpassword attribute is an example of critical data. Edit the file: **listauth**. Change the '####' to your unsecured port number, save the file, and then run it. In this request, you are authenticating with a user that is authorized to critical data. Notice on the output from this request that the administrator id has the authority to display all types of data including critical data - therefore you should see the userpassword is now displayed for the cn=admin.o=sharedb2 user.

Phase 3 - Add a new ACL to the LDAP server (list the old ones, then add the new ACLs):

10. Now edit the **acllist** file. Change the '####' to your unsecured port number and save the file. This file will return the ACL attributes of each entry under the root **'o=sharedb2'**. If you examine the ldapsearch command in the file, you will notice that we have requested that all the ACL attributes be displayed. This is necessary because if they are not explicitly requested they will not be returned by the ldapsearch command. Run the **acllist** file.

11. Now we are ready to start adding ACL information. In this step you will add an additional aclEntry value to the ACL to allow another person or group to access the entry(s). Edit the **aclinfo** file. Change the '####' to your unsecured port number and save the file. Run the **aclinfo** file. This will display the current ACL values for an entry (in this case, Mary Burnnet). Notice that current ACL lists the **aclsource** as o=sharedb2 and the **ownersource** of default. This means that we are propagating the ACL from o=sharedb2 (the one we displayed in the previous step) for this entry.

12. Now browse (no changes are required) the **addaclentry.ldif** file. Once we load this file into the LDAP Server, we will add Judy Simms to the ACL for Mary Burnnet. Exit this file

13. Edit the **addaclentry** file. Change the '####' to your unsecured port number and save the file. Then run the **addaclentry** file. This will load the new ACL information that we reviewed in the previous step (**addaclentry.ldif**). Now rerun the **aclinfo** file to display the new ACL information for Mary Burnnet. Note that 1 attribure has changed (aclsource) and a new aclentry has been added.

Phase 4 - Build and maintain Group definitions within the LDAP server:

14. Now we are going to define a group for the ACL. First browse the **addgroup.ldif** file. No changes are required. This file will load a group definition into the LDAP directory and identify which users are members of the group. Exit the file.

15. Edit the **addgroup** file. Change the '####' to your unsecured port number and save the file. Then run the **addgroup** file. This will load the **addgroup.ldif** data and then display the results for you. You now have a group defined in your LDAP directory.

16. To change the members within a group definition, you use the ldapmodify command. Review the **delmem.ldif** file (no changes are required in this file). Note that we are going to delete David Delbert from this group. Exit this file and edit file **delmem**. Change all the '####' to your unsecured port number and save the file. Run the **delmem** file. This will delete the member and display the new results for you. Note that you have deleted David Delbert from the group.

Phase 5 - Add our Group to the ACL:

17. Now we can add the group to our ACL. Edit the **addaclgroup** file. Change the '####' to your unsecured port number and save the file. Run the **addaclgroup** file. Then rerun the the **aclinfo** file to display our ACL for Mary Burnnet.

