Redbooks Paper

Erich Amrehn
Ulrich Boche
Dr. Manfred Gnirss

Securing Linux for zSeries with a Central z/OS (RACF) LDAP Server

Preface

This IBM Redpaper provides information to help customers, Business Partners, and IBM technical people plan, implement and manage a central security solution for authentication and user information for multiple Linux systems using an LDAP server and RACF on z/OS.

The team that wrote this paper

This paper was produced during a workshop, and subsequent experiments and implementation, at the Technical Marketing Competence Center in the IBM Laboratory in Boeblingen, Germany.

Erich Amrehn is a certified Senior IT Specialist at the EMEA Technical Marketing Competence Center (TMCC), Boeblingen, Germany. Before joining the TMCC, he worked as a project leader at the International Technical Support Organization, Poughkeepsie Center. During that time, he wrote redbooks and taught Linux topics worldwide. Before joining the ITSO in 1998, he worked as a technical consultant to the IBM System/390 division for e-commerce on S/390 in Europe, the Middle East, and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe and worldwide.

Ulrich Boche is an IT security technical consultant in eServer zSeries Technical Support, supporting customers in Germany, Switzerland and Austria in the area of z/OS and e-business security. His areas of expertise include all aspects of security, such as z/OS Security Server (RACF), communications security (SSL and IPsec), LDAP and cryptography support. During numerous ITSO residency projects at the Poughkeepsie, San Jose, and Raleigh centers, he co-authored a large number of IBM Redbooks on security topics.

Manfred Gnirss is an IT specialist at the IBM Technical Marketing Competence Center (TMCC) and the Linux Center of Competence, Boeblingen, Germany. He holds a PhD in theoretical physics from the University of Tuebingen, Germany**.** Before joining the TMCC, he worked in z/VM and z/OS configuration development. Currently he is involved in several customer Linux for zSeries Proof of Concept projects running at the TMCC.

Thanks to the following people for their contributions to this paper:

K. Gdaniec, S. Gybas, T. Hahn, F. Kirschner, E. Puritscher and M. Weisbach.

Introduction

Running a large number of distributed servers involves a great deal of effort to install, administer, maintain, and provide security for them. To reduce the total effort, you can not only consolidate these servers under Linux for zSeries on zSeries hardware—but also benefit from virtualization technologies in order to use the hardware effectively and simplify administration tasks. Furthermore, using a *centralized repository* to handle and maintain user information for multiple Linux systems can reduce both the complexity and effort involved in user administration. In this paper, we describe how you can plan and implement a central security solution for multiple Linux systems and a z/OS.

To run multiple Linux for zSeries servers on one zSeries machine securely, you would use virtualization technology (z/VM and/or LPAR). For more information about the security and isolation of these technologies, refer to the following sites:

```
http://www-3.ibm.com/security/standards/st_evaluations.shtml
http://www.ibm.com/servers/eserver/zseries/library/techpapers/gm130145.html
```

Here you will find the latest security evaluation information on IBM Processor Resource Systems Manager (PR/SM)—which is the technology for proven separation of logical partitions (LPAR)—as well as a discussion of z/VM technology for secure guest isolation in *z/VM Security and Integrity* by Altmark, Alan and Laking, Cliff.

Another aspect of security is to provide secure access to the Linux systems only for authorized users, and the most common way is by using a userid and password for authentication. If there are users for multiple Linux systems to maintain, it's advantageous to keep this sensitive information in a central place, and not spread among various systems.

In a z/OS environment, a place already exists where user information and passwords are kept centrally: the Resource Access Control Facility (RACF) database. This paper describes how to set up an infrastructure for keeping user information and passwords for Linux users centrally on a z/OS system in combination with RACF security.

We show how to set up such an environment by using existing technologies and open source programs to configure Linux servers to cooperate with a z/OS system on which all Linux user information is kept. The technology to achieve this uses Pluggable Authentication Modules (PAMs) and the Name Service Switch (NSS), and configures them to use appropriate Lightweight Directory Access Protocol (LDAP) clients to pass the authentication and user identification requests to an z/OS LDAP server with its backend. (LDAP is an open standard protocol for accessing information services.) The z/OS LDAP server can be configured to use RACF as one backend.

Using a z/OS LDAP server in combination with RACF to keep Linux user information and passwords allows you to make use of existing RACF user definitions for Linux users, and to benefit from the known availability and scalability of z/OS systems for central data storage.

Shared security definitions for user information with LDAP and RACF

In an environment with many Linux for zSeries servers, having a single instance of user information in a central data store (a directory with this information) helps to maintain users from a single management point (add, delete, change user account information, reset of passwords, and so on). If the information is stored in an LDAP directory centrally in a network, Linux for zSeries systems can access this information by using LDAP clients to send messages and requests to the central LDAP server.

PAMs are the standard authentication framework for many Linux distributions. They allow integration of various authentication technologies into Linux system services, such as login, passwd, ssh, ftp, su, rlogin, and so on without changing these services. The modules can be configured to pass authentication requests to LDAP.

Once a user is authenticated in a Linux system, there are many services and applications that need access to user information and resolution of user information, such as resolving a user name out of the uidnumber. This service is provided by NSS (which can also be configured to use LDAP to access information).

By using PAM and NSS, you can configure your Linux for zSeries server to pass user authentication requests (password check) as well as user identification requests (accessing user information) by configuring LDAP clients to a central LDAP server in the network. This helps to avoid storing password information (in */etc/shadow)* as well as user information (in */etc/passwd, /etc/shadow, /etc/group*) in files on the local system.

If you have a z/OS server in your company, it's possible to have the central LDAP server on the z/OS system, and combine and share the existing information of RACF users with Linux accounts, while simultaneously keeping the passwords protected by RACF. This enables you to enjoy the same high quality of service in both your z/OS environment and Linux for zSeries user administration.

For Linux for zSeries servers that run on the same zSeries machine as the z/OS, the necessary communication between LDAP clients and the central z/OS LDAP server may not generate any external network traffic, since technologies like HiperSockets, etc. could be used.

Also, users with access to multiple Linux for zSeries systems can use the same account information on all these systems.

If you have Linux for zSeries systems that do not reside on the same zSeries machine as the z/OS LDAP server (and therefore use an external network connection)—and depending on your environment, needs, and enterprise policies—consider encrypting the communication between the LDAP client and the LDAP server using Secure Socket Layer (SSL), to avoid sending plain text password information over the network!

Recommendation: We recommend that you do *not* store the definitions of administration users like *root* on a central server. Why? So that you won't lose all access to your Linux system in case of a network problem, or when the LDAP server is stopped accidentally. Place all other Linux user definitions in a central LDAP directory.

Authentication with LDAP and RACF

Checking the authentication of a user (with a correct password) is one of the most important parts of establishing a secure computer environment. Passwords are therefore very sensitive

information and must be protected from unauthorized use in an effective way. So instead of storing passwords in a Linux system (usually in a specific file, */etc/shadow*) and performing the authentication check of a user locally, you can perform this check for Linux users in a central z/OS system with RACF by using LDAP technologies.

It's quite easy to enable a Linux system for LDAP authentication checking and set up RACF to be used for password checking via LDAP. RACF can be configured as a backend for the z/OS LDAP server (SDBM), which makes all RACF user information available automatically to LDAP clients while passwords remain protected by RACF. You don't need to add additional data to the LDAP directory for authentication checking; any RACF-defined user, with an OMVS segment in RACF, can log in to a Linux system if the Linux username is equal to the RACF userid.

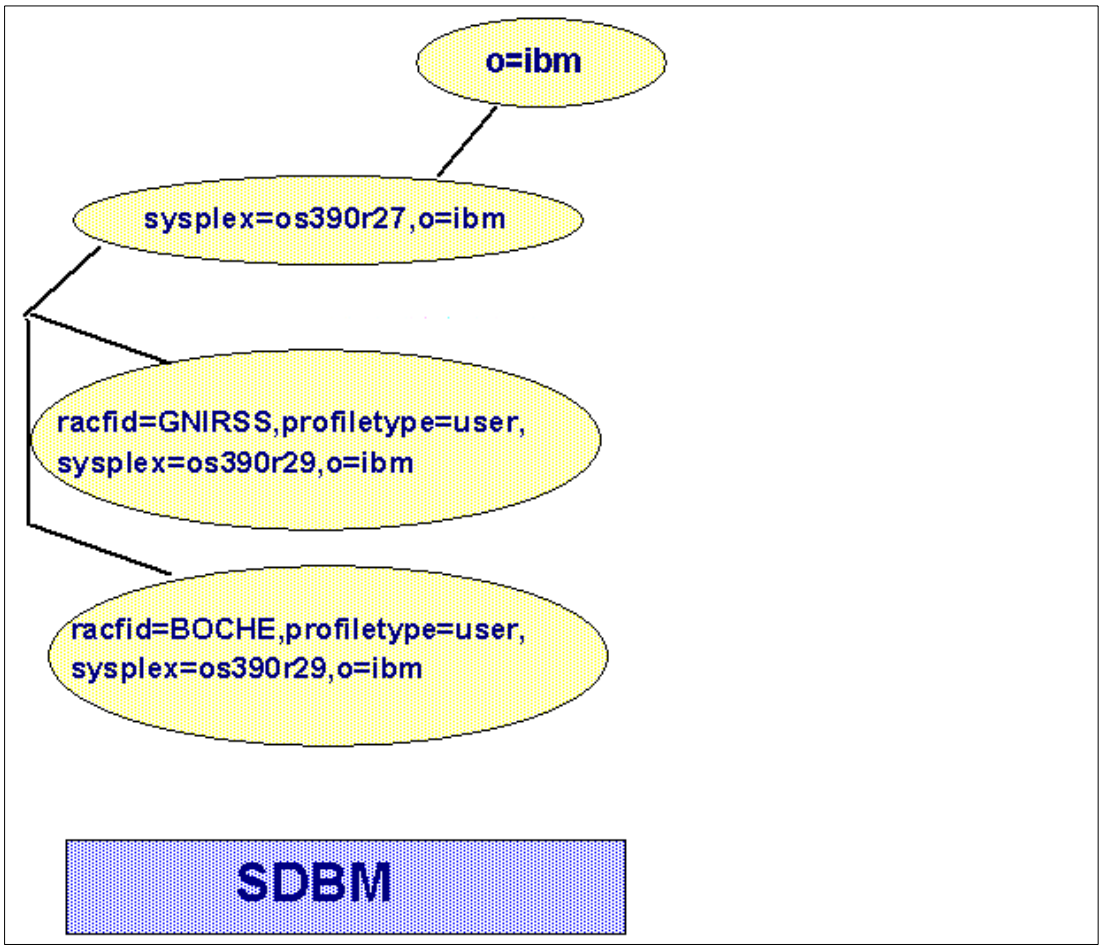


Figure 1 LDAP SDBM tree with RACF user information

With this solution, users always have the same password on all systems where they perform a login. Passwords are kept centrally, and are protected by RACF. On the Linux side, there are no more passwords necessary in */etc/shadow*.

Note: You still have to keep and maintain user information in */etc/passwd* for each of your Linux systems for local user information retrieval (user identification).

Using LDAP does not change the well-known behavior of the RACF database; that is, it's not possible, with any LDAP request, to retrieve the user password from RACF. For this reason, verifying the password via LDAP is performed by connecting with a specific identity to the

LDAP server with the RACF backend. This connection is established with an LDAP bind request.

If it's possible to bind successfully from the Linux system with the specified userid and password to the z/OS LDAP server, the password has been specified correctly. If the bind is rejected, the password was not valid for the specified userid. Figure 1 shows sample RACF information, represented in the SDBM LDAP tree, that is used for authentication checks via LDAP bind requests.

Each Linux service, which should use an authentication check performed by LDAP and RACF, must be configured to pass the check to RACF by a PAM module to an LDAP client. The LDAP client will then forward the request to the LDAP server. Figure 2 shows the elements involved and the information flow for an authentication check for Linux services with RACF via LDAP.

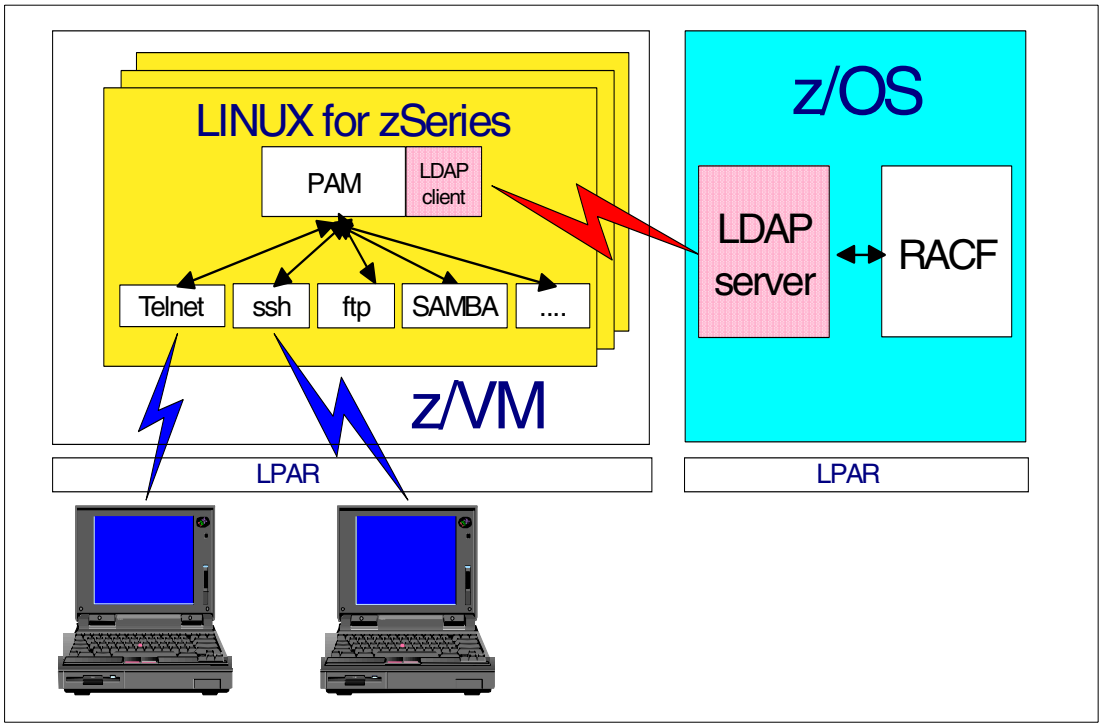


Figure 2 Authentication via LDAP with RACF

To configure Linux to perform the authentication of users with RACF on z/OS, follow these steps:

1. Install the PAM for LDAP package.
2. Configure services to use the PAM LDAP.
3. Configure the PAM LDAP client for authentication.
4. Set up the z/OS LDAP server with SDBM to use RACF for authentication check via LDAP.

In the following sections, we discuss these steps in more detail.

Install the PAM for LDAP package on Linux for zSeries

In order to use the Pluggable Authentication Modules (PAM) for authentication via LDAP, the appropriate PAM LDAP package must be installed on your Linux for zSeries system. Depending on the Linux for zSeries distribution that you're using on your server, the package itself might already be available, although its name might be slightly different (SuSE: *pam_ldap*; Debian: *libpam-ldap;* and so on).

If the package is not part of your distribution, you can download the pam_ldap package from the Internet site of PADL Software Pty Ltd. at:

```
http://www.padl.com/OSS/pam_ldap.html
```

Installation of the package is performed with a simple command. Depending on the Linux distribution, the commands are slightly different. For example:

- SuSE: **rpm -i pam_ldap-your_actual_version.rpm**
- Debian: **apt-get install libpam-ldap**

After the package is installed, you're ready to set up the configuration for PAM authentication via LDAP.

Configure PAM on Linux for zSeries

The configuration of PAM for authentication is very simple. Each service that needs user authentication, must be configured separately, and you need to specify which service should perform authentication with LDAP. In the */etc/pam.d* directory is a configuration file for each service that can use authentication via PAM LDAP. You'll find the files *login* (for telnet access), *ssh*, *ftp*, and so on. These files determine how the authentication behavior during a user access for each of these services is defined.

All PAM configuration files for these services contain information about the following:

- The order of authentication check (*auth* keyword)
- Where to get accounting information (*account* keyword)
- How a password change is performed (*password* keyword)
- Information about the session (*session* keyword)

To provide the desired values for these keywords, you can use any editor to edit these configuration files and modify the *auth, account* and *password* statements for LDAP usage.

Note: It is *not* necessary to modify the session statement for authentication via LDAP.

Telnet

For Telnet access to your system, you might want to insert the keywords *auth*, *account* and *password* with the parameter *pam_ldap.so* at the highlighted positions (see below) in the configuration file */etc/pam.d/login*. Our file contains:

```
auth       requisite  pam_nologin.so
auth       required   pam_env.so
# Authentication via an LDAP server.
auth       sufficient  pam_ldap.so
auth       required   pam_unix.so use_first_pass
# LDAP Server account and session
account    sufficient pam_ldap.so
account    required   pam_unix.so
session    required   pam_unix.so
session    optional   pam_motd.so
session    optional   pam_mail.so standard noenv
password   sufficient pam_ldap.so
password   required   pam_unix.so nullok obscure min=4 max=8  md5
```

The order of the statements in the configuration file is important; therefore, ensure that the following statement is listed *before* the other statements with the *auth* keyword:

```
auth       requisite  pam_nologin.so
```

This statement is used to disallow logins, other than *root* logins, when the file */etc/nologin* exists. The nologin facility can be used during maintenance mode of Linux to prevent logins

from non root users. If you put the *auth* statement with *pam_ldap.so* before the statement with *pam_nologin.so*, then the nologin would be ineffective!

The following statement must be listed *after* the LDAP authentication statement, in order to allow an additional check against the local information stored in */etc/passwd* and */etc/shadow*:

```
auth       required   pam_unix.so use_first_pass
```

This ensures that even when the central LDAP server cannot be used for authentication (for example, because the LDAP server stopped, or because there's a network problem, or because the root user account is not in the LDAP-RACF directory), there's still a way to gain access to the system.

Typically, this is kept for access with the root user. The *use_first_pass* parameter means that for the local authentication check, the same password is used as for the prior LDAP check, without asking the password again.

Note: Do not have a *nullok* parameter in this *auth* statement. A *nullok* would allow passwordless accounts to access the system.

The statements with the keyword *password* are only used when a password is expired and must be changed during login. Ensure that the password statements in this configuration file are identical with those in */etc/pam.d/passwd*. This will ensure the same behavior when a user changes his password manually.

Note: We do not recommend using Telnet as an access method to your Linux system, because data sent via the network is not encrypted (this is also true for userid and password). Instead, consider using ssh (secure shell) to access a Linux system.

Using secure shell (ssh)

The recommended way to access a Linux system with a terminal emulation is by using secure shell (ssh), because communication is done via the network in encrypted form. To configure ssh for authentication check via LDAP when accessing the system, the preceding remarks in "Telnet" on page 7 also apply accordingly to ssh. You might want to insert the keywords *auth*, *account* and *password* with *pam_ldap.so* at the highlighted positions in the file */etc/pam.d/ssh* as shown:

```
auth       required     pam_nologin.so
auth       sufficient   pam_ldap.so
auth       required     pam_unix.so use_first_pass
auth       required     pam_env.so # [1]
account    suf&icient   pam_ldap.so
account    required     pam_unix.so
session    req5ired     pam_unix.so
session    optional     pam_lastlog.so # [1]
session    optional     pam_motd.so # [1]
session    optional     pam_mail.so standard noenv # [1]
session    reqqired     pam_limits.so
password   sufficient   pam_ldap.so
password   required     pam_unix.so
```

ftp, passwd, etc.

As already mentioned, one configuration file has to be adapted for each service which should use authentication via LDAP to RACF. Check all files in */etc/pam.d* and decide whether you want to configure for LDAP RACF authentication—particularly the *passwd* service.

The configuration file *other* is used for services that do not, explicitly, have their own configuration file. This provides you with some nice possibilities to determine the behavior for these services. Decide carefully how you want to handle *sudo*, *su*, and *cron*.

If you also want to authenticate Samba services with LDAP and RACF, you have to install and configure the PAM module for Samba. However, Windows usually sends the user's password in Windows-encrypted format to the Samba server, but RACF uses a different algorithm to encrypt passwords. To enable the implementation of RACF authentication for Samba, disable the Windows encryption.

Configuration of PAM LDAP client for authentication on Linux for zSeries

By using the definitions of PAM in the preceding section, you have configured some (or all) services to perform an authentication check via LDAP before they try to check on the local system. The PAM modules send the authentication requests to the PAM LDAP client.

Now this PAM LDAP client has to be configured to pass the requests using TCP/IP to the LDAP server with the RACF backend on z/OS. In the configuration file of the PAM LDAP client, you need to provide the following information:

- name (IP address) of the host where the LDAP server resides
- distinguished name of the search base
- distinguished name to bind to the server with an effective userid
- password of the binding userid
- LDAP Version
- pam login attribute

When using the RACF backend, it is not possible to bind anonymously to an LDAP server on z/OS. For an authentication check via LDAP, a search for the desired userid in the LDAP tree is done at first. To perform this search, in the LDAP RACF tree, the LDAP client must bind with a userid that has access authority to the LDAP RACF tree and is allowed to perform a search for userids. The authentication check for the userid who wants to access the Linux for zSeries system is done *only* if the search for the userid is successful; this is a bind to the LDAP server with the userid and password.

The name and location of the PAM LDAP client configuration file varies, depending on the Linux distribution (for Debian: *pam_ldap.conf;* for SuSE: */etc/openldap/ldap.conf*). Edit the configuration file to add the necessary information.

Note: The entries for the distinguished name for search base and bind must match the definitions in the LDAP server configuration file on z/OS slapd.conf (see also in Chapter , "Setting up RACF for authentication checking via LDAP" on page 10).

Here is an example for the PAM LDAP configuration file:

```
host os390r27.boeblingen.de.ibm.com
base profiletype=user,sysplex=os390r27,o=ibm
binddn racfid=lxlogin,profiletype=user,sysplex=os390r27,o=ibm
bindpw *******
ldap_version 3
pam_login_attribute racfid
```

This file contains the password of the bind user *lxlogin*, who binds to the LDAP server and performs a search for the username (racfid) in the SDBM (RACF) database of the user to be authenticated. Therefore, we strongly recommend that you limit the authority of this *lxlogin* user on the z/OS side to a minimum. This user needs only to bind to the LDAP server and to

perform a search (this corresponds to the commands RACF LISTUSER and SEARCH CLASS USER).

Note that this bind user does *not* need to be the ldap administrator userid; a userid with access to IRR.LISTUSER in class FACILITY or the AUDITOR attribute should be sufficient. (For example, this user does not have to be capable of performing a TSO logon.) In addition, it is recommended that you protect this configuration file from read access by other Linux users.

Because authentication of users via LDAP to the RACF backend (SDBM) of the z/OS LDAP server requires a non-anonymous bind to search for the username, there must be a userid and a password specified in the PAM LDAP configuration file. To avoid having any userid and password stored in this file, consider using the DB2 backend (TDBM) and the Native Authentication method instead, as described in Chapter , “Native authentication” on page 14.

Setting up RACF for authentication checking via LDAP

On the z/OS side, there’s not much you have to do to allow RACF authentication via LDAP; just bring up the LDAP server with RACF as the backend (i.e, use SDBM). For detailed instructions on this task, refer to *z/OS Security Server LDAP Server Administration and Use*, SC24-5923-02.

Here is an extract of the configuration file of the LDAP server *slapd.conf* in which the highlighted lines indicate our adaptations:

```
# ----------------------------------------
# adminDN <distinguished name>
#   This option specifies the distinguished name (DN) of the
#   administrator for this LDAP server.
# ----------------------------------------
adminDN "racfid=LDAPADM,profiletype=user,sysplex=os390r27,o=ibm"
# ----------------------------------------
# adminPW <password>
#   This option specifies the password for the administrator (adminDN)
#   for this server.
#    (It is not necessary to provide here a password)
# ----------------------------------------
#adminPW "%ADMINPW%"
# ------------------------------------------------------------
#  SDBM-specific CONFIGURATION SETTINGS
# ------------------------------------------------------------
database sdbm GLDBSDBM
# ----------------------------------------
#   This option specifies the suffix for the SDBM backend
# ----------------------------------------
suffix "sysplex=os390r27,o=ibm"
```

Note that the SDBM suffix is used in the PAM LDAP configuration file as the search base for RACF authentication.

User identification with LDAP and DB2 backend - instead of using passwd file

By using the method presented in the previous part of this paper, we can avoid storing any user passwords in a Linux system. If a user is authenticated successfully, there are additional services and applications which still need user information (like retrieving the name of the home directory, or resolution of *uid* to *username*). This information is usually kept in plain text files (such as */etc/passwd, /etc/shadow*, and */etc/group*), but can be provided by name services, too.

To avoid having to store user information on each individual Linux system in */etc/passwd* and */etc/shadow*—and to avoid having to maintain user information multiple times—you can store this information in a central LDAP directory. This central directory can then be accessed by LDAP clients from the Linux system, and the information can be provided through Name Service Switch (NSS). NSS is a service of the GNU C Library, and can be configured to retrieve data from an LDAP server and provide it for the desired purposes.

Because of its fixed schema and the limited amount of user information stored in RACF, the SDBM backend of the LDAP server is not well suited as a user registry for Linux.

If you want to store and access the Linux user information centrally from the z/OS LDAP server, you cannot use the SDBM backend. Instead, you must use the DB2 (TDBM) backend. TDBM is flexible enough to fulfill all requirements.

All necessary data for authentication and user information retrieval (NSS) of Linux users can be kept in a central z/OS LDAP server, but in different backends (SDBM and TDBM) simultaneously, as shown in Figure 3.