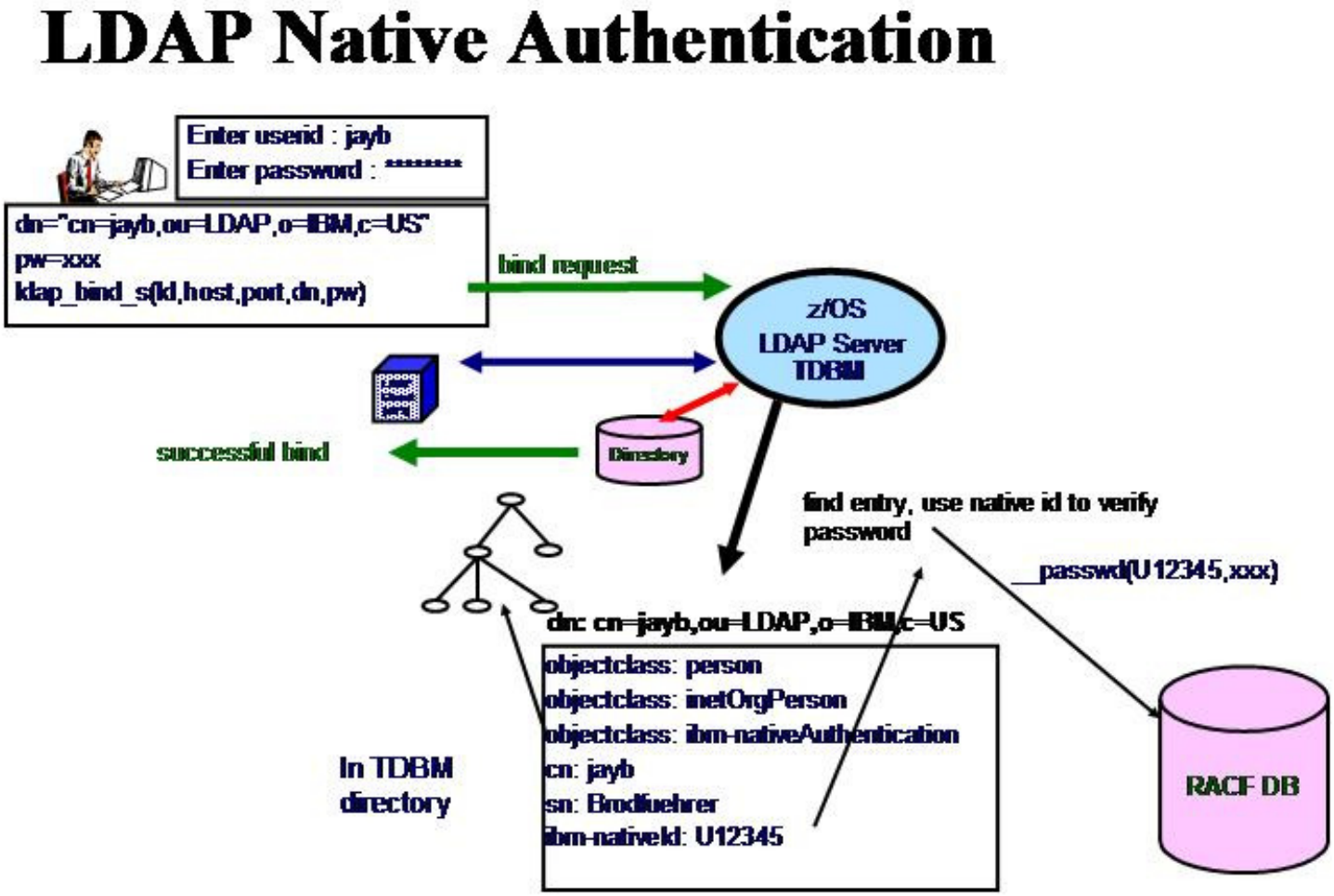
18. We have now built a new ACL for the Mary Burnnet entry. What access does group1 have to the entry with the dn of Mary Burnnet?

Hands-on Lab: Optional Lab #2

Setting up Native Authentication for the z/OS LDAP Server.

Background:

The z/OS TDS server can be set up to use data that is stored in a TDBM or LDBM backend store. It can also use RACF, the z/OS security environment to bind (authenticate) users to the z/OS TDS Server. Sometimes it is an advantage to combine these two functions. To use the security database (RACF) to store highly sensitive password information, while using the DB2 backend store for the more flexible and non-security related data that users and applications need to perform their daily business is a big advantage of the z/OS TDS server. This can be done very easily and in a fashion that the applications and users are not impacted. In other words, all the changes and configuration is done within the z/OS TDS server. This is called native authentication. Below is a picture that demonstrates the inner workings of native authentication:



As depicted, the LDAP client performs an LDAP Bind with a 'standard' DN and the password. By 'standard' DN, this means a DN that follows the X.500 or dc format for DN. The z/OS TDS server will find the entry with the requested DN. The z/OS TDS server will check to see if this entry is within the scope of native authentication, if native

authentication has been configured. If this z/OS TDS server has been configured for native authentication and if this entry has been set up to use native authentication, then the z/OS TDS server will find the associated RACF userid for this entry and perform a SAF call to RACF to authenicate this user.

Exercise Instructions:

Phase 1 - Verify that your LDAP server is running correctly (you can bypass this phase if appropriate):

1. Logon to the MVS system with the information provided by the lab instructor(s).

2. Get into SDSF in DA option. Check to see if your z/OS TDS server is running. Your LDAP server's STC name will be provided by the lab instructor.

3. In OMVS, check to be sure that there is a subdirectory in your home directory called **tdslab/tdsna**. This contains several Unix scripts that execute LDAP commands to setup and test your native authentication environment.

4. From OMVS, get into your tdslab/tdsna subdirectory by entering the following Unix command:

cd tdslab/tdsna

5. Edit the file called **tdbmivp** by entering the following command: **oedit tdbmivp** Change '####' to your unsecured port number and save the file by pressing the PF3 key.

6. Run the **tdbmivp** file. This will display data from the TDBM (the DB2 database). If this command gets an error message, please notify your instructor(s).

7. Now we want to repeat the last 2 steps for the SDBM. Edit the file called **sdbmivp**. Change the '%%%%%%%%' to your RACF password. Change the '####' to your unsecured port number. Change '!' to your room letter, Change '??' to your userid number. Save the file. Run the **sdbmivp** file. This will display data from the SDBM (the RACF database). This has optional been set up for your LDAP Server - it is not a requirement for native authentication.

Phase 2 - Update your SLAPD conf file and use the new conf file:

8. We are going to set up your LDAP Server to authenticate a user defined in the TDBM with a RACF password. To start this process, go to SDSF and stop your LDAP Server by entering the following command:

/p <yourLDAPServername>

9. Now the z/OS TDS server must be configured to support native authentication. In ISPF, edit the **DSCONFIG** member of the **<yourHLQ>.TDSTEST.CNTL** dataset. Find first occurrence of 'nativeAuth'. Scroll down to the '#nativeAuthSubtree All' line. First delete the '#' to uncomment the line. Change 'All' to ' "ou=Home Town, o=sharedb2" '. Note the double quotes must be included. This indicates that you want to use native authentication for only part of the z/OS TDS data instead of all the data – only the data whose DN ends with "ou=Home Town,o=sharedb2". Next scroll down the line with '#nativeUpdateAllow off'. Delete the '#' and change the 'off' to 'on' This line indicates that you will allow the users to change their passwords if needed. Lastly scroll down several more lines to the line with '#useNativeAuth off'. Delete the '#' and change 'off' to 'selected'. This indicates that you want to use native authentication but only for selected entries. So the three updated lines in your z/OS TDS server's configuration file should look like:

nativeAuthSubtree "ou=Home Town, o=sharedb2"
nativeUpdateAllowed on
useNativeAuth selected

10. Go into SDSF and restart your LDAP Server by entering the following command:
/s <yourLDAPServername>

11. To check that the z/OS TDS server is running with native authentication support, under SDSF, go into the DA option and select your z/OS TDS server. Find the first occurrence of 'TDBM'. Look down the list of configuration options for your TDBM backend. You should see that nativeAuthSubtree is set to your RDN, that nativeAuthUpdateAllow is set to on, and that useNativeAuth is set to selected. If these are not set as you expected, then you need to review your configuration file again.

Phase 3 - Build some users that use NativeAuth (that is, they use RACF passwords) and some users that do not use NativeAuth (that is, they use passwords stored in the TDBM):

12. Now we are ready to test the native authentication with some LDAP client commands. Go into OMVS and get into your **tdslab/tdsna** subdirectory by entering the following command:
cd tdslab/tdsna

13. To test native authentication we need to set up the users' environment. We are going to use four (4) users for our testing environment. Edit the **nasr1** file. Replace all the '####' to your unsecured port number and save the file.

14. Run the **nasr1** script. This command will search the z/OS TDS server for the current attributes of the 3 users that we will be using to test native authentication. The output of this script will be placed in a file called **user1.outlist**. If you look at this file you will see the DN, TELEPHONENUMBER, USERPASSWORD, TITLE, and POSTALCODE for the following users.

- cn=Henry Nguyen, ou=Home Town, o=Share
- cn=Kyle Nguyen, ou=Home Town, o=Share
- cn=Wayne Nguyen, ou=Home Town, o=Share

Note that none of these users currently have a userPasswordd attribute.

15. Next we will add a userPassword to two of these users. These passwords are not RACF passwords but are passwords stored in the TDBM. They will be used for testing purposes. Edit the **naaddpass** file. Replace all the '####' to your unsecured port number and save the file.