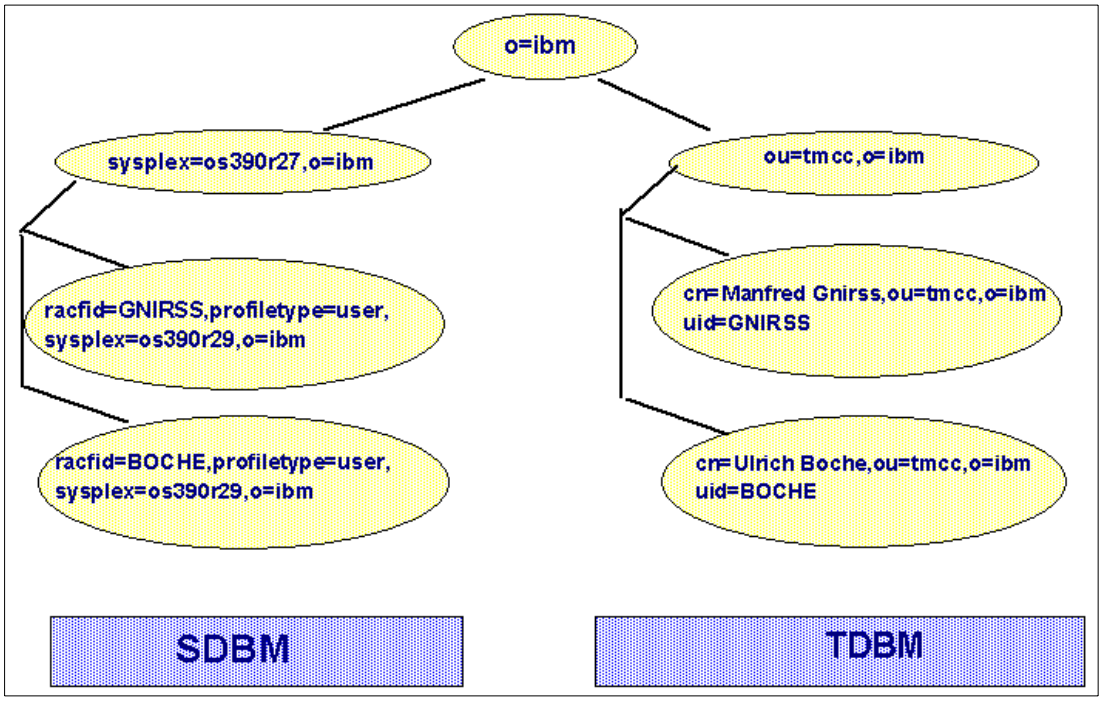


Figure 3 LDAP trees for authentication (SDBM) and identification (TDBM)

Authentication requests or NSS requests sent by the Linux system to the z/OS LDAP server are parsed by the LDAP protocol handler and sent automatically to the appropriate backend, as illustrated in Figure 4 on page 12.

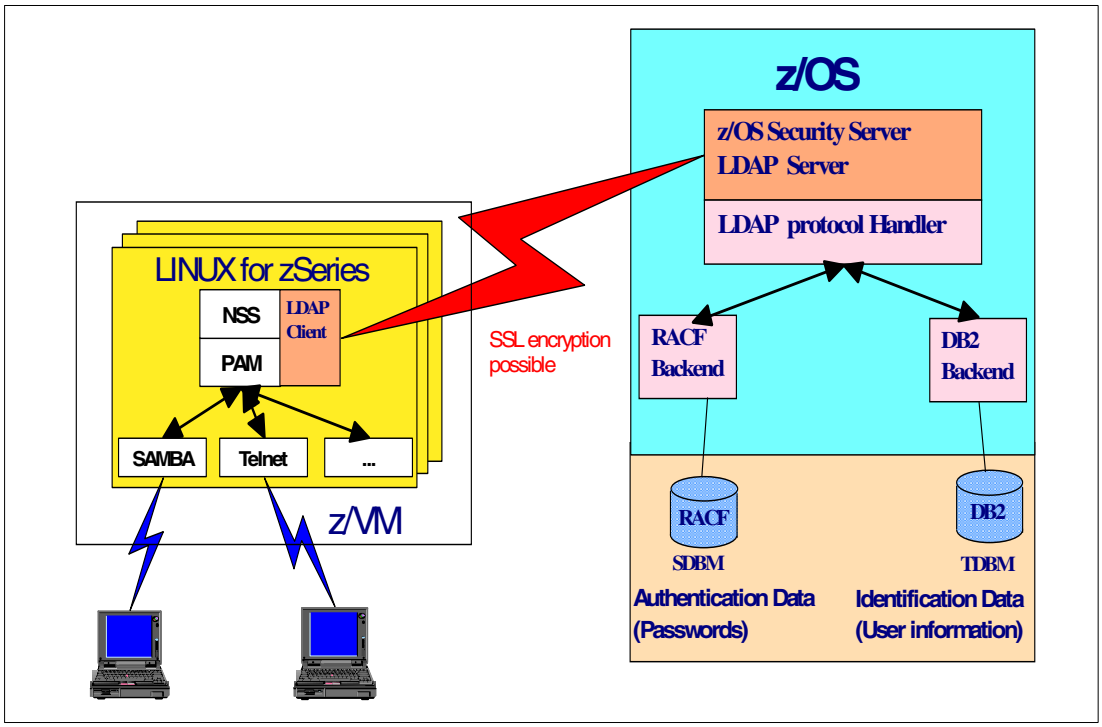


Figure 4 Authentication and identification on one z/OS LDAP server

To configure Linux to use NSS with LDAP to access information from a central z/OS LDAP server, follow these steps:

1. Install the NSS PAM package.
2. Configure the LDAP client to use NSS.
3. Edit the NSS configuration file.
4. Provide user information in the TDBM backend.

Install and configure Name Service Switch (NSS) on Linux for zSeries

To use the NSS via LDAP, the appropriate package must be installed on your Linux for zSeries system. Depending on the Linux distribution you're using, the package itself might already be available, although its name might be slightly different (SuSE: *nss_ldap*; Debian: *libnss-ldap*, and so on).

If the package is not part of your distribution, then download it from the Internet. Installation of the package is performed with a simple command. Depending on the Linux distribution, the commands are slightly different. For example:

SuSE: `rpm -i nss_ldap-your_actual_version.rpm`

Debian: `apt-get install libnss-ldap`

After the package is installed, you're ready to adapt the configuration for NSS.

Configure the LDAP client on Linux for zSeries to use NSS

The configuration of the LDAP client for NSS is very simple. The LDAP configuration file (Debian: */etc/libnss-ldap.conf*; SuSE: */etc/nss_ldap.conf*, and so on) must contain the following information:

- Name (IP address) of the host where the LDAP server resides

- Distinguished name of the search base

Because the information for identification is stored in the DB2 backend of the LDAP server on z/OS and not within the RACF backend, an anonymous bind to the LDAP server is possible. Therefore, there's no userid and password to be provided in the LDAP client for using NSS.

Note: This is a different configuration file of the LDAP client. It is not identical with the PAM LDAP; this one is used for NSS only.

Following is an extract of the config file:

```
#...
# Your LDAP server. Must be resolvable without using LDAP.
host os390r27.boeblingen.de.ibm.com
# The distinguished name of the search base.
base o=ibm
# ...
# The LDAP version to use (defaults to 3
# if supported by client library)
ldap_version 3
# ...
```

Note: The distinguished name of the search base here is different from the search base of the SDBM backend.

Configure NSS

The Name Service Switch functionality file */etc/nsswitch.conf* has to be modified. The *ldap* keyword must be inserted at the highlighted positions indicated below. The file should look as follows:

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.
passwd:         ldap compat
group:          ldap compat
shadow:         ldap compat
hosts:          files dns
. . .
```

Note: After any modification of the Name Service Switch functionality file, you should stop and restart a running Name Service Caching Daemon (nscd).

Provide user information via the LDAP DB2 backend on z/OS

In order to be able to store all the user information that's necessary for Linux user identification in the TDBM backend of the z/OS LDAP server, you have to ensure that all relevant information for a *posixAccount* (in addition to the object classes *person* and *account*) can be kept in the LDAP directory tree. The object class *posixAccount* is contained in the *nis* schema, which you may have to install first.

Then provide all information that is relevant for the object classes *person* and *posixAccount*. Following is the minimum information to be provided:

```
cn: common name
sn: surname (last name)
uid: Linux username (userid)
uidnumber: numeric UID
gidnumber: numeric GID of user' group
```

```
home directory: home directory for shell
login shell: login shell for the user
```

Note: The *uid* attribute must be unique in the LDAP directory subtree, as it will be used as a search key for NSS. For information on how to install the necessary *nis* schema and how to provide the data for the user entries, refer to Chapter , “Schema for posixAccount (used for identification)” on page 17.

Storing user information in a LDAP directory means that maintenance of user information can be done using LDAP. This can considerably reduce the administration effort.

Test your configuration

During testing of identification via NSS LDAP, you should stop the Name Service Caching Daemon (**`/etc/init.d/nscd stop`**) in order to prevent unexpected results when changing data/content in the LDAP directory for test purposes. (Typically you would need changes in the LDAP directory to be reflected immediately on the Linux for zSeries side. Caching on the Linux side would prevent getting the changed data for the test.)

After successful testing, remember to start the Name Service Caching daemon again (**`/etc/init.d/nscd start`**), because it is really needed in a production Linux environment.

Native authentication

In previous sections, we saw how the combination of authentication and identification of a Linux for zSeries user by using LDAP technologies allows you to avoid having any user information stored on the local Linux for zSeries system. In the solution described in the previous section, we had to use two different backends on the z/OS LDAP server to benefit from the RACF protection of passwords.

But is there a way to use only *one* backend for both purposes? Previously, it was possible to use only one data store in the LDAP server for both purposes (this could also have been done with other LDAP servers on any platform), but without the advantage of RACF protection.

Starting with OS/390 2.10 and the PTF for native authentication (SPE APAR OW47596), it is now possible to have both advantages: native authentication allows you to store user information centrally and simultaneously protect passwords with the proven quality of RACF by using only one backend of the z/OS LDAP server (namely, the TDBM).

When using native authentication, all LDAP requests are sent to the TDBM backend. Retrieval of user information is handled by TDBM directly, and password verification requests are sent automatically to RACF (the __passwd() service is used) under the covers, as the password information is not kept in TDBM —but this is transparent to the LDAP client.

Figure 6 on page 16 shows the information flow when a Linux for zSeries user is authenticated with native authentication.

For existing user accounts in RACF, you have only one additional entity of user information for any number of Linux systems. For a Linux user that is not already a RACF user, you need to store only a small amount of additional information in RACF to benefit from RACF password protection.

Note: While not necessary, it does no harm to have the SDBM backend defined in a z/OS LDAP server that uses native authentication. Only the directory tree in the TDBM backend of the LDAP server must be accessible to the Linux servers (see Figure 5 on page 15.)

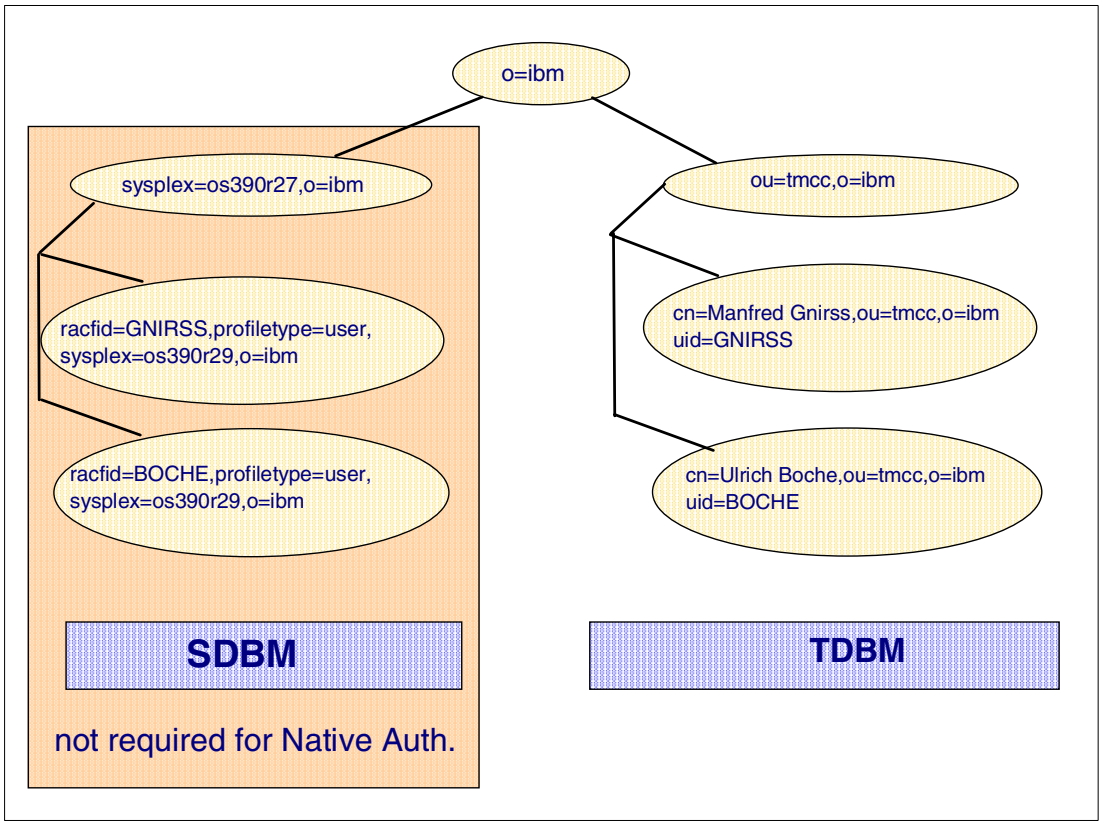


Figure 5 Native authentication and identification using TDBM

For native authentication, you need to set up both sides: Linux for zSeries and z/OS. On the Linux side, you need the setup for the authentication check via LDAP and for retrieving user identification using NSS and LDAP.

As for native authentication, all information is accessed via TDBM. This means that you can use the same LDAP search base for authentication and NSS configuration. It also means that, on the Linux side, you don't need to care about which backend of the LDAP server on z/OS side is used.

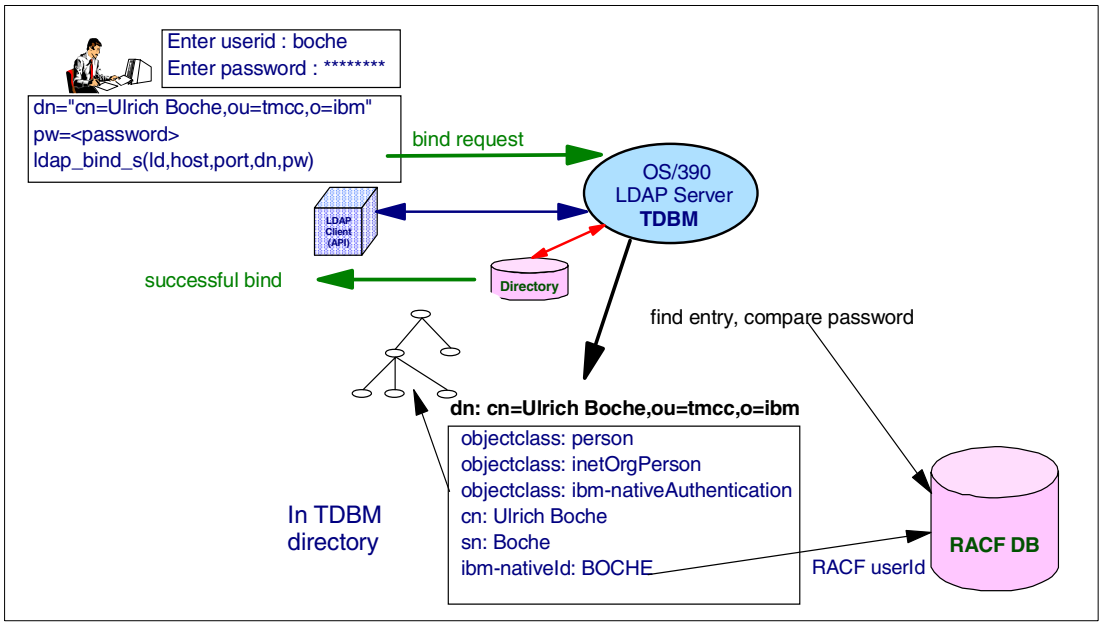


Figure 6 Information flow and native authentication

To use native authentication, you simply have to set up LDAP authentication and NSS on the Linux side. On the z/OS side, you need to have TDBM as the backend for the LDAP server. You also need to configure the LDAP server for native authentication, and provide the user information in the LDAP directory.

The setup requires the following steps:

1. Install and configure the PAM LDAP and NSS.
2. Set up the z/OS LDAP server with TDBM and native authentication.
3. Provide the necessary user information in TDBM.

Install and configure PAM and NSS on Linux for zSeries

If you have already installed PAM LDAP and NSS for authentication and identification as described in the preceding sections, then you only need to adapt the configuration file of PAM LDAP—change the distinguished name of the search base to the base of the TDBM LDAP tree. Also remove the two entries regarding the bind of the “technical” user to search in the SDBM backend (*binddn* and *bindpw*) and adapt the *pam_login_attribute*. See the example below.

If you do *not* have PAM LDAP and NSS already installed and configured, then perform these tasks as described in the preceding sections. Installation and configuration for native authentication on the Linux for zSeries system is nearly identical to what we have already described. The only difference is in the configuration of the PAM LDAP client for authentication.

Authentication and identification are both handled by the DB2 backend of the LDAP server on z/OS. Therefore, you’ll have the same value for the search base as in the configuration file of the LDAP client for the NSS services.

Furthermore, because authentication is now no longer performed by LDAP with SDBM RACF as the backend (which always requires a non-anonymous bind), it is now possible to omit any userid and password in the configuration file of the LDAP client on Linux for zSeries side. The *pam_login_attribute* also has to be set to *uid* (not to *racfid*).

Note that the *base* entry in the configuration files for both PAM LDAP and NSS are now identical (in the case without native authentication, we had the SDBM backend as the search base (*base sysplex=os390r27,o=ibm*) in the PAM LDAP configuration file). Following is an extract of the PAM LDAP configuration file:

```
host os390r27.boeblingen.de.ibm.com
base o=ibm
#The user ID attribute (defaults to uid)
#pam_login_attribute uid
```

Note: Instead of the attribute *uid*, the attribute *ibm-nativeId* could be used.

Here an extract of the LDAP configuration file for NSS:

```
host os390r27.boeblingen.de.ibm.com
base o=ibm
```

Set up the z/OS LDAP server

Refer to *z/OS Security Server LDAP Server Administration and Use*, SC24-5923, for information on configuring and setting up the LDAP server on z/OS with DB2 (TDBM) for native authentication, starting with the release of z/OS 1.2 documentation. If you're running a prior release, you can also use the z/OS 1.2 documentation, or you can check the PTF description (SPE APAR OW47596). In our implementation, we used the RACF userid *ldapadm* to administer the LDAP server.

Configuration file slapd.conf

In the TDBM-specific configuration section, the suffix is different from the suffix for SDBM (remember, for SDBM we specified *sysplex=os390r27,o=ibm*). The TDBM suffix looks as follows:

suffix ="o=ibm"

In addition, we used the following parameters:

```
nativeUpdateAllowed on
nativeAuthSubtree all
useNativeAuth all
```

This enables all users who have an entry in the LDAP tree, with either the *ibm-nativeId* attribute or the *uid* attribute, to bind natively to the LDAP server (i.e, the password is verified by RACF) and to allow a native password modify (i.e, change the password in the Security Server through the TDBM backend).

The parameter *useNativeAuth* determines which entries are participating in native authentication. The value *all* means that the attributes *uid* and *ibm-nativeId* in the person entry will trigger RACF authentication. Specifying *selected* as value means that the attribute *ibm-nativeId* triggers the RACF authentication (you should use *selected* only if a subset of Linux users should participate in RACF authentication).

Schema for native authentication

To perform native authentication, you need a schema in the LDAP server that will allow you to specify an attribute for user entries to participate in native authentication. Therefore, you have to add the schema found in */usr/lpp/ldap/etc/NativeAuthentication.ldif* on z/OS.

Schema for posixAccount (used for identification)

Furthermore, if you want to keep all user information needed by a Linux system in the LDAP directory, you will need an additional schema to provide information like the *uid*, *home directory, gidnumber*, *uidnumber*, etc., for a Linux user. This information is part of the *posixAccount* object class, and is available in *nis.schema.2.* Unfortunately, this schema is not

part of the z/OS LDAP server shipment. You may request a copy from your IBM representative, or download it from the ITSO Web site:

```
ftp://www.redbooks.ibm.com/redbooks/REDP0221
```

Schema for posixGroup (used for identification)

If you want to keep group membership information (as well as user information) in the LDAP directory instead of having it on each individual Linux server in the */etc/group* file, you also need the *posixGroup* object class. The *posixGroup* is used to contain the following information: *cn, gidNumber* and, optionally, *userPassword, memberUid* and *description.* The definitions of the *posixGroup* are also found in *nis.schema.2*.

Updating and adding the schema

Before the schema for native authentication, *posixAccount*, and *posixGroup* can be used, you have to adapt the schema files to your LDAP server settings. The parameter *<suffix>* in one of the first lines must be replaced with the suffix of the TDBM backend. (In our implementation, we changed *dn:cn=schema, <suffix>* to *dn:cn=schema, o=ibm.*)

Install the two schema files *(nativeauth.ldif* and *nis.schema.2.ldif)*, each with the following command, where the ldif (LDAP Data Interchange Format) file named *the_filename.ldif* contains the adapted schema:

```
ldapmodify -h 127.0.0.1 -D racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w
the_password -f /the_path/the_filename.ldif
```

Miscellaneous

The LDAP server needs the following z/OS libraries to be program-controlled:

- <DB2HQ>.SDSNLOAD
- <DB2HQ>.SDSNEXIT (This is not mentioned in the documentation.)

Organization of the LDAP directory tree

Before you store user information and group information in your LDAP directory, you should carefully plan the structure of your directory tree. How you structure your data depends on your organization and your business needs. A good structure simplifies the usage of the information and determines how flexible your structure will be when you need to adapt it to new requirements.

Typically, the hierarchy in the LDAP directory is set up in a way that corresponds to the organizational structure of your company. You would not place your user information directly under the root of your LDAP tree, but rather below some organizational unit.

In this paper we use only one level of organizational unit in order to keep our examples small and simple. However, for real-life scenarios you would include more levels to get a more structured directory. For example, within an organizational unit, it's very common to distinguish between persons and resources; therefore, person entries are typically kept below an additional *people* level.

If not already available, you need a suffix for your directory. A suffix specifies the distinguished name for the root of a directory in the database. It is the highest entry stored in the directory by an LDAP server. Further, you can add an organizational unit into your LDAP directory tree. It is conventional to have just one "organization" level but multiple nested levels of "organizational unit". Both suffix and organizational unit can be added with an **ldapadd** command in combination with a lidf file; see the following example:

```
ldapadd -h os390r27.boeblingen.de.ibm.com -D
racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w the_password -x -f
/the_path/datafile.ldif
```

To add the suffix, *datafile.ldif* contains:

```
dn: o=ibm
objectclass: top
objectclass: organization
ou: ibm
```

To add an organizational unit, *datafile.ldif* contains:

```
dn: ou=tmcc,o=ibm
objectclass: top
objectclass: organizationalUnit
ou: tmcc
```

Provide user information via an LDAP DB2 backend

In order for each user to participate with native authentication, and to keep all the necessary user information of a Linux user, you need an entry in the LDAP directory with the necessary information of a POSIX account—including a *uid* or *ibm-nativeId* attribute with the value of the RACF userid of the user. It is evident that you can store much more information in a user entry in the LDAP directory than is required for authentication and identification purposes.

For example, the user information can be added to the LDAP directory by using the **ldapadd** command, as follows:

```
ldapadd -h os390r27.boeblingen.de.ibm.com -D
racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w the_password -x -f
/the_path/addusr.ldif
```

in combination with an ldif file *adduser.ldif*, which looks as follows:

```
dn: cn=Manfred Gnirss, ou=tmcc, o=ibm
roomnumber: 14-039
givenname: Manfred
objectclass: top
objectclass: person
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: posixAccount
uid: gnirss
cn: Manfred Gnirss
sn: Gnirss
uidnumber: 10020211
gidnumber: 1113
telephonenumber: 120-4093
street: Schoenaicher Str. 220
homedirectory: /home/gnirss
loginshell: /bin/bash
mail: gnirss@de.ibm.com
departmentnumber: 3300
employeenumber: 0994711
l: Boeblingen
postalcode: 71032
postaladdress: Schoenaicher Str. 220
```

Note: Depending on the security policy of your company, you should consider protecting all or part of the information stored in an entry that is not necessary for the NSS (uidnumber, gidnumber, uid, cn, homedirectory, loginshell) from general read access.

Provide group information via an LDAP DB2 backend

Before you can store group information in the LDAP directory, you have to establish a group structure. This can be done with an **ldapadd** command in combination with an ldif file, as shown:

```
ldapadd -h os390r27.boeblingen.de.ibm.com -D
racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w the_password -x -f
/the_path/grp.ldif
```

where the file *grp.ldif* contains:

```
dn: ou=Group,ou=tmcc,o=ibm
objectclass: top
objectclass: organizationalUnit
ou: Group
```

After the group structure has been added, you can provide the individual group information by using the **ldapadd** command and an appropriate ldif file, as shown:

```
ldapadd -h os390r27.boeblingen.de.ibm.com -D
racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w the_password -x -f
/the_path/pgrp.ldif
```

where the file pgrp.ldif contains:

```
dn: cn=tmccusr,ou=Group,ou=tmcc,o=ibm
objectclass: posixGroup
objectclass: top
cn: tmccusr
gidNumber: 1005
```

Native authentication: changing passwords

In an environment where one userid is used on multiple systems, it is important to be able to change the value of the password from any of the involved systems. In the environment we describe, the password check is done via LDAP on the central server. Therefore, we also look for a method to change passwords via LDAP.

To allow native password changes (for example, changing the RACF password of a user via LDAP through the TDBM), you will need to specify the following in the LDAP server configuration file:

```
nativeUpdateAllowed YES
```

You also need a specific Access Control List (ACL) to allow users to change their own passwords, because with the default ACL, only the LDAP administrator can update any fields in the LDAP directory (TDBM). Therefore, in our environment we added an ACL with the **ldapmodify** command on the z/OS system:

```
ldapmodify -h 127.0.0.1 -D racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm -w
the_password -x -f /the_path/newACL.ldif
```

where the file *newACL.ldif* contains:

```
ou=tmcc, o=ibm
aclpropagate=TRUE
aclentry=cn=this:critical:w
aclentry=cn=anybody:NORMAL:RSC:SYSTEM:RSC
ownerpropagate=TRUE
entryowner=access-id:RACFID=LDAPADM,PROFILETYPE=USER,SYSPLEX=OS390R27,o=ibm
aclsource=ou=tmcc, o=ibm
ownersource=default
```

With this ACL, the current user (indicated by the keyword *this*) is allowed to change critical fields. Since *userPassword* is the only critical field in a *person* entry, this ACL allows users to change their own passwords (actually, with the setup for native authentication, the password is changed in RACF).

Having this ACL active enables you now to change passwords natively using the **ldapmodify** command. The following example shows how we tested a password change for user Manfred Gnirss:

```
ldapmodify -h os390r27.boeblingen.de.ibm.com -D "cn=Manfred Gnirss,ou=tmcc,o=ibm" -w
myoldpasswd -x -f /the_path/pwmodify.ldif
```

The content of the file pwmodify.ldif:

```
dn: cn=Manfred Gnirss,ou=tmcc,o=ibm
changetype: modify
delete: userPassword
userPassword: myoldpasswd
-
add: userPassword
userPassword: mynewpasswd
-
```

Typically, Linux users do not change passwords with an LDAP command; instead, they either use the Linux **passwd** command, or change passwords during a login request. Today, the existing services within pam_ldap do not support password changes in the way that is required by z/OS LDAP with native authentication. Native password changes are to be made with **ldapmodify** to perform a *delete* followed by an *add* of the *userPassword* attribute, but this is not coded in the existing pam_ldap modules. This means that, by default, native password changes cannot be triggered by an existing parameter for the *passwd* statement in the pam_ldap configuration file.

You can fix this relatively quickly and easily: A slight modification in the pam_ldap module allows you to enhance it to the desired behavior. You need only to edit the source code of the pam_ldap module, add or modify a few lines, and recompile it. A proposal for a modification to create a new parameter for the *session* keyword to invoke a password change with the required format is shown in the appendix.

After adapting the pam_ldap module, you can include the appropriate *session* statement with the new parameter in the pam_ldap configuration file. Now a Linux user can change his password during login or using the **passwd** command.

Creation of a home directory on Linux for zSeries

With native authentication it is possible to keep all the user information in a central LDAP directory. It is not necessary to keep user information locally on the Linux system (*/etc/passwd* and */etc/shadow*) and to maintain it locally. There is one exception: Typically, users have their own home directory. If a home directory does not exist for a user, a message similar to the following is issued after a successful login:

```
login as: gnirss
Sent username "gnirss"
gnirss@donald18.boeblingen.de.ibm.com's password:
Linux donald18 2.4.17 #1 SMP Thu May 2 13:21:20 CEST 2002 . . .
Could not chdir to home directory /home/gnirss: No such file or directory
gnirss@donald18:/$
```

When the user administration is done via LDAP, you want also to avoid the administrator task for Linux to manually create a home directory for a user on each system on which that user is allowed to log in. There are two possibilities:

- The home directory of a user can be located on a network file server (e.g., access via NFS)
- The home directory is created automatically when a user performs the first login.

To enable the Linux system to create a home directory for a user when logging in for the first time, you need a *session* statement such as:

```
session required pam_mkhomedir.so skel0/etc/skel/ umask=0077
```

In the PAM configuration file for each service (Telnet: */etc/pam.d/login,* ssh: */etc/pam.d/ssh, etc.*), which should eventually trigger the creation of the home directory. The home directory is created with a name for the user as found in the LDAP directory via the NSS services if it does not exist. *umask=0077* creates directory permission 700. Use *umask=0027* for a group-readable directory (750). To successfully create the home directory, the NSS services via LDAP must be enabled (see “Setting up the LDAP client on Linux for zSeries for SSL with server authentication” on page 27) and the *session* statement

```
session required pam_unix.so
```

must also be found in the PAM configuration files, as *pam_unix.so* uses the NSS service under the covers.

With this setup, the home directory will be created with the first successful login of a user:

```
login as: gnirss
Sent username "gnirss"
gnirss@donald18.boeblingen.de.ibm.com's password:
Linux donald18 2.4.17 #1 SMP Thu May 2 13:21:20 CEST 2002 . . .
Creating home directory '/home/gnirss'.
gnirss@donald18:~$
```

Note: The *pam_mkhomedir* function is available starting with PAM version 0.71.

Attention: Starting with OpenSSH level 3.3, the automatic creation of a home directory during the first login with ssh no longer works due to a recent security-related change in ssh. If you still want to keep *pam_mkhomedir*, you can disable Privilege Separation in the file /etc/ssh/sshd_config by using the statement:

```
UsePrivilegeSeparation no
```

User administration

As some user information is duplicated in the RACF and DB2 backends, you have to ensure that this data is kept synchronized. You should consider getting some way to make sure that the overlapping data (*userid, name* of a user, etc.) is always updated in the RACF data base and in the DB2 backend with identical values. This might be done with some directives for the user administration process or by some locally written administration utility.

Initial setup in LDAP DB2 backend with RACF data

For users who should get access to Linux systems and who are already z/OS users with user information stored in a RACF database, it is probably effective to populate the TDBM backend with that existing information and complete it with the information necessary for a Linux account.

Recommendation:

1. Unload the RACF data base with IRRDBU00.
2. Extract all desired data from the flat file with a locally written REXX program.
3. Create an ldif file.
4. Insert the data into TDBM with LDAP.

   To do this (depending on the size of the file), use either the **ldapmodify** command or, for mass insertion, the ldif2tdbm utility.

If you set up the SDBM backend for the z/OS LDAP server, then there is also another way to initially fill the TDBM with RACF data: If you have a program available that can read the contents of an LDAP directory and can create an ldif file out of this information, you can use it to extract all the data out of RACF via the SDBM backend (using the suffix of the SDBM). The generated ldif file can then be examined and the entries that are not to be put into the TDBM can be removed. Then complete the remaining entries with the necessary information and import the modified ldif file into DB2 (using the suffix of the TDBM).

To be able to use the entries in the TDBM for Linux authentication and identification, the entries must contain at least the information for a POSIX user (*uidnumber, gidnumber, username, loginshell*, and *home directory*).

Note: For Linux users you should ensure that uid numbers are unique.

Changing user data

After you have set up and populated the TDBM with user data, you have to keep some information synchronized between RACF and TDBM when adding, deleting or updating user entries. If you also set up SDBM, you can do this simply by sending an **ldapmodify** request for the appropriate entry with specific information to the SDBM and to the TDBM backend. It is relatively simple to write a small program for this purpose.

Maintaining user data for users not known to RACF

It is not necessary for all users who need to have access to a Linux system to be defined in RACF. If for any reason a specific user or user group is not to be included in RACF (that is, no entry for them in RACF and their passwords are not to be protected by RACF), you can also keep the user information in the LDAP directory and store the password belonging to the entry directly in the LDAP TDBM backend.

To enable a mixture of user entries within an LDAP directory tree of users participating in native authentication and those not participating, you can specify *useNativeAuth selected* in the LDAP configuration file slapd.conf and set the *ibm-nativeId* attribute to the value of the *uid* attribute for the entries that should use native authentication. An example for this setup is shown in Figure 7 on page 24.

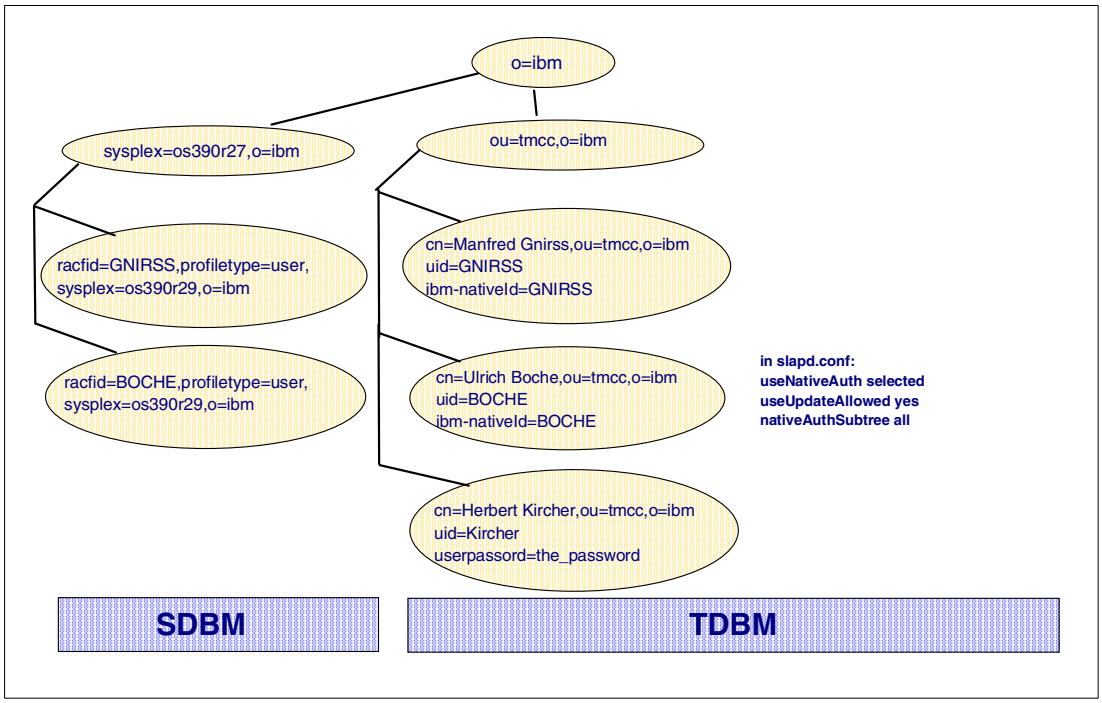


Figure 7 Mixed authentication methods within one subtree

Another way is to use a separate subtree for entries without native authentication.

If you have at least one LDAP directory subtree that should not participate in native authentication, then you have to mention explicitly those subtrees in the configuration file that *will* participate in native authentication. For example, if *ou=tmcc, o=ibm* is the subtree for users with a RACF password, and *ou=boe*, *o=ibm* is the subtree for non-RACF users, then you could have the following settings in *slapd.conf* for the LDAP server:

```
usenativeAuth all
nativeUpdateAllowed yes
nativeAuthSubtree ou=tmcc,o=ibm
```

An example of how data can be organized for this situation is shown in Figure 8 on page 25.

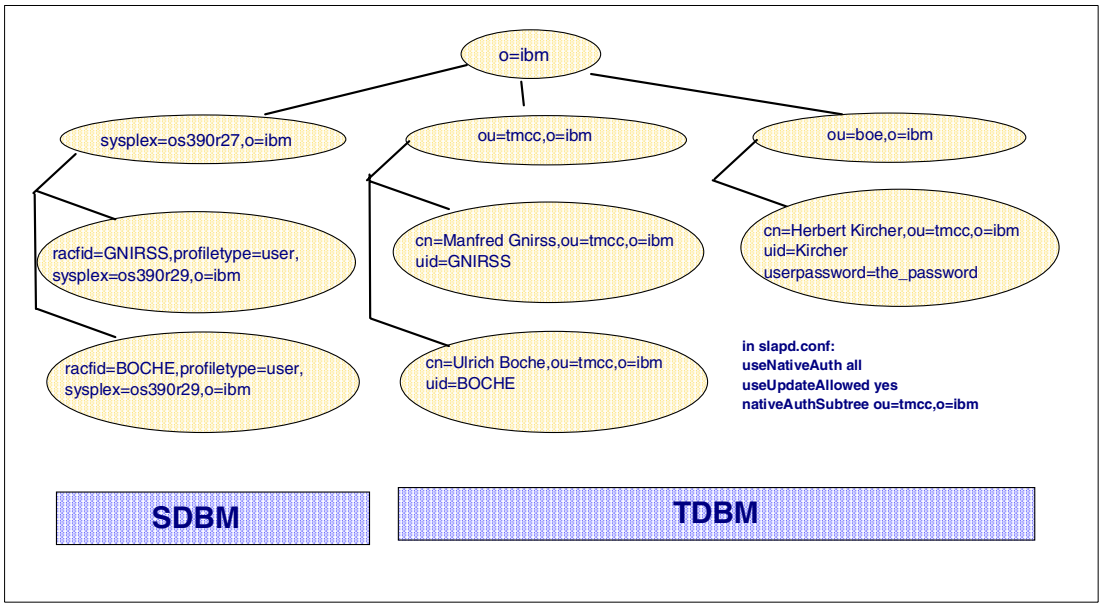


Figure 8 Mixed authentication methods in separate subtrees

Which way, or which combination of ways, will be used depends on your needs and how you estimate the effort to set up your complete directory: Starting from a huge RACF data base and adding only a few users that are not using native authentication might lead to a setup with a separate subtree and not specifying *ibm-nativeId* attributes. If you start from an existing corporate directory and adapt it for a few RACF users to use native authentication, it might be less effort to add the additional required *ibm-nativeId* attributes with the value of the RACF userids (respectively, the *uid* value) to the existing TDBM directory entries.

The LDAP server makes it possible to prevent unauthorized access to the user passwords that are stored in the TDBM backend. User entries in a subtree which are not participating in native authentication, need a *userPassword* attribute. The values of this attribute (i.e., the stored passwords) can be encrypted when stored in the directory, which prevents clear text passwords from being accessed by any user, including the system administrator. For details on how to set up the LDAP server to store encrypted passwords, see the chapter entitled “Configuring for user password encryption” in *z/OS Security Server LDAP Server Administration and Use,* SC24-5923.

Central management of different access rights for multiple Linux servers

What we discussed up to now leads to an environment in which we have identical security information for all Linux servers. The case where users should have different access authorities on different servers can also be handled and administered in the central LDAP data base.

In the user entries in the LDAP tree, add an attribute *host* (this is already included in the objectclass *account* of the schema defined in *schema.user.ldif*) and specify as attribute values the names of the hosts to which a user is allowed to have access.

Note: Wild cards, as well as multiple values for the attribute, are allowed.

Figure 9 on page 26 shows how the different host access rights can be kept in the LDAP directory tree.

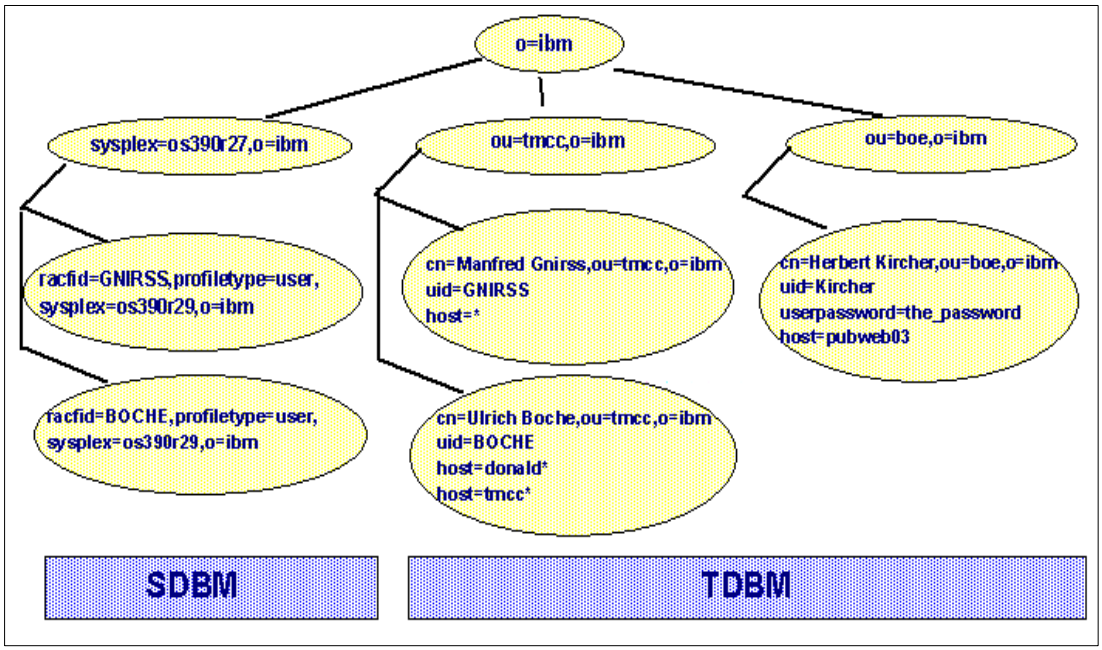


Figure 9 LDAP trees with entries for different host access

In the configuration file of PAM LDAP, *pam_ldap.conf*, you need an additional statement:

```
pam_check_host_attr yes
```

and in the configuration files for the different services (*login, ssh*, *other*, etc.) in */etc/pam.d,* you need to define the desired behavior as a result of the response of the LDAP server to the authentication request. The *auth* statement would contain some more complicated directives such as:

```
auth [success=ok perm_denied=bad default=bad] pam_ldap.so
```

For a detailed description of the possibilities for specifying the desired behavior, check the original documentation of PAM at:

```
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam-4.html
```

Encryption of LDAP communication

If the Linux for zSeries system with the LDAP client and the z/OS LDAP server are not running on the same hardware, for example not running in different partitions of one zSeries machine, then the communication between these systems is performed via the usual external network. If the cabling infrastructure of this external network is not completely and obviously protected, then the communication between the LDAP client and LDAP server should be encrypted because it is used for transport of sensitive data, such as passwords.

The LDAP server has the ability to protect LDAP access with Secure Sockets Layer (SSL) security. When using SSL to secure communication with the LDAP server, the server is configured to provide server and, optionally, client authentication. With server authentication, the LDAP server must have a digital certificate (based on the X.509 standard). This digital certificate is used to authenticate the LDAP server to the LDAP client application. In addition, if the LDAP server is configured to use server and client authentication, and the client sends a digital certificate on the initial SSL handshake, it must be validated by the LDAP server before the secure encrypted communication channel is established between them.

Note that these certificates authenticate the partners of the SSL communication (the z/OS LDAP server and the Linux client); they are *not* the authentication of the Linux for zSeries user!

Setting up the LDAP server for SSL

The following high-level steps are required to enable SSL support of the LDAP server on z/OS:

1. Configure the LDAP server to listen for LDAP requests on the SSL port for server authentication and, optionally, client authentication.
2. Generate the LDAP server private key and server certificate and mark them as the default in the key database, or use the label sslCertificate in the LDAP configuration file.
3. Restart the LDAP server.

The detailed description of these steps can be found in the chapter "Setting up for SSL" in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923.

Setting up the LDAP client on Linux for zSeries for SSL with server authentication

To establish an SSL communication between Linux for zSeries and the LDAP server for native authentication, you need to adapt the PAM LDAP and NSS configuration files. In both files (*pam_nss.conf* and *pam_ldap.conf)* you need a *host* and a *uri ldaps* statement with the hostname of the LDAP server. For example:

```
host os390r27.boeblingen.de.ibm.com
uri ldaps://os390r27.boeblingen.de.ibm.com/
```

Note that the hostname must match the common name in the server certificate. Further, you really need both statements. The *host* statement is used by PAM LDAP and the *uri ldaps* statement is used by the underlying SSL services.

To be able to use SSL with server authentication, you need OPPENLDAP V2 with SSL support on your Linux for zSeries system (for example, this is available with SuSE sles7).

Setting up the LDAP client on Linux/390 for SSL with server and client authentication

If you have PAM LDAP with SSL support available on your Linux/390 system, you can also consider using client authentication. This would require a digital certificate on each Linux client, and these certificates must be known on the LDAP server side. For details, check the documentation for PAM LDAP.

To be able to use SSL with server and client authentication, you need pam_ldap with SSL support on your Linux for zSeries system (for example, this available with SuSE sles7).

Miscellaneous

Providing unique uid numbers

Hint: Linux users should have unique uid numbers. This is especially important if the users have resources on NFS. If you store all user information in a central LDAP directory, you have a good chance to assure uniqueness. When adding users to the LDAP directory, this is not guaranteed. You could write a program that scans the content of the LDAP tree for duplicate uid numbers.

General read access of the complete directory?

A Linux system that uses NSS via LDAP must be able to get all required information for all relevant users from the LDAP directory. If you store more than the minimum required data for NSS in the LDAP directory tree, allow read-access to the POSIX account information, such as *uid, uidnumber, gidnumber, home directory* and *login shell*. Other personal information (such as e-mail address or phone number) can be protected, if it is not intended to have enterprise-wide read access to these data.

User information for users with access to a DMZ server can be stored on a separate LDAP server and database, so that there is a clear separation of data that can be accessed from these systems.

LDAP Client utilities and tools

On the Internet, many utilities and tools are available for simplification of working with LDAP. The LDAP Browser-Editor from Jarek Gawor, for example, can be very helpful. This tool provides a user-friendly interface to LDAP directories with tightly integrated browsing and editing capabilities. You can update directory entries as well as create ldif files out of a directory or import ldif files into an LDAP directory.

Information can be found on the Internet at:

```
http://www-unix.mcs.anl.gov/~gawor/ldap/
```

RACF password expiration

Since we use an LDAP server with RACF interaction for user authentication, we also have the RACF function that user passwords can expire after a certain time period. This time period depends on the individual security policy of the installation; it is set by the RACF administrator.

With native authentication, a login of a Linux user with an expired RACF password is answered by the LDAP server with specific return and reason codes, which indicate that the used password is correct but expired. But the PAM module does not handle this situation appropriately and the user will get an *Access denied* message, which means that the user is not able to log in with the expired password in order to change it using the Linux **`passwd`** service. For practical reasons, it is not a solution for the user to always have to ask the system administrator to reset the RACF password.

To overcome this problem, you have the following possibilities:

1. Disable RACF password expiration.
2. Provide a Web interface for password change.
3. Change the PAM module.
4. Use a workstation program (Linux, Windows, etc.) with an LDAP connection.

These solutions have advantages and disadvantages, as follows:

1. Disabling RACF password expiration is very easy to implement, but in most cases against corporate security policies. This solution can only be regarded as a temporary fix. It should not be seen as a permanent solution.
2. Providing a Web interface where users can change their expired passwords is a rather elegant solution. This page can also be used for a password change in general, independent of password expiration or applications that require RACF passwords.

   On a Web page, the user would provide the current password and the new password (the new password is entered twice and will not be shown on the screen). The content of this

page is used as input for either a CGI program or a servlet. The CGI or servlet takes the userid, the current expired password and the new password to handle the password change request. At first an **ldapsearch** with the userid is issued to get the *distinguished name (DN)* of the user. Then an **ldapmodify** request with the retrieved DN of the user is issued to first *delete* the old and then *add* the new *userPassword* attribute (see also “Native authentication: changing passwords” on page 20). You can use a CGI written in either C, Perl, or REXX. Or, if you want you can write a Java servlet, which is easy to maintain and very flexible. But for a Java servlet you not only need an HTTP server, but also a servlet engine such as IBM WebSphere Application Server or Tomcat.

Note: To be able to change passwords when the current password has expired, you need to install the PTF for APAR OW53591.

3. Changing the PAM module and inserting return and reason code handling to trigger the invocation of the Linux **passwd** service would be a nice solution.
4. A program that connects to the LDAP server and performs a password change with the logic described in 2. has the disadvantage that each user has to install the program on his workstation.

It is evident that for solutions 2., 3. and 4. the network communication has to be protected by encryption (SSL).

At the time of writing, we were in the process of providing a sample solution as described in option 2. for a customer, which we will publish in the future on the ITSO Web site.

Summary and conclusion

We have demonstrated the possibility of setting up an environment for multiple Linux for zSeries systems in cooperation with z/OS to keep all Linux user information, including passwords, on a central z/OS LDAP directory where the password information is protected by RACF.

This offers the advantages of central user administration: Increased security on Linux systems (no user data need be kept locally), RACF protection and password verification, and the high quality of services of the z/OS environment.

When setting up such an environment, existing RACF user information can be used as a base for the initial setup. The password information continues to be protected by RACF. It is also possible to integrate employee information from a corporate-wide directory, or even, under specific circumstances, to modify the corporate-wide directory and use it directly for authentication purposes.

The necessary effort to set up such an environment is minimal:

1. Start with setting up RACF authentication checking for your Linux for zSeries systems.

   You need only install and configure the PAM part and enable RACF to act as SDBM for the z/OS LDAP server. This may be a matter of only a few hours, and you benefit immediately from RACF-protected passwords in a complete solution.
2. Next set up NSS for user identification with an LDAP server and configure DB2 as TDBM backend for the LDAP server.
3. Add the schema and provide user data in TDBM.
4. Enable native authentication.

These four steps can take several days.

User administration is thus handled completely on the central LDAP server, which reduces the user administration tasks (adding, deleting, revoking user accounts, resetting passwords, etc.).

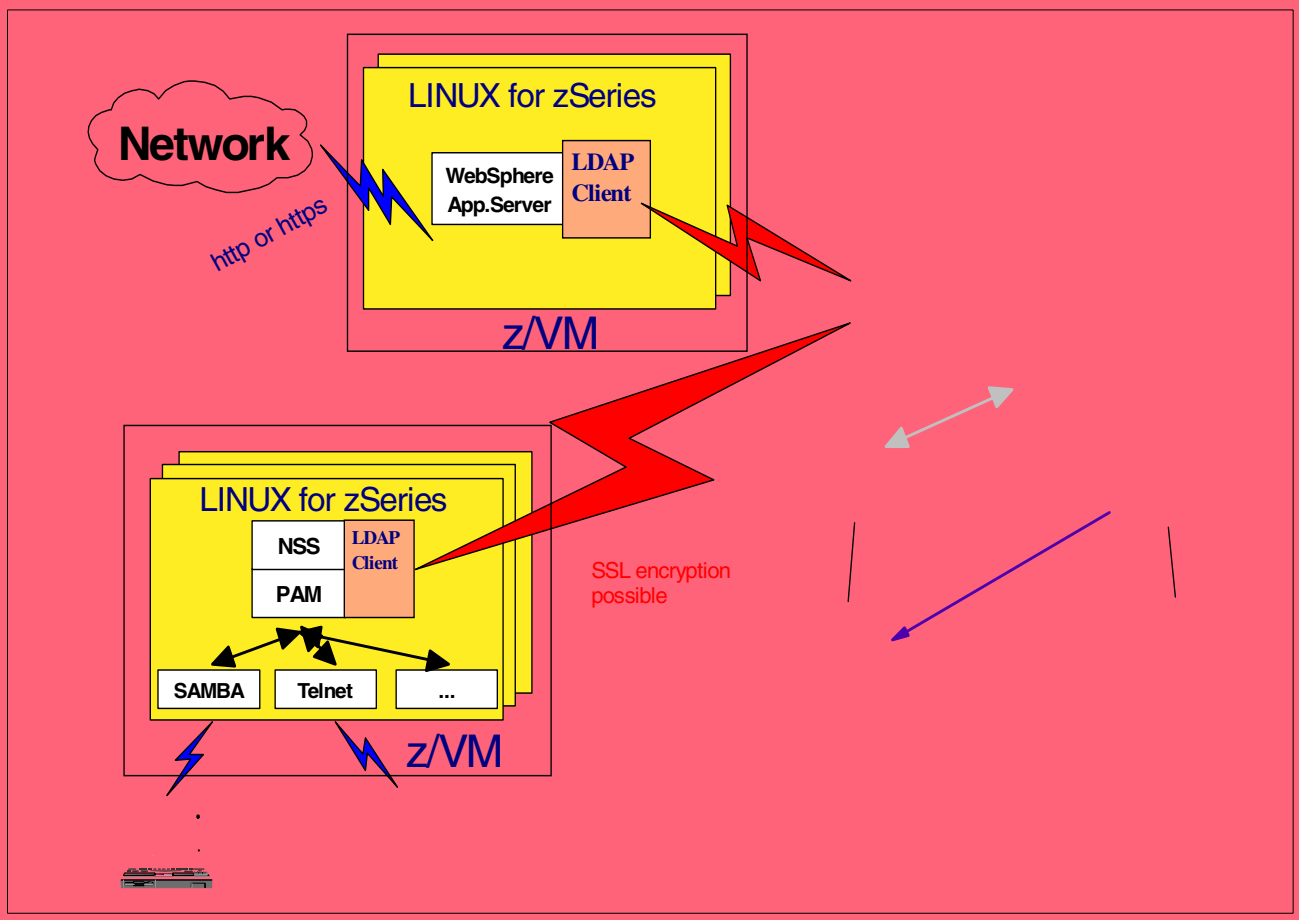
Also, when cloning Linux systems, you need not maintain user accounts on each of the cloned systems. The more Linux systems you have to maintain, the more effective a centralized user management becomes.

If your z/OS system is a member of a Parallel Sysplex, then you gain another big advantage by having a centralized LDAP directory within z/OS: Availability and scalability questions are covered by Parallel Sysplex so that it is not necessary to replicate LDAP directories for availability reasons (as you would have to do in other environments).

Authenticating user information requests passed via the network is not critical because the communication can be protected with SSL, and it will not lead to an awful amount of network traffic because not every request for user information of a Linux system automatically leads to an access of the LDAP server via the network. On the Linux side user information is cached with the Name Service Caching Daemon (nscd) so that subsequent NSS requests can be handled directly out of the local cache.

Organizational hints:

- Keep all the user information in the central LDAP directory, but do keep the definitions and information of the administrative accounts (e.g., *root*) locally on the Linux for zSeries system.


Network
http or https
LINUX for zSeries
WebSphere App.Server
LDAP Client
z/VM
LINUX for zSeries
NSS
PAM
LDAP Client
SAMBA
Telnet
...
z/VM
SSL encryption possible

References

IBM Redbooks

Understanding LDAP, SG24-4986

LDAP Implementation Cookbook, SG24-5110

Other

z/OS Security Server LDAP Server Administration and Use, SC24-5923

Giuseppe Lo Biondo, “LDAP as a Network Information Service”, Istituto Nazionale di Fisica Nucleare, Sezione di Milano, INFN/TC-00/15, September 8, 2000

Referenced Web sites

Web site of PADL Software Pty Ltd contains Open Source products like nss ldap and pam ldap:

`http://www.padl.com`

Alan Altmark, Cliff Laking, “z/VM Security and Integrity.” Technical paper at:

`http://www.ibm.com/servers/eserver/zseries/library/techpapers/gm130145.html`

Web site with latest IBM security evaluation information:

`http://www.ibm.com/servers/eserver/zseries/library/techpapers/gm130145.html`

David “Del.” Elson, “Linux Authentication Using pam_ldap”, Part One, at:

`http://online.securityfocus.com/infocus/1427`

Appendix

Sample configurations

You do not need to manually create the following sample files in your installation. They are already available or will be created automatically. You only have to adapt these files according to your needs.

Example of a PAM configuration file for login (Telnet)

```
#
# The PAM configuration file for the Shadow `login' service
#
# NOTE: If you use a session module (such as kerberos or NIS+)
# that retains persistent credentials (like key caches, etc), you
# need to enable the `CLOSE_SESSIONS' option in /etc/login.defs
# in order for login to stay around until after logout to call
# pam_close_session() and cleanup.
#

# Outputs an issue file prior to each login prompt (Replaces the
# ISSUE_FILE option from login.defs). Uncomment for use
auth       required   pam_issue.so issue=/etc/issue

# Disallows root logins except on tty's listed in /etc/securetty
# (Replaces the `CONSOLE' setting from login.defs)
#auth       requisite  pam_securetty.so

# Disallows other than root logins when /etc/nologin exists
# (Replaces the `NOLOGINS_FILE' option from login.defs)
auth       requisite  pam_nologin.so

# This module parses /etc/environment (the standard for setting
# environ vars) and also allows you to use an extended config
# file /etc/security/pam_env.conf.
# (Replaces the `ENVIRON_FILE' setting from login.defs)
auth       required   pam_env.so

# Authentication via an LDAP server.
auth       sufficient  pam_ldap.so

# Standard Un*x authentication. The "nullok" line allows passwordless
# accounts.
auth       required   pam_unix.so use_first_pass

# This allows certain extra groups to be granted to a user
# based on things like time of day, tty, service, and user.
# Please uncomment and edit /etc/security/group.conf if you
# wish to use this.
# (Replaces the `CONSOLE_GROUPS' option in login.defs)
# auth       optional   pam_group.so

# Uncomment and edit /etc/security/time.conf if you need to set
# time restrainst on logins.
# (Replaces the `PORTTIME_CHECKS_ENAB' option from login.defs
# as well as /etc/porttime)
# account    requisite  pam_time.so
```

```
# Uncomment and edit /etc/security/access.conf if you need to
# set access limits.
# (Replaces /etc/login.access file)
# account  required       pam_access.so

# Standard Un*x account and session
# LDAP Server account and session
account    sufficient pam_ldap.so
account    required   pam_unix.so
session    sufficient pam_ldap.so
session    required   pam_unix.so

# Sets up user limits, please uncomment and read /etc/security/limits.conf
# to enable this functionality.
# (Replaces the use of /etc/limits in old login)
# session    required   pam_limits.so

# Prints the last login info upon succesful login
# (Replaces the `LASTLOG_ENAB' option from login.defs)
#session    optional   pam_lastlog.so

# Prints the motd upon succesful login
# (Replaces the `MOTD_FILE' option in login.defs)
session    optional   pam_motd.so

# Prints the status of the user's mailbox upon succesful login
# (Replaces the `MAIL_CHECK_ENAB' option from login.defs). You
# can also enable a MAIL environment variable from here, but it
# is better handled by /etc/login.defs, since userdel also uses
# it to make sure that removing a user, also removes their mail
# spool file.
session    optional   pam_mail.so standard noenv

# The standard Unix authentication modules, used with NIS (man nsswitch) as
# well as normal /etc/passwd and /etc/shadow entries. For the login service,
# this is only used when the password expires and must be changed, so make
# sure this one and the one in /etc/pam.d/passwd are the same. The "nullok"
# option allows users to change an empty password, else empty passwords are
# treated as locked accounts.
#
# (Add `md5' after the module name to enable MD5 passwords the same way that
# `MD5_CRYPT_ENAB' would do under login.defs).
#
# The "obscure" option replaces the old `OBSCURE_CHECKS_ENAB' option in
# login.defs. Also the "min" and "max" options enforce the length of the
# new password.

password   sufficient pam_ldap.so
password   required   pam_unix.so nullok obscure min=4 max=8  md5

# Alternate strength checking for password. Note that this
# requires the libpam-cracklib package to be installed.
# You will need to comment out the password line above and
# uncomment the next two in order to use this.
# (Replaces the `OBSCURE_CHECKS_ENAB', `CRACKLIB_DICTPATH')
#
# password required       pam_cracklib.so retry=3 minlen=6 difok=3
# password required       pam_unix.so use_authtok nullok md5
# Test M. Gnirss 4/5/2002 automatically creation of an homedirectory
# for a user who is authenticated and identified via LDAP
```

```
session required pam_mkhomedir.so skel=/etc/skel/ umask=0077
```

Example of a PAM configuration file for ssh

```
#%PAM-1.0
auth       required     pam_nologin.so
auth       sufficient   pam_ldap.so
auth       required     pam_unix.so use_first_pass
auth       required     pam_env.so # [1]

account    sufficient   pam_ldap.so
account    required     pam_unix.so

session    required     pam_unix.so
#session    required     pam_ldap.so
session    optional     pam_lastlog.so # [1]
session    optional     pam_motd.so # [1]
session    optional     pam_mail.so standard noenv # [1]
session    required     pam_mkhomedir.so skel=/etc/skel/ umask=0077
session    required     pam_limits.so

password   sufficient   pam_ldap.so
password   required     pam_unix.so

# Alternate strength checking for password. Note that this
# requires the libpam-cracklib package to be installed.
# You will need to comment out the password line above and
# uncomment the next two in order to use this.
#
# password required       pam_cracklib.so retry=3 minlen=6 difok=3
# password required       pam_unix.so use_authtok nullok md5
```

Example of a PAM LDAP client configuration file

```
###DEBCONF###
# the configuration of this file will be done by debconf as long as the
# first line of the file says '###DEBCONF###'
#
# you should use dpkg-reconfigure to configure this file
#
# @(#)$Id: ldap.conf,v 1.24 2001/09/20 14:12:26 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# PADL Software
# http://www.padl.com
#

# Your LDAP server. Must be resolvable without using LDAP.
# *** this is our host with the z/OS LDAP server ***
host os390r27.boeblingen.de.ibm.com

# The distinguished name of the search base.
# *** The following line contains the search base used for native auth. ***
base o=ibm
# *** the following line contains the search base for RACF (SDBM) auth. ***
#base profiletype=user,sysplex=os390r27,o=ibm

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
```

```
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=example,dc=net
#  *** The following line can be used for RACF (SDBM authentication) ***
#  *** user lxlogin can bind to SDBM backend and perform a LDAP search ***
# *** password of user lxlogin is needed for bind to SDBM , see below ***
#binddn racfid=lxlogin,profiletype=user,sysplex=os390r27,o=ibm

# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret
# *** The following line can be used for RACF (SDBM authentication) ***
# *** Password of user lxlogin is necessary, if bind to SDBM, see above ***
#bindpw the_password

# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn racfid=ldapadm,profiletype=user,sysplex=os390r27,o=ibm
#rootbinddn cn=this

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search time limit
time limit 30