16. Run the **naaddpass** script. This command will modify Henry Nguyen's and Kyle Nguyen's definition by adding a userpassword. It will then search the z/OS TDS server for the new current attributes of the 3 users that we will be using to test native authentication. The output of this script will be placed in a file called **user2.outlist**. Note that now Henry Nguyen and Kyle Nguyen have a userpassword associated with them. Wayne Nguyen does not have a userpassword attribute in the TDBM.

17. Next we will set Henry Nguyen and Wayne Nguyen to use native authentication. To do this, edit the **namod1.ldif** file. Change all the '??' to your userid number. Change all the '!' to your room letter. Save this file and then edit the **namod1** file. Change all the '####' to your unsecured port number. Save this file.

18. Now run the **namod1** script. This command will modify Henry Nguyen's and Wayne Nguyen's definition by adding a new objectclass to their definition called **ibm-nativeAuthentication**. This new objectclass allow the users to

have a new attribute in their definition. The new attribute is **ibm-nativeId**. This script has added both the objectclass and attribute to these two user that are required for native authentication. The output of this script will be placed in a file called **user3.outlist**. Note that now Henry Nguyen and Wayne Nguyen have an attribute called ibm-nativeId and your RACF userid associated with it. Note that Wayne Nguyen does not have a userpassword attribute in the TDBM and Kyle Nguyen does not have a ibm-nativeId.

So a quick review at this time - our test environment has three (3) users - Henry Nguyen, who has a userpassword of 'tomorrow' in the TDBM and who is also using native authentication - Kyle Nguyen, who has a userpassword of 'yesterday' in the TDBM and who is NOT using native authentication Wayne Nguyen who does NOT have a userpassword in the TDBM but who is using native authetnication.

Phase 4 - Test our NativeAuth environment:

19. Now we are ready to test this environment. First edit the **naivp1** file. Change all the '####' to your unsecure port number. Then save the file and run the **naivp1** script. This should produce an authentication error message. The file is doing an lapdsearch under the Henry Nguyen userid. To authentication Henry Hguyen we are using password, 'tomorrow' which is the password from the TDBM. But Henry Nguyen is set up to use native authentication, therefore the authentication is done in RACF.

20. To correct this error, edit the **naivp1** file again. Change the line that has the '-w' parameter. This is the password parameter. Change the password from 'tomorrow' to your RACF password (firstpw). Now save and rerun the file. You should now see several items of data from your z/OS TDS server being displayed. This means that Henry Nguyen was authenticated with the SHAREyxx userid and its associated RACF password.

21. Another example of native authentication is in **naivp2**. Edit the **naivp2** file and change all the '####' to your unsecured port number. Also change the '%%%%%%%%' parameter to your RACF password. Now save and run the **naivp2** file. When you run this ldapsearch, the z/OS TDS server will authenticate Wayne Nguyen with your RACF userid and password - and then display the DNs and postal codes in your LDAP directory.

22. The last test of native authentication, is in the **naivp3** file. Edit the **naivp3** file and change all the '####' to your unsecured port number. Also change the '%%%%%%%%' parameter to your RACF password (firstpw). Now save and run the **naivp3** file. You should get an authentication error message when you run the file. This is because you tried to authenticate Kyle Nguyen with your RACF userid and password. Kyle Nguyen has not been configured to use native authentication.

23. Therefore, to correct our error, edit the **naivp3** file again and change the password parameter (that is, the -w parameter) to 'yesterday'. Save and rerun the **naivp3** file. This will authenticate the Kyle Nguyen with the TDBM userpassword and then display information about Kyle Nguyen.

24. At this point you have got your z/OS TDS server running with native authentication which allows non-RACF userids to be authenticated with RACF passwords.

Hands-on Lab: Optional Lab #3

A lab that accesses the RACF general resources:

Background:

Exercise Instructions (Part 1):

Phase 1 -

Phase 2 –

Phase 3 –

Hands-on Lab: Optional Lab #4

A lab that uses the new CDBM in z/OS 1.11:

Background:

Exercise Instructions (Part 1):

Phase 1 -

Phase 2 –

Phase 3 -

Hands-on Lab: Optional Lab #5

A lab that uses the new password security policy in z/OS 1.11:

Background:

Exercise Instructions (Part 1):

Phase 1 -

Phase 2 –

Phase 3 -