# Bind time limit
#bind_time limit 30

# Filter to AND with uid=%s
#pam_filter objectclass=account

# The user ID attribute (defaults to uid)
# *** For Authentication with RACF (SDBM) use racfid ***
#pam_login_attribute racfid
#pam_login_attribute uid

# Search the root DSE for the password policy (works
# with Netscape Directory Server)
#pam_lookup_policy yes

# Check the 'host' attribute for access control
# Default is no; if set to yes, and user has no
```

```
# value for the host attribute, and pam_ldap is
# configured for account management (authorization)
# then the user will not be allowed to login.
#pam_check_host_attr yes

# Group to enforce membership of
#pam_groupdn cn=PAM,ou=Groups,dc=example,dc=net

# Group member attribute
#pam_member_attribute uniquemember

# Specify a minium or maximum UID number allowed
#pam_min_uid 0
#pam_max_uid 0

# Template login attribute, default template user
# (can be overriden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody

# HEADS UP: the pam_crypt, pam_nds_passwd,
# and pam_ad_passwd options are no
# longer supported.

# Do not hash the password at all; presume
# the directory server will do it, if
# necessary. This is the default.
#pam_password clear
# *** The following line has been inserted for test of temporary fix ***
# *** to test password change if backend is z/OS LDAP server ***
# *** with TDBM as backend and Native Authentication active ***
pam_password zos

# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
#pam_password crypt

# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
#pam_password nds

# Update Active Directory password, by
# creating Unicode password and updating
# unicodePwd attribute.
#pam_password ad

# Use the OpenLDAP password change
# extended operation to update the password.
#pam_password exop

# configure --enable-mssfu-schema is no longer supported.
# For MSSFU now do:
#pam_login_attribute msSFUName
#pam_filter objectclass=User
```

```
#pam_password ad

# configure --enable-authpassword is no longer supported
# For authPassword support, now do:
#pam_password nds

# For IBM SecureWay support, do:
#pam_login_attribute userName
#pam_filter objectclass=aixAccount
#pam_password clear
```

Example of a Names Service Switch functionality file

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         ldap compat
group:          ldap compat
shadow:         ldap compat

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Example of an LDAP client configuration file for NSS

```
###DEBCONF###
# the configuration of this file will be done by debconf as long as the
# first line of the file says '###DEBCONF###'
#
# you should use dpkg-reconfigure libnss-ldap to configure this file.
#
# @(#)$Id: ldap.conf,v 2.30 2001/09/22 10:57:56 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# PADL Software
# http://www.padl.com
#

# Your LDAP server. Must be resolvable without using LDAP.
host os390r27.boeblingen.de.ibm.com

# The distinguished name of the search base.
base o=ibm

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
```

```
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=padl,dc=com

# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret

# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn cn=manager,dc=padl,dc=com

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search time limit
time limit 30

# Bind time limit
#bind_time limit 30

# Idle time limit; client will close connections
# (nss_ldap only) if the server has not been contacted
# for the number of seconds specified below.
#idle_time limit 3600

# RFC2307bis naming contexts
# Syntax:
# nss_base_XXXbase?scope?filter
# where scope is {base,one,sub}
# and filter is a filter to be &'d with the
# default filter.
# You can omit the suffix eg:
# nss_base_passwdou=People,
# to append the default base DN but this
# may incur a small performance impact.
#nss_base_passwdou=People,dc=padl,dc=com?one
#nss_base_shadowou=People,dc=padl,dc=com?one
#nss_base_groupou=Group,dc=padl,dc=com?one
#nss_base_hostsou=Hosts,dc=padl,dc=com?one
#nss_base_servicesou=Services,dc=padl,dc=com?one
#nss_base_networksou=Networks,dc=padl,dc=com?one
#nss_base_protocolsou=Protocols,dc=padl,dc=com?one
#nss_base_rpcou=Rpc,dc=padl,dc=com?one
#nss_base_ethersou=Ethers,dc=padl,dc=com?one
#nss_base_netmasksou=Networks,dc=padl,dc=com?ne
```

```
#nss_base_bootparamsou=Ethers,dc=padl,dc=com?one
#nss_base_aliasesou=Aliases,dc=padl,dc=com?one
#nss_base_netgroupou=Netgroup,dc=padl,dc=com?one

# attribute/objectclass mappin'
# Syntax:
#nss_map_attributerfc2307attributemapped_attribute
#nss_map_objectclassrfc2307objectclassmapped_objectclass

# configure --enable-nds is no longer supported.
# For NDS now do:
#nss_map_attribute uniqueMember member

# configure --enable-mssfu-schema is no longer supported.
# For MSSFU now do:
#nss_map_objectclass posixAccount User
#nss_map_attribute uid msSFUNaie
#nss_map_attribute uniqueMember posixMember
#nss_map_attribute userPassword msSFUPassword
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_objectclass posixGroup Group
#nss_map_attribute cn msSFUName

# Alternatively, if you wish to equivalence W2K and POSIX
# groups, change the uniqueMember mapping line to:
#nss_map_attribute uniqueMember member

# configure --enable-authpassword is no longer supported
# For authPassword support, now do:
#nss_map_attribute userPassword authPassword

# For IBM AIX SecureWay support, do:
#nss_map_objectclass posixAccount aixAccount
#nss_base_passwd ou=aixaccount,?one
#nss_map_attribute uid userName
#nss_map_attribute gidNumber gid
#nss_map_attribute uidNumber uid
#nss_map_attribute userPassword passwordChar
#nss_map_objectclass posixGroup aixAccessGroup
#nss_base_group ou=aixgroup,?one
#nss_map_attribute cn groupName
#nss_map_attribute uniqueMember member
```

Extracts of z/OS slapd.conf

```
# @(#)70
# 1.30
# 4/2/01 16:08:34
# ********************************************************************
# This file is shipped in code page IBM-1047 and must remain in
# code page IBM-1047.
# ********************************************************************
#
# ********************************************************************
# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2000
# ********************************************************************
#
```

```
# ********************************************************************
# This was generated by ldapcnf on Tue Feb 12 13:39:31 GMT 2002.
# This was generated by ldapcnf with the input
# file: '/u/ldapadm/ldap.profile'.
# WARNING: Any manual updates to this file will be lost
# if ldapcnf is executed again and the OUTPUT_DATASET option is
# set to BOCHE.CNFOUT.
# ********************************************************************
... some lines deleted ...
include /usr/lpp/ldap/etc/schema.system.at
include /usr/lpp/ldap/etc/schema.system.oc
include /usr/lpp/ldap/etc/schema.IBM.at
include /usr/lpp/ldap/etc/schema.IBM.oc
include /usr/lpp/ldap/etc/schema.user.at
include /usr/lpp/ldap/etc/schema.user.oc
... some lines deleted ...
#-------------------------------------------------------------------
#
# listen <ldapURL>
#
# Description:
#   This option will be used to bind non-SSL connections to the LDAP
#   server for a particular port on a hostname/IP address.
#
#-------------------------------------------------------------------
listen ldap://:389
... some lines deleted ...
# ---------------------------------------
#
# adminDN <distinguished name>
#
# Description:
#   This option specifies the distinguished name (DN) of the
#   administrator for this LDAP server.
#
# ---------------------------------------
adminDN "racfid=LDAPADM,profiletype=user,sysplex=os390r27,o=ibm"
# ---------------------------------------
#
# adminPW <password>
#
# Description:
#   This option specifies the password for the administrator (adminDN)
#   for this server.
#
# ---------------------------------------
#adminPW "%ADMINPW%"
... some lines deleted ...
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
#
#  SDBM-specific CONFIGURATION SETTINGS
#
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
database sdbm GLDBSDBM
# ---------------------------------------
#
# suffix <toplevelname>
#
```

```
# Description:
#   This option specifies the suffix for the SDBM backend
#
# ----------------------------------------
suffix "sysplex=os390r27,o=ibm"
... some lines deleted ...
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
#
#  TDBM-specific CONFIGURATION SETTINGS
#
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
database tdbm GLDBTDBM
# ----------------------------------------
#
# suffix <toplevelname>
#
# Description:
#   This option specifies the suffix for the TDBM backend
#
# ----------------------------------------
suffix "o=ibm"
# ----------------------------------------
#
# servername <name>
#
# Description:
#   The option specifies the name of the DB2 data source.
#   This name is also referred to as the location name.
#
# ----------------------------------------
servername OS390R27
# ----------------------------------------
#
# dbuserid <userid>
#
# Description:
#   This option specifies the userid that was used to create
#   DB2 tables that are used by the TDBM database. DB2 table
#   names are prefixed with the userid that was used to create
#   the tables.
#
# ----------------------------------------
dbuserid LDAPADM
# ----------------------------------------
#
# databasename <name>
#
# Description:
#   This option specifies the name of the database that was
#   created to hold the tables that the LDAP server uses.
#
# ----------------------------------------
databasename TDBMDB
# ----------------------------------------
#
# dsnaoini <data_set_name|filename>
#
# Description:
```

```
#   This option specifies the name of CLI init file also
#   known as dsnaoini.
#
# --------------------------------------
dsnaoini /u/ldapadm/dsnaoini
... some lines deleted ...
#
# Description:
#   This option defines the password encryption used by the LDAP
#   Server.
#
# --------------------------------------
pwEncryption  crypt
... some lines deleted ...
#-----------------------------------------------------------------------
#
# useNativeAuth <selected|all|off>
#
# Description:
#   This option enables native authentication in the TDBM
#   backend. If the value is:
#       selected - only entries within native subtrees with the
#                  ibm-nativeId attribute will use native authentication.
#       all      - all entries within native subtrees will use native
#                  authentication. These entries can contain the
#                  ibm-nativeId or uid attribute to specify the RACF ID.
#       off      - no entries will participate in native authentication
#
#-----------------------------------------------------------------------
useNativeAuth all
#-----------------------------------------------------------------------
#
# nativeUpdateAllowed <on|off>
#
# Description:
#   This option enables native password changes in RACF through
#   the TDBM backend.
#-----------------------------------------------------------------------
nativeUpdateAllowed on
#-----------------------------------------------------------------------
#
# nativeAuthSubtree <all|distinguished name>
#
# Description:
#   This option specifies the distinguished name of the parent entry
#   of a subtree where all of its entries participate in Native
#   Authentication.
#
#   If this parameter is omitted, contains no value, or is set
#   to 'all' then the entire directory will be subject to
#   native authentication.
#
#-----------------------------------------------------------------------
nativeAuthSubtree all
```

Miscellaneous

Proposal for a patch for native password changes

The following shows a possible patch to include the new parameter `zos` for the session statement in the *pam_ldap* configuration file to enable a password change via an **`ldapmodiy`** with *delete* followed by an *add*. This syntax is required for changing passwords with native authentication.

Insert a new constant in pam_ldap.h for the new keyword. In pam_ldap.c, insert a new check for the new keyword and a `case` statement for the new behavior.

The following is a proposed patch. The lines prefixed with "+" are new or modified in the *pam_ldap* code.

```
--- libpam-ldap-144.orig/pam_ldap.c
+++ libpam-ldap-144/pam_ldap.c
@ -843,6 +848,8 @@
       result->password_type = PASSWORD_AD;
     else if (!strcasecmp (v, "exop"))
       result->password_type = PASSWORD_EXOP;
+    else if (!strcasecmp (v, "zos"))
+      result->password_type = PASSWORD_ZOS;
   }
       else if (!strcasecmp (k, "pam_crypt"))
   {
@@ -2408,6 +2451,27 @@

       break;

+    case PASSWORD_ZOS:
+      /* z/OS LDAP with Native Authentication requires a  DELETE followed by an ADD */
+      strvalsold[0] = (char *) old_password;
+      strvalsold[1] = NULL;
+      strvalsnew[0] = (char *) new_password;
+      strvalsnew[1] = NULL;
+
+      mod.mod_vals.modv_strvals = strvalsold;
+      mod.mod_type = (char *) "userPassword";
+      mod.mod_op = LDAP_MOD_DELETE;
+
+      mod2.mod_vals.modv_strvals = strvalsnew;
+      mod2.mod_type = (char *) "userPassword";
+      mod2.mod_op = LDAP_MOD_ADD;
+
+      mods[0] = &mod;
+      mods[1] = &mod2;
+      mods[2] = NULL;
+
+      break;
+
     case PASSWORD_NDS:
       /* NDS requires that the old password is first removed */
       strvalsold[0] = (char *) old_password;
--- libpam-ldap-144.orig/pam_ldap.h
+++ libpam-ldap-144/pam_ldap.h
@ -100,6 +100,7 @@
 #define PASSWORD_NDS     3
 #define PASSWORD_AD      4
 #define PASSWORD_EXOP    5
```

```
+#define PASSWORD_ZOS     6
     int password_type;
     /* min uid */
     uid_t min_uid;
```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks(logo)™ 	Parallel Sysplex®	SP™
AIX®	PR/SM™	z/OS™
DB2®	PR/SM™	z/VM™
Home Director™	RACF®	zSeries™
IBM®	S/390®	
OS/390®	SecureWay®	

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